AUTOMATIC SURFACE CRACK DETECTION IN CONCRETE STRUCTURES USING OTSU THRESHOLDING AND MORPHOLOGICAL OPERATIONS

Sattar Dorafshan
Graduate Research Assistant
Utah State University
Department of Civil and Environmental Engineering

Marc Maguire
Assistant Professor
Utah State University
Department of Civil and Environmental Engineering

Xiaojun Qi
Professor
Utah State University
Department of Computer Science

UTC Report 01-2016

April 2016

1. Report No. UTC Report 01-2016	**2. Government Accession No.** --	**3. Recipient's Catalog No.** --
4. Title and Subtitle Automatic surface crack detection in concrete structures using otsu thresholding and morphological operations		**5. Report Date** August 2016 **6. Performing Organization Code** --
7. Author(s) Sattar Dorafshan, Marc Maguire, Xiaojun Qi		**8. Performing Organization Report No.** UTC Report 01-2016
9. Performing Organization Name and Address Utah State University 4110 Old Main Hill Logan, UT 84322-4110		**10. Work Unit No.** -- **11. Contract or Grant No.** --
12. Sponsoring Agency Name and Address Utah Transportation Center Utah State University 4110 Old Main Hill Logan, UT 84322-4110		**13. Type of Report and Period Covered** Final Report September 2015 – August 2016 **14. Sponsoring Agency Code** --
15. Supplementary Notes --		

16. Abstract

Concrete cracking is a ubiquitous phenomenon, present in all types of concrete structures. Identifying and tracking the amount and severity of cracking is paramount to evaluating the current condition and predicting the future service life of a concrete asset. Concrete cracks can indicate reinforcement corrosion, the development of spalls or changing support conditions. Therefore, monitoring cracks during the life span of concrete structures has been an effective technique to evaluate the level of safety and preparing plans for future appropriate rehabilitation.

One growing technique are unmanned inspections using Unmanned Aerial Vehicles (UAV). UAVs are drones equipped with cameras, sensors, GPS, etc. RGB images (color images in Red, Green and Blue color space) are obtained from a camera mounted on a UAV flying around the structure, to detect cracks and other defects.

Each image captured by UAV needs to be evaluated to track the crack formations. To save time, this task can be done by applying image processing techniques to automatically detect and report cracks rather than using a human to identify them. In addition, processing RGB images with sufficient information, such as the distance of camera to surface for each picture, will provide the dimension of the cracks (length and width).

The report consists of the following sections: A literature review of image processing techniques used in structural health monitoring and other fields of interest is provided in chapter 2. The Proposed method to identify cracks is demonstrated in Chapter 3. Experimental results, conclusion and future work are presented in Chapter 4. Appendix A includes the processed images using the proposed method and Appendix B includes the comparison between Talab's method and the proposed method. In Appendix C, a "readme" file is given to run the program, and finally Appendix D shows the Matlab Code.

17. Keywords: Crack Detection, Concrete Pavement, Image Processing and Filtering	**18. Distribution Statement** No restriction

19. Security Classifi. (of this report) Unclassified	**20. Security Classifi. (of this page)** Unclassified	**21. No. of Pages:** 151	**22. Price** --

CONTENTS

Table of Figures

1. INTRODUCTION

Concrete cracking is a ubiquitous phenomenon, present in all types of concrete structures. Identifying and tracking the amount and severity of cracking is paramount to evaluating the current condition and predicting the future service life of a concrete asset. Concrete cracks can indicate reinforcement corrosion, the development of spalls or changing support conditions. Therefore, monitoring cracks during the life span of concrete structures has been an effective technique to evaluate the level of safety and preparing plans for future appropriate rehabilitation.

Using human visual inspection is the oldest and most reliable method to recognize concrete cracks. However, using human inspectors is time consuming, expensive, and can pose risks to human safety. The inspection routine also generally requires closure to traffic, for instance, on a bridge deck or pavement, to allow the inspection. Thus, unmanned inspection methods have been on the rise for the past decade, letting the inspection be done in a more efficient manner. One of the techniques in unmanned inspection is using Unmanned Aerial Vehicles (UAV). UAVs are drones equipped with cameras, sensors, GPS, etc. RGB images (color images in Red, Green and Blue color space) are obtained from a camera mounted on a UAV flying around the structure, to detect cracks and other defects (See Figure 1).

Each image captured by UAV needs to be evaluated to track the crack formations. To save time, this task can be done by applying image processing techniques to automatically detect and report cracks rather than using a human to identify them. In addition, processing RGB images with sufficient information, such as the distance of camera to surface for each picture, will provide the dimension of the cracks (length and width).

Figure 1. Picture of a UAV

The crack detection in this report was built on the work of Talab et al. (Talab, et al. 2015) with modifications. RGB images are captured from concrete surfaces using a 12-megapixel Nikon digital camera.

The report consists of the following sections: A literature review of image processing techniques used in structural health monitoring and other fields of interest is provided in chapter 2. The Proposed method to identify cracks is demonstrated in Chapter 3. Experimental results, conclusion and future work are presented in Chapter 4. Appendix A includes the processed images using the proposed method and Appendix B includes the comparison between Talab's method and the proposed method. In Appendix C, a "readme" file is given to run the program, and finally Appendix D shows the Matlab Code.

Forty-one pictures have been taken from the Riverwood Conference Hall and Utah State University Library in Logan, UT. A computer program has been written in Matlab2015a and executed on a PC with the following configuration: Intel® Core™2 Quad Q6600 @ 2.4 GHz 2.39 GHz, RAM of 6 GB (5.87 usable), and System type of 64-bit operating system. This configuration

is modest compared to the many available computers. Its capabilities are similar to the ones that could be housed on a UAV now, or in the near future.

2. LITERATURE REVIEW

Image processing techniques (in particular crack detection techniques) are divided into several categories based on the nature of calculation and the intuitive approach to process images. Each image consists of pixels. These pixels are presented by integer numbers from 0 to 255 (unassigned integer 8 bits variable). If the operations are conducted directly on the pixel's values, the technique is accomplished in the spatial domain (or real-time). However, it is not always possible to achieve desirable outcome in the spatial domain. In this case, a transformation (usually Fourier or wavelet transformation) is performed on the image. The outcome image is processed in the frequency domain, after which an inverse transformation is applied to reconstruct the image in the spatial domain.

Another category can be defined based on the training algorithm to solve a particular problem. Some of the techniques directly deal with the image without any training. They do not require any training image and several appropriate image processing techniques to find cracks in the images based on representative properties of the cracks in each new image. On the contrary, some techniques first construct a database, which is divided into two subsets: the subset with cracked images, and the subset with un-cracked images. For each subset, a characteristic index, such as the shape factor in form of a decision line (line because there are two subsets in this case) is defined. The training database Having enough images in a training dataset, the program will be able to compare a new image characteristic index with the decision line and assign it to one of the subsets. Neural network and clustering methods are among the most popular training techniques to learn the decision line.

In the area of image-based crack detection, two approaches have been applied. Dye based detection is based on injection of a particular dye solution such as fluorescent on the surface of concrete or other materials. Once the dye has been applied, the surface will be scanned with ordinary, or special, cameras depending on the solution's type and color. Dye based methods were used in the past for detecting micro-cracks in different types of materials (concrete, ceramic tiles, etc.). The other approach is to find the cracks based on the color image without dye solution. This method is currently more popular since it is less costly and less time-consuming.

Surface crack detection techniques have been used to inspect the cracks in different materials. One of these attempts can be seen in the work of Kittler et al. (Kittler, Marik, et al. 1994). They proposed a training-based algorithm to get samples of the regions, which are void of defects in the spatial domain. When the training is done, the new pictures were analyzed for the presence of any defects as well as finding the location of the defects. Clustering techniques and morphological operations were performed to analyze the images. Figure 2 shows the experimental output of this method.

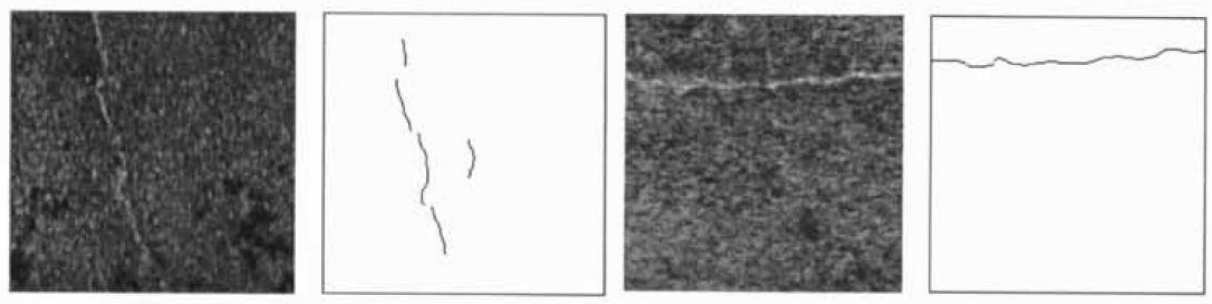


Figure 2. Crack detection on ceramic tiles (Kittler, Marik, et al. 1994)

The inspection of cement-based materials for micro-cracks and other microdefects (porosity, air bubbles) has been done using image processing techniques (Ammouchea, et al. 2000). The prepared surface is impregnated with a red dye solution. Then the excessive dye is removed by polishing the sample surface. The main reason to use the dye solution is to penetrate

the porous regions of the surface. After the dye injection, 256 × 265-pixel color images are taken of the surface. On the image, the following operation is carried out:

$$O(x,y) = max\begin{cases} R(x,y) - G(x,y) \\ 0 \end{cases} \quad (2\text{-}1)$$

$O(x,y)$ stands for the intensity of the output gray-level image at (x,y) coordinate and R and G indicate intensities at (x,y) coordinate in the red and green channel of the original image, respectively. The pixels corresponding to defects will be significantly clearer. Then, using a thresholding value, the gray-level image is converted to the binary image. The value of the thresholding is evaluated using the entropy maximization method. Entropy maximization method is based on the maximization of the entropy function Φ of the gray-levels pixels.

$$\Phi(k) = -\left[\sum_{min}^{k}\left(\frac{p_i}{w_0}\right).log\left(\frac{p_i}{w_0}\right) + \sum_{k+1}^{max}\left(\frac{p_i}{w_1}\right).log\left(\frac{p_i}{w_1}\right)\right] \quad (2\text{-}2)$$

Where p_i is the probability for a given pixel to have an intensity of i ($p_i = N_i/N$), N_i is the number of pixels having the i intensity, and N is the total number of pixels. w_0 and w_1 are the lower and upper bounds of the probability to find a pixel intensity, respectively. They are computed as:

$$(k)\left[w_0 = \sum_{min}^{k} p_i\,;\; w_0 = \sum_{kH}^{max} p_i\right] \quad (2\text{-}3)$$

A lower bound thresholding operation is performed based on the threshold to obtain binary image. The binary image is then post-processed using two morphological operations of eroding and rebuilding in order to eliminate noisy and small objects (less than 10 pixels). However, a second series of operations are needed to obtain the shape factor, which can be used to decide whether the pixels in a binary image are associated with cracks or other defects (porous zone, voids, etc.) To do so, a dimensionless packing density index F_c is defined:

$$F_c = \frac{A_{ob}}{A_{cc}} \quad (2\text{-}4)$$

Where A_{ob} is the object area and A_{cc} is the area of its circumscribed circle for which its diameter is the maximum value of Ferret's diameter, computed at every 10 degrees. F_c is equal to 1 for a circle and tends toward 0 for a very elongated object (cracks). Figure 3 shows the intermediate results obtained in this method for an original image containing cracks.

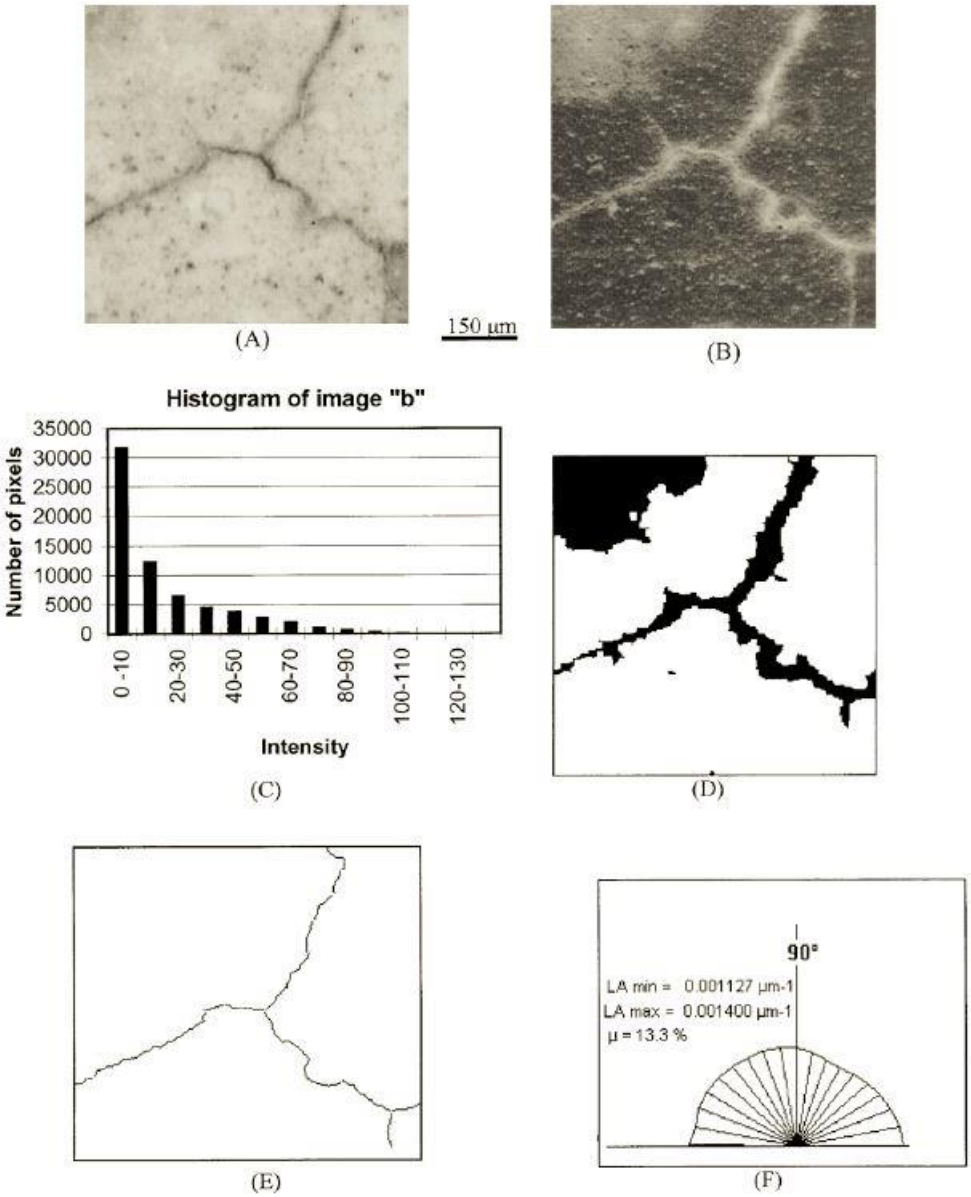


Figure 3. (A) Original color image; (B) grey level-enhanced image; (C) histogram of image B; (D) binary image after thresholding; (E) treated microcrack network; and (F) rose of specific crack length (Ammouchea, et al. 2000)

Another attempt to find cracks on concrete surface using image-processing techniques can be found in the work of Abdel-Ghader (Ikhlas Abdel-Qader, Osama Abudayyeh and Kelly 2003).

They implemented and compared four crack detection methods including fast Haar transform (FHT), fast Fourier transform (FFT), Sobel edge detector, and Canny edge detector based methods, using 50 concrete images from a bridge deck (25 pictures with cracks, 25 pictures without cracks). Their study showed that FHT was the most effective method among the four methods investigated in the study. FHT, (a simplified wavelet transformation) decomposes the image into low and high frequency components, which are basically. The mother wavelet Ψ(t) of Haar function and scaling function Φ(t) are defined as:

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \\ 0 & elsewhere \end{cases} \quad (2\text{-}5)$$

$$\phi(t) = \begin{cases} 1 & 0 \le t < 1 \\ 0 & elsewhere \end{cases} \quad (2\text{-}6)$$

The relationship between Haar wavelet and scaling function and scaling function with itself is presented by:

$$\psi(t) = \phi(2t) - \phi(2t-1) \quad (2\text{-}7)$$
$$\phi(t) = \phi(2t) + \phi(2t-1) \quad (2\text{-}8)$$

To detect cracks, the three quadrants of the first level transformation i.e. 2, 3, and 4 portions, passed through a high pass filter to reduce noises and then combined together to get a magnitude of images. An empirical thresholding value is then used to decide whether crack exists or not.

Fast Fourier transformation (FFT) can be used to derive the frequency component of the image intensity. The transformation and the inverse transformation were carried out using:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)} \quad (2\text{-}9)$$
$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)e^{-2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)} \quad (2\text{-}10)$$

The edges in the frequency domain (transformed image) can be identified based on the brightness.

The Sobel edge detector is a convolution filter defined for vertical and horizontal edges in images:

$$G_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}, \; G_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} \quad (2\text{-}11)$$

The Sobel algorithm detects the change in the image intensity in neighboring pixels. Filtering was done by using the convolution operation for each filter (x direction and y direction) separately. Then the two filtered images combined together at each location and the result is a gray-level image with enhanced edges. The Canny edge detector is also a convolution filter. Using the smoothing techniques before the convolution, the algorithm is less likely to be fooled by noises. In addition, the thresholding value is used to see if a pixel belongs to an edge region or not. The default value for thresholding is $\sqrt{4 \times mean(Fim)}$, where *Fim* is the root of the sum of the square of derivatives in both directions *x* and *y*.

Optical fluorescent microscopy was used to detect cracks in the concrete specimen (Litorowicz 2006). The specimen's surface was grounded and polished in order to remove roughness. The fluorescent dye was then used to cover the concrete surfaces. The specimens were subjected to ultraviolet light using a Nikon optical microscope at a magnification of 10 times. The images were captured using a Sony DXC-950P video camera. The image processing operation was carried out by Image Pro Plus analysis software. The crack network was identified by converting the RGB image into a binary image using a segmentation method. The threshold level was calculated based on the effect of the dye on the surface. To distinguish crack patterns in the concrete specimen, several parameters were used: Angle between the vertical axis and the major axis of the ellipse covering the connected component; Area of each connected component;

Dendritic length, which is the total length of all the one-pixel thick branches; Area Ratio (i.e. of the area of the counted object and the entire area of the active image); Radius Ratio (i.e. the ratio of max radius and min radius for each connected component) and Roundness which is calculated for each object using $perimeter^2/4\pi^2 \times area.$

Percolation-based image processing techniques were developed to detect and measure cracks (Yamaguchi, Nakamura, et al. 2008) and (Yamaguchi and Hashimoto 2010). Percolation is a physical model based on natural phenomenon of permeation of the liquid. The percolation begins at an initial point and spreads towards neighbors according to the probability of *p*. The point in the nearest neighborhood exhibiting the largest value when *p* is percolated, and by repetition, the region grows towards boundaries. The first step was to define a structuring element as a local window. The window was set on the initial point and the initial thresholding value was set to the value of the initial pixel brightness. The threshold was updated by:

$$Case\ 1:\ T = max\left(max_{p\epsilon D_p}\left(V(p)\right), T\right) + w \quad (2\text{-}12)$$

$$Case\ 2:\ T = min\left(min_{p\epsilon D_p}\left(V(p)\right), T\right) + w \quad (2\text{-}13)$$

Where V(p) is the brightness of pixel *p* and *w* is used to accelerate the percolation. Then the following definitions were made. Case 1: the region D_c is defined as the region consisting of eight neighbor pixels of D_p, and D_p is the percolated region, the pixels with brightness below the threshold T were considered a part of D_p. Case 2: The neighbor pixels having brightness greater than the threshold in D_c were regarded as a part of D_p. When D_p reached the boundary of the local window, the percolation process was terminated, and the process was repeated again. Finally, D_p was constructed by the percolation process as a cluster. The size of the local window plays a major role in crack detection using this method. A scalable window was used to automatically determine the size of the local window. Cracks were assumed to be a percolated region, and using this

technique, this region should be recognized and measured. To evaluate whether a region belongs to crack region, the factor of circularity (F_c) was defined:

$$F_c = \frac{4\times C_{count}}{\pi \times C_{max}^2} \quad (2\text{-}14)$$

Where C_{count} is the number of pixels in D_p and C_{max} is the maximum length of D_p. F_c varies from 0 to 1 and is closer to 1 when the region is more circular. For cracks, this number should be close to 0. The brightness of the focal pixel in an output image is associated with F_c value by setting it to $F_c \times 255$. The percolation process was carried out for every pixel in the image and crack pixels were identified.

A robust automated image processing method was applied to crack detection on concrete surfaces by Nishikawa et. al. (Nishikawa, et al. 2012). The proposed method consisted of two stages: 1. Automatic construction of a filter for crack detection and 2. Noise removal and determination of indistinct cracks using an iterative process around the cracked regions. The first stage was carried out by the generic algorithm, where every genetic individual was represented as a tree structure. Using a system based on generic programming, several simple image filters were converted into a tree structure, and numerous image combinations were used to train the program (Aoki and Nagao 1999). A schematic illustration of the tree structure concept is shown in Figure 4.

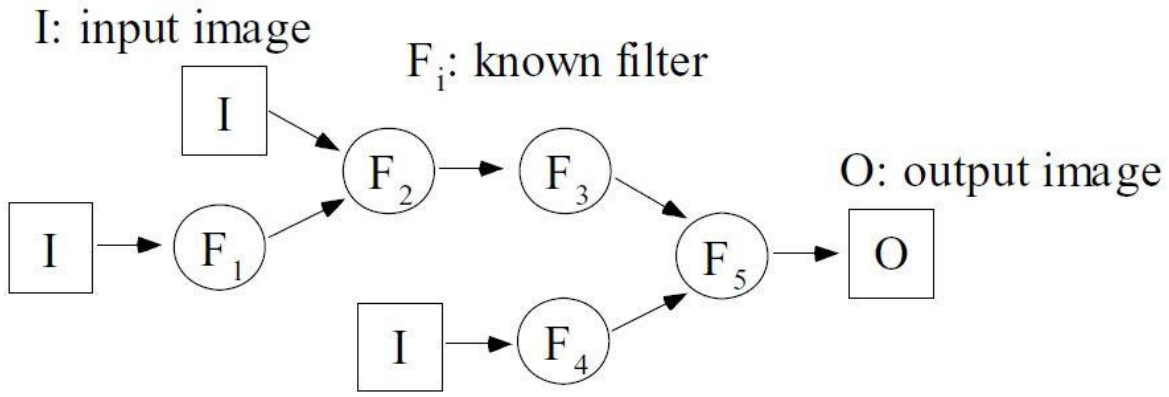


Figure 4. The construction of a tree structures with multiple filters (Aoki and Nagao 1999).

To train the program, a combination generated by generic programming was computed, and the results were compared to the target images that have been manually generated by visual inspection. The comparison was between the distinct distribution of brightness in the output image and the corresponding target image. The following formula was used as a measure of the filter performance:

$$E = \frac{1}{N}\sum_{k=1}^{N}\left[R_k \frac{\sum_{i=1}^{w_k}\sum_{j=1}^{h_k} W_k(i,j).|O_k(i,j)-T_k(i,j)|}{\sum_{i=1}^{w_k}\sum_{j=1}^{h_k} W_k(i,j).V_{max}}\right] \quad (2\text{-}15)$$

Where N is the number of training images in the database. T_k, O_k, and W_k denote the k-th target image, output image, and weighted image, respectively. w_k and h_k are the width and the height of the images, and $T_k(i,j)$ indicates the brightness of the kth target image at the position (i,j). V_{max} is the brightness of the weighted image in crack region. Index R_k is the weighting function that normalizes the difference between O_k and T_k. In addition, a superimposition operation using a low-resolution image was implemented to reduce the false negative results. This algorithm is shown in Figure 5.

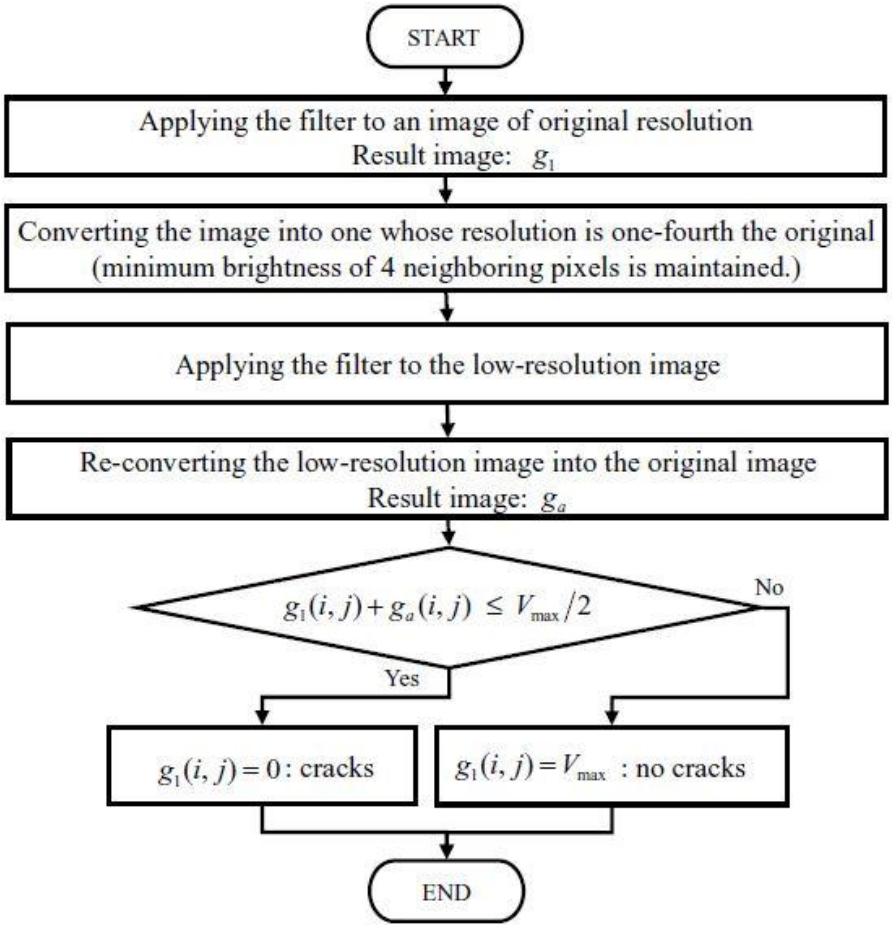


Figure 5. The process to detect cracks and remove false negative from the results (Aoki and Nagao 1999)

Another example of training-based image processing techniques can be found in work of Moon (Moon and Kim 2011). They used neural network to identify cracks using two-stage programming. The first stage used filtering, subtraction, and morphological operations to extract candidate crack pixels extraction from the background. The second stage was image classification which was done by aid of a neural network with 105 images as the training data. The image processing portion of the algorithm (e.g., the first stage) was initiated by converting the original color image into the gray-level followed by a subtraction transformation as follows:

$$I_s(x_i) = max\left\{median_{x_j \epsilon R_j}\left[I(x_j) - I(x_i)\right], 0\right\} \quad (2\text{-}16)$$

Where $I(x_i)$ is the intensity of the pixel x_i, R_i is the neighborhood of that pixel, and $I(x_j)$ is the intensity of the pixels in R_i. When the result of the subtraction is a negative number, the result

is represented with 0. In addition, the subtraction result from the above equation was used to get an enhanced image by a manually decided threshold value *T*. The thresholding operation was as follows:

$$I_{Is}(x,y) = \begin{cases} 2 \times I_s(x,y) & if\ I_s > T \\ I_s(x,y) & if\ I_s \leq T \end{cases} \quad (2\text{-}17)$$

A Gaussian Low-pass filter was then applied to smooth the image further. The next step in the image processing stage (the first stage) was construction of a binary image. The threshold value was selected using the Otsu's method or the valley emphasis method. Morphological operations including closing and labeling were applied on the image. The optimal parameters associated with the above operation are listed Figure 6.

Factors		*Levels*		
		1	2	3
Median filter size	A	30	40	60
Subtraction threshold value	B	15	20	25
Thresholding method (Binarization)	C	Global threshold	Otsu	Valley-emphasis
Gaussian filter size	D	3	5	7
Gaussian filter sig. (standard deviation of gaussian curve)	F	0.1	0.5	0.8

Figure 6. The parameters and their level (Moon and Kim 2011).

Crack image classification with artificial neural network (ANN) can distinguish cracked images from non-cracked images. The purpose of the ANN is to automatically distinguish the cracked and non-cracked images. The output value represented the classification result, which is either cracked (value 1), or non-cracked (value 2).

Oliveira proposed a method for automatic road crack detection and characterization which was based on learning from the samples paradigm (Oliveira and Correia 2013). The algorithm used in this research is shown in Figure 7.

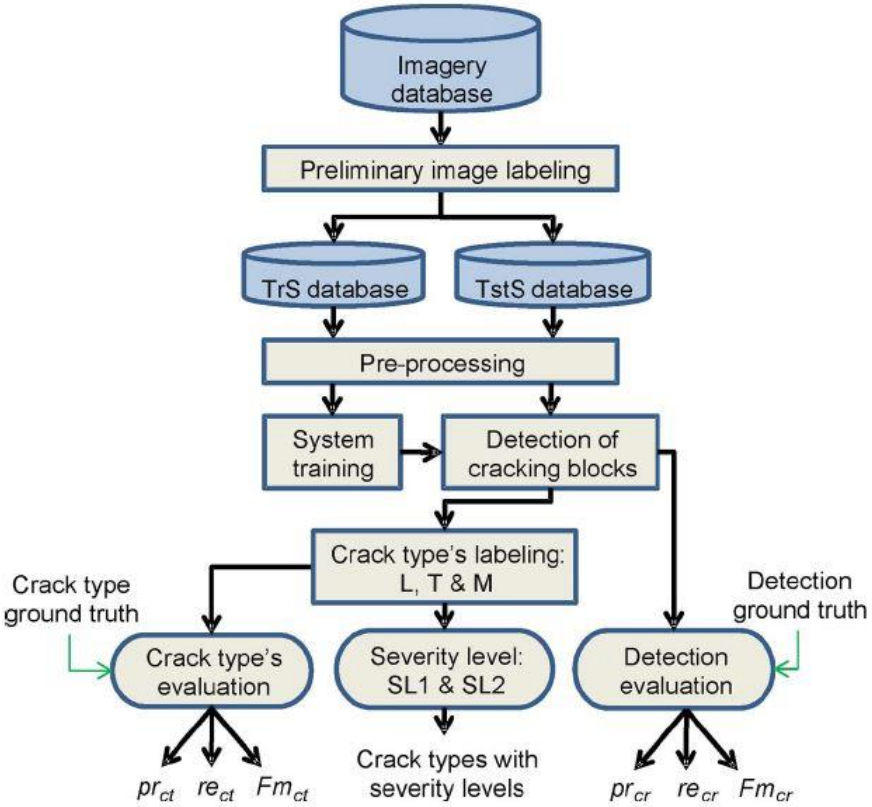


Figure 7. Crack detection algorithm (Oliveira and Correia 2013).

The input images were gray-level captured by a camera at sunlight with its optical axis perpendicular to the road surface, and its lateral edges parallel to the road axis. TrS is the training set and was chosen automatically, whereas the remaining images (TstS) were used for testing of the algorithm. The preprocessing steps are shown in Figure 8.

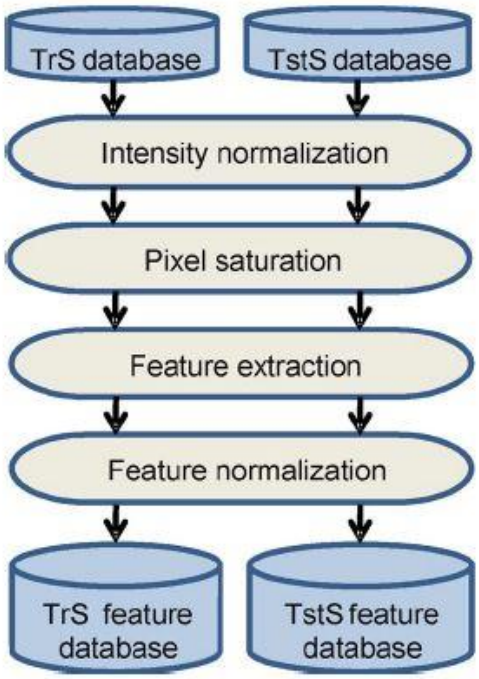


Figure 8. The preprocessing step (Oliveira and Correia 2013).

Having the features of each database, the learning process from the examples was implemented, and a boundary decision was estimated based on the training samples. Clustering techniques including: 1) hierarchical; 2) k-means; and 3) the mixture of two Gaussians were applied to divide images into two groups c_1 and c_2 with similar characteristics. For each TstS image, estimated decision boundaries were superimposed on the computed 2D feature points to determine whether the image blocks were cracked (class c_1) or non-cracked (c_2). The crack type characterization was carried out using the distances between the decision line and the feature graph. This method was able to identify three types of cracks: 1) longitudinal cracks (class c_L); 2) transverse cracks (class c_T); and miscellaneous cracks (class c_M).

Another framework was proposed to identify the cracks on the road pavement (Wang, Chen and Sun 2013). Ten different filters, including both low-pass and high-pass filters, were carried

out the reduce noise. For instance, the third step filter was a high pass filter designated as $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$. Figure 9 shows the outcome of applying each filter on the image.

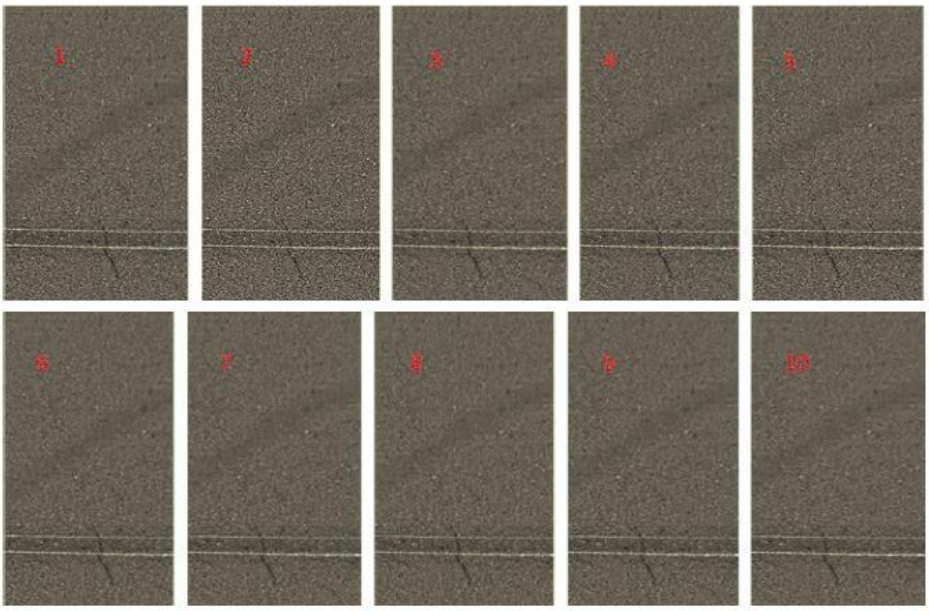


Figure 9. Image enhancement results of the pavement (Wang, Chen and Sun 2013).

A segmentation operation was proposed and applied on the enhanced image, and the region without crack was separated from calculations. The range method was used to obtain representative pixels of the cracks as shown in Figure 10.

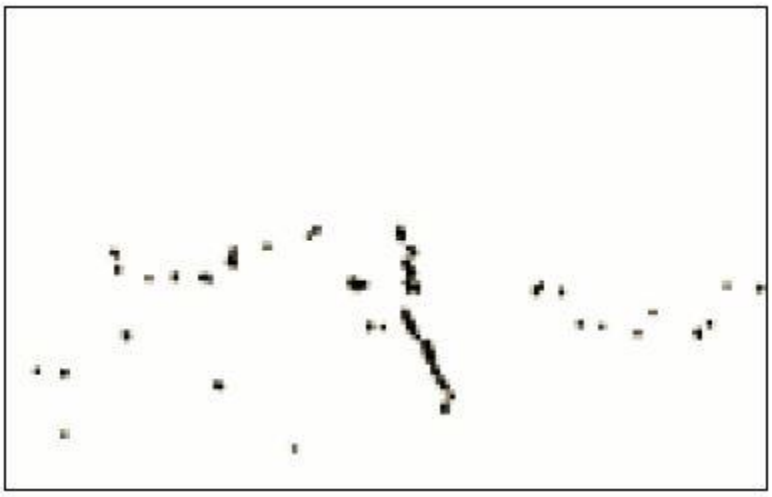


Figure 10. Illustration of representative crack pixels on the road pavement (Wang, Chen and Sun 2013).

Infrared thermography was incorporated with RGB images in the work of Matsumoto (Matsumoto, Mitani and Catbas 2013) to monitor long term bridge performance, and predict the crack pattern in the future of the structure during the service loads. Matsumoto used color images captured by HRDI and HDV inspection systems to assess the surface condition of the structure while using infrared technology to find the inner void delamination and spalling based on the difference between the damaged and non-damaged areas. To ensure the meaningful change of temperature, concrete set-up pieces were used.

Using point clouds to gain the 3D mesh model of cracks was proposed by Zheng (Zheng and Moen 2014). The images were ran through a program to generate the point cloud by Structure-from-Motion (SfM) (open source program VisualSFM was used). Considering one particular characteristic point in an image, such as edges with intensity gradient, and tracking that point in all pictures, a 3D point cloud was constructed. Then, an image was set as the reference with as many feature points as possible and all the images were compared to it. Feature points were matched based on previous steps. The next images were replaced with the reference image, and were compared with all the other pictures in the set. The process continued until all pictures were compared to each other. Trimming operations were carried out by MeshLab to remove unnecessary point cloud information. A 3D mesh of the model was created based on the trimmed model using Poisson mesh. A trimmed model along with the mesh model is shown in Figure 11. The cracks are clearly visible on the mesh model. To detect cracks, the face normal of the model was tested. Some faces were not aligned with a surface (vertical or horizontal) since they included cracks. The presence of the cracks caused the out of plane feature in the faces. This feature was measured using following formula:

$$\alpha = |90 - \beta| \quad (2\text{-}18)$$

Where α is the absolute difference between the perpendicular angle (90 degrees) and angle β, which is defined in Figure 12.

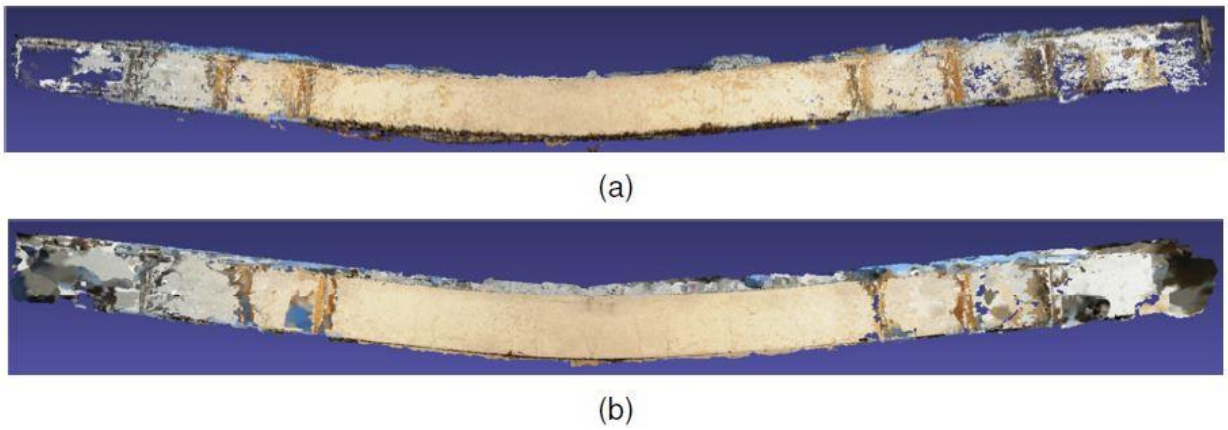


Figure 11. Slab Specimen (a) Before and (b) After Meshing (Zheng and Moen 2014)

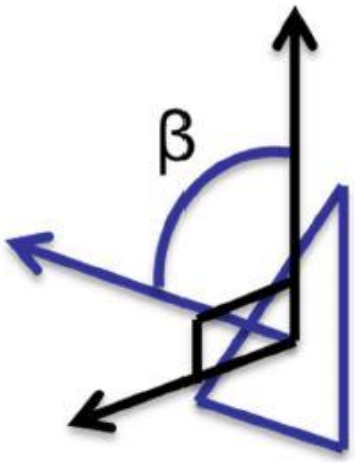


Figure 12. The face and the angle (Zheng and Moen 2014).

A threshold value of θ was defined by the user. If $\alpha<\theta$, it is un-cracked; otherwise, it is cracked.

Image segmentation and morphological operations were used by Zhang to detect and classify cracks in subway tunnel safety monitoring (Zhang, et al. 2014). Local dark regions with potential crack defects were segmented from the original gray-level image. The proposed algorithm used in their paper is illustrated in Figure 13.

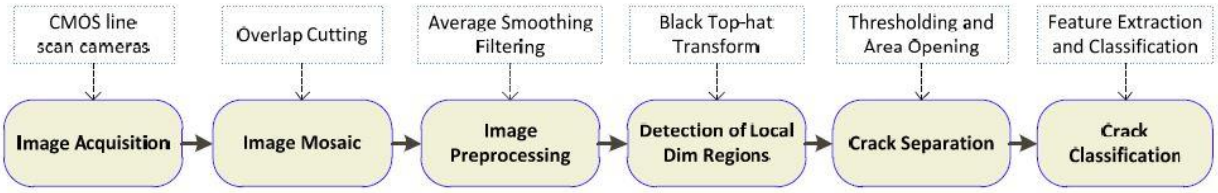


Figure 13. Flowchart of the proposed approach by Zhang (Zhang, et al. 2014).

The original images were captured by CMOS line scan cameras, and transformed to grey scale images for further processing. The mosaic image was constructed based on the collected images to remove overlaps. Then the average filter was applied followed by a top-hat transformation to identify the local dim regions, where cracks were most likely present. The black top-hat transformation was obtained using the following formula:

$$f_{BTH} = \emptyset_{Bn}(f) - f \quad (2\text{-}19)$$

Where f is the gray-scale image, and $\emptyset_{Bn}$ is the closing image set of f with respect to the structure element B_n, which was obtained by detailing n times from B_1. The first structure element B_1 was a disk, a square, or a hexagon. The regions in the transformed image consisted of cracks and other components, thus a thresholding operation was applied to distinguish cracks from background. The threshold value was calculated with Otsu's method (Otsu 1979):

$$w_0 w_1 (\mu_1 - \mu_0)^2|_{l=T_0} = \max_{0 \le k \le 255} w_0 w_1 (\mu_1 - \mu_0)^2 \quad (2\text{-}20)$$

Where w_0, w_1, μ_0, and μ_1 denote the background occurrence probability, objective occurrence probability, background mean levels, and objective mean levels, respectively. The obtained binary image included a clear presentation of the cracks compared with the cracks shown in the binary image obtained from the original gray-scale image (Figure 14.)

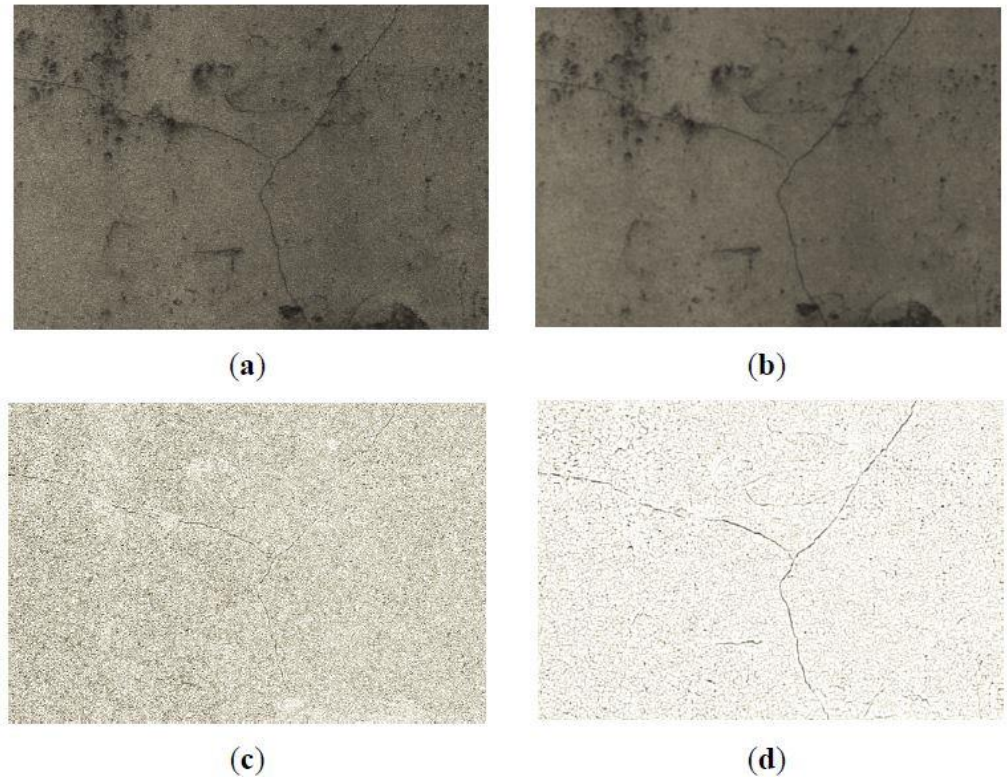


*Figure 14. Comparison of the resultant binary images: (a) original gray-scale image; (**b**) the smoothed image; (**c**) binary image obtained from (**a**); binary image obtained from (b) (Zhang, et al. 2014).*

The morphological operation "opening" was implemented to remove most of the misidentified regions. However, there were still components in the image that were not cracks. The standard deviation of the shape distance histogram technique was proposed to distinguish cracks from other objects in the binary images. This method was based on the assumption that the irregular non-crack objects had patterns with irregular shape patterns while cracks had slender patterns. In a binary image, a connected component, s_c, with the pixels location $\{(x_1, y_1), \dots (x_{Nb}, y_{Nb})\}$, the central point of the location was calculated as follows:

$$x_c = \frac{1}{N_b}\sum_{i=1}^{Nb} x_i \; ; \; y_c = \frac{1}{N_b}\sum_{i=1}^{Nb} y_i \quad (2\text{-}21)$$

The distance shape histogram was defined as:

$$p_i = \frac{N_{di}}{N_b} \quad (2\text{-}22)$$

Where N_{di} is the number of pixels with the same distance from the centroid coordinate. The standard deviation of the shape distance histogram was calculated. Assuming the distance distribution of the irrelevant objects with an irregular shape was heterogeneous. The standard deviation of irrelevant objects was larger than the standard deviation of cracks as shown in Figure 15. So irrelevant objects could be removed from the binary image. According to Figure 15, objects 1, 2, 3, 6, 8, 16, 19, 22, 23, and 24 were irrelevant objects and were removed from the crack picture.

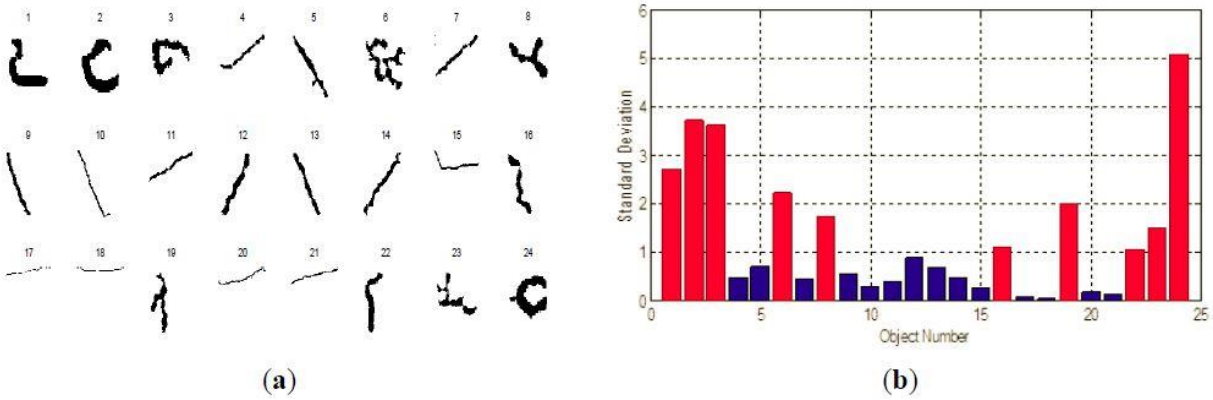


Figure 15. Comparison of the standard deviation of the shape distance histogram: (a) the candidate objects; (b) the corresponding standard deviations of distance histograms (Zhang, et al. 2014).

The cracks detection system, compatible with Unmanned Aerial Vehicles (UAV) using image-processing techniques, was developed by Kim (Kim, et al. 2015). The crack detection procedure was accomplished using image segmentation, feature extraction, and decision making. The steps of proposed algorithm are listed in Figure 16.

Step 1	Load original raw images
Step 2	Convert RGB images to gray scale images
Step 3	Cut images to only focus on the target area
Step 4	Sharpen images with Gaussian low pass filter
Step 5	Using bottom hat method to extract crack from background images
Step 6	Binary images with Otsu threshold
Step 7	Filter out those components with small area as pre-denoise
Step 8	Grouping components if the distances between them are smaller than certain distance
Step 9	If the length of the grouped component is smaller than a certain distance, this grouped component is filtered out. (The rest grouped components are detected cracks)
Step 10	Measure the grouped cracks' area
Step 11	Measure the grouped cracks' length
Step 12	Calculate the grouped cracks width in pixels
Step 13	Calculate the grouped cracks width in mm

Figure 16. Crack detection and quantification steps (Kim, et al. 2015).

In Figure 17, intermediate results for one of the pictures is shown.

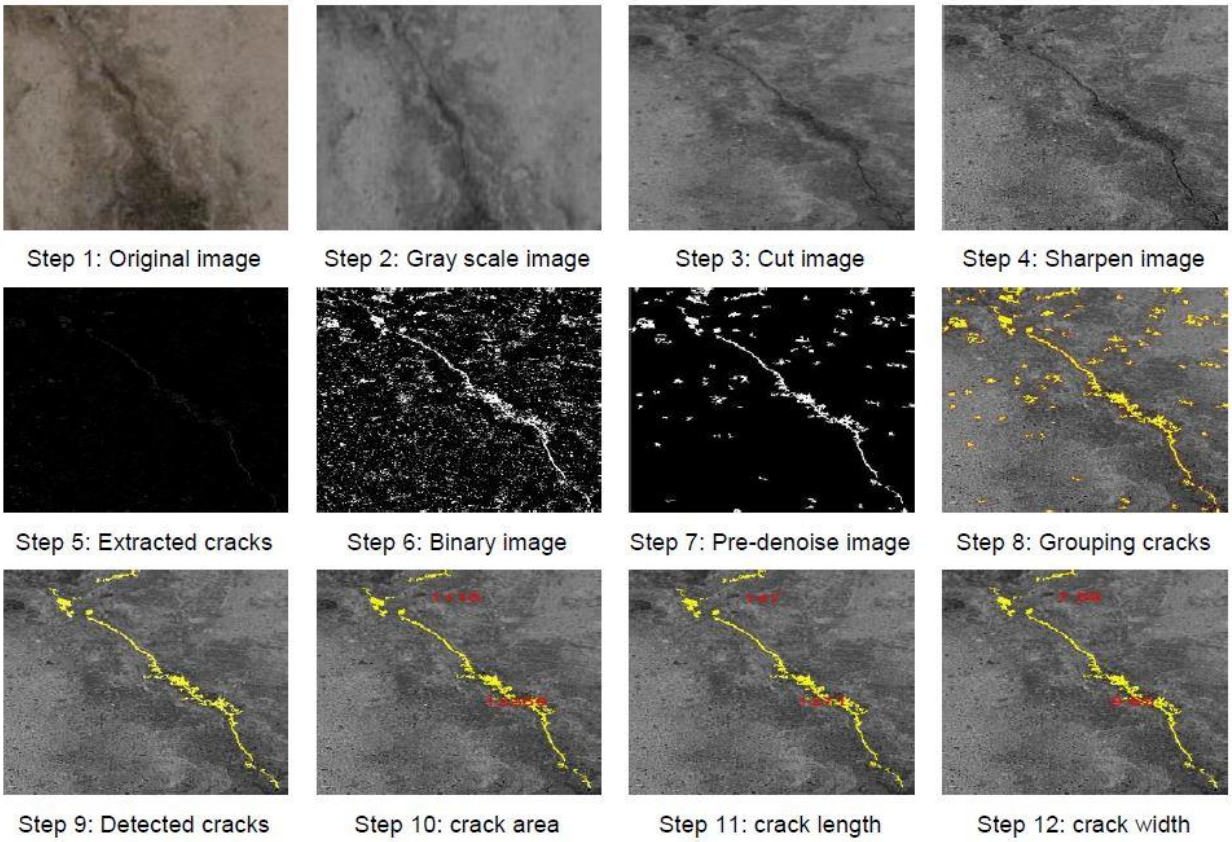


Figure 17. Intermediate results for proposed algorithm by Kim (Kim, et al. 2015).

Another use of image processing in UAVs to detect defects in structures can be found in the work of Sankarasrinivasan (Sankarasrinivasana, et al. 2015). They proposed a framework to monitor the structures using UAVs and image processing techniques to find cracks from images.

The crack detection algorithm was based on two particular characteristics of cracks in the images, i.e. thinner shape and low luminance of the crack pixels. The Hat transform and HSV thresholding were implemented to extract the crack pixels from background. HSV thresholding was based on the properties of the pixels in hue, saturation, and value color space domain. Since crack pixels have saturation and value in HSV space. To convert from RGB color space to HSV, the following transformation can be used (http://www.had2know.com/ 2016):

$$R' = R/255\,;\; G' = G/255\,;\; B' = B/255 \quad (2\text{-}23)$$

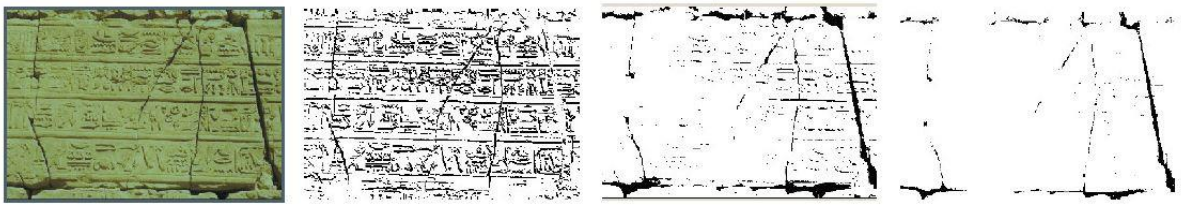
$$C_{max} = max(\acute{R}, \acute{G}, \acute{B})\,;\; C_{min} = min(\acute{R}, \acute{G}, \acute{B}) \quad (2\text{-}24)$$

$$H = \begin{cases} cos^{-1}\left[\frac{(R-0.5G-0.5B)}{\sqrt{R^2+G^2+B^2-RG-RB-GB}}\right] & G \geq B \\ 360 - cos^{-1}\left[\frac{(R-0.5G-0.5B)}{\sqrt{R^2+G^2+B^2-RG-RB-GB}}\right] & B \geq G \end{cases} \quad (2\text{-}25)$$

$$S = \begin{cases} 1 - \frac{C_{min}}{C_{max}} & C_{max} > 0 \\ 0 & C_{max} = 0 \end{cases} \quad (2\text{-}26)$$

$$V = C_{max} \quad (2\text{-}27)$$

Figure 18. shows the crack detection with resultant images.



(a) Monument Test Image (b) Hat Transform Output (c) HSV Output (d) Overall Output

Figure 18. Crack detection with the Hat transform and HSV thresholding (Sankarasrinivasana, et al. 2015)

Another technique, developed by Rikmus (Rimkus, Podviezko and Gribniak 2015), used "Agglomerative Hierarchical Clustering" to identify the crack pixels with noise reduction filters in concrete structures (Figure 19).

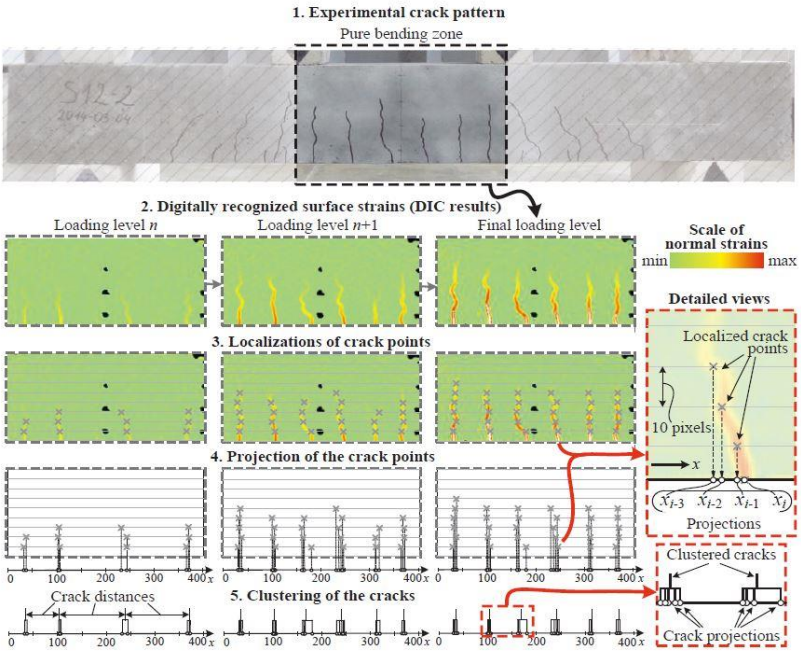


Figure 19. Crack clustering procedure by Rikmus (Rimkus, Podviezko and Gribniak 2015).

In 2015, Pereira (Pereira and Pereira 2015) applied alternative image processing algorithms for crack detection in building façades in UAV computing platforms. The algorithms used to detect edge was based on the segmentation, which was the computation of gradient of an image intensity using Sobel operators. The original RGB image was converted into the gray-scale by:

$$C(x,y) = 0.2989R(x,y) + 0.5870G(x,y) + 0.1140B(x,y) \quad (2\text{-}28)$$

Where $R(x,y)$, $G(x,y)$, and $B(x,y)$ are the pixel intensity value at (x,y) location in red, green, and blue color space, respectively. And $C(x,y)$ is the gray-level intensity at (x,y) location. The method was involved with the Sobel edge detector operation and a thresholding value comprised between 20% and 40% of the gradient magnitude. Then structure elements were defined for dilation and erosion. The magnitude of the gradient obtained from partial derivative in each direction by:

$$grad_x \cong [f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)] - [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)] \quad (2\text{-}29)$$
$$grad_y \cong [f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)] - [f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)] \quad (2\text{-}30)$$

The image with enhanced edges was converted to a binary image using the threshold value. Then the dilation operation was carried out by a 7×7 cross structuring element to connect the

edges. The morphological operation closing and erosion were performed using a 3×3 cross structuring element to fill the holes in connected components and remove of pixels out of the connected component. Figure 20 illustrates the two structuring elements used in the morphological operations.

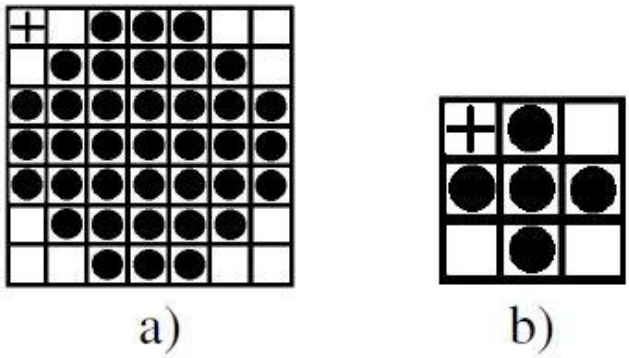


Figure 20. Structuring element. a) (7x7) for dilation of the image; b) (3x3) for closing and erosion of the image (Pereira and Pereira 2015).

Talab (Talab, et al. 2015) worked out a simple algorithm to identify cracks based on the binary images, converted from the original RGB images.

The following steps were required to detect cracks by this algorithm:

- Convert RGB image to grayscale
- Use the Sobel edge detector to find edges (including cracks)
- Use the Otsu's thresholding value to obtain the binary image
- Find the connected components in binary image with the area less than 30 and add them to the background.

The Sobel filter can be designed according to Sobel operation as follows:

$$G_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}, \; G_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} \quad (2\text{-}31)$$

Where G_x and G_y are the filters to identify vertical edges (in *x*-direction) and horizontal edges (in *y*-direction), respectively.

Using Otsu's thresholding value, the image was converted to a binary image. Then the connected objects in the images were labeled. The objects with area less than 30 were added to the background. This method was compared to Kittler method and Otsu's method to detect cracks. Kittler's method is also known as "minimum error thresholding" (Kittler and Illingworth, Minimum error thresholding 1986). They assumed that the object and pixel grey-level values are normally distributed. In Figure 21, the results of this comparison is shown.

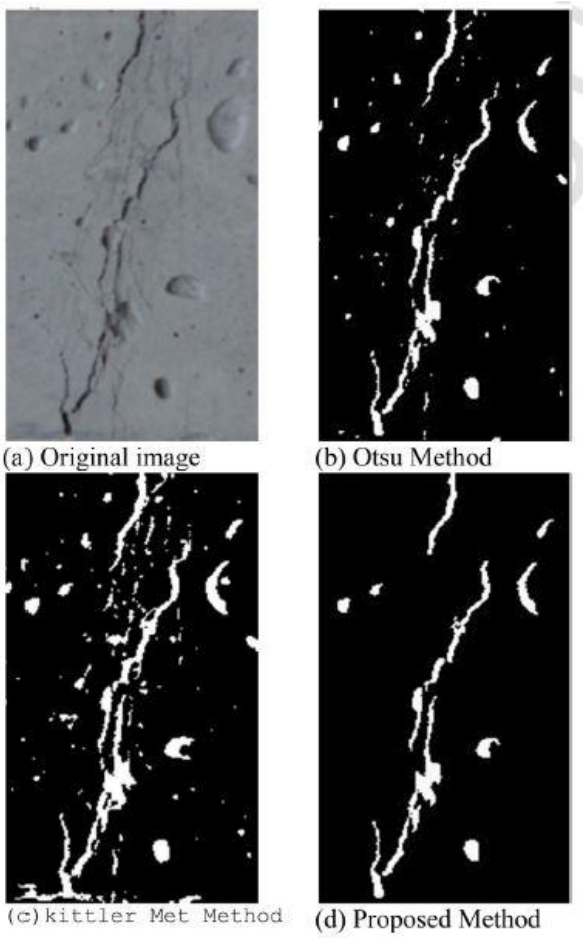


Figure 21. Experimental results for global binarization methods (Talab, et al. 2015).

3. PROPOSED METHOD

In this section, the proposed method will be explained. The proposed method provides a simple and fast automatic crack detection algorithm on concrete structures using multiple filtering operations, multiple thresholding operations, and morphological operations. In addition, several empirical strategies are used to classify cracks and non-cracks objects in the images. Examples of non-cracks objects are edges of the concrete member, boundaries, paint or dirt on surface, water marks, shadows, and background scenery lines.

3.1 STEPS OF THE PROPOSED METHOD

The input of the proposed method is any RGB image with 'jpg' extension of any size. However, the program can be easily modified for any other extensions. The proposed algorithm has been written in Matlab 2015 software. The proposed method is carried out in the spatial domain without using training and dye solution, thus it is fast and efficient. The main idea of this method has been inspired from the method proposed by Talab. However, several modifications are innovated to improve the previous method (Talab, et al. 2015).

The following steps are required to find cracks in the proposed method:

1. The original RGB is read directly from the camera
2. The RGB image is converted to the grayscale image
3. A median filter is applied on the grayscale image to smooth the surface
4. Sobel edge detectors are applied to intensify edges in the image
5. Otsu's thresholding method is applied to obtain the binary image
6. Connected components with an area less than 200 are identified and removed
7. Connected components with an orientation of 0, 90, and -90 degrees are identified and removed

8. Morphological operation ‘majority’ is applied to connect the objects and fill the holes in them
9. Objects with total pixels less than 50 are detected and removed
10. The components of the original image in the HSV color space are calculated
11. Pixels within the connected components is the HSV color space are kept as candidate crack pixels
12. A new thresholding value is defined based on the S values of the candidate crack pixels:

$$T = min(S) + std(S) \quad (3\text{-}1)$$

Where *min*(*S*) is the minimum value of all *S*, and *std*(*S*) is the standard deviation of the *S*.

13. If a candidate crack pixel’s S value is less than the threshold value calculated by Eq. (3-1), the pixel is added to the background (non-cracked)
14. If a candidate crack pixel’s S value is equal or greater than the threshold value calculated by Eq. (3-1), the pixel is preserved in the binary image (cracked)
15. The cracked pixels are superimposed on the original image with and the total number of crack pixels are computed.

3.2 DETAILED EXPLANATION OF THE PROPOSED METHOD

The images have been taken by a 12 megapixels Nikon digital camera at the Utah State University Library building and Riverwood conference hall, Logan, UT. Forty-one images have been captured with cracks, form holes, watermarks, edges and boundaries, and no-crack surfaces. The program stores all the input images in matrices for further operations.

The first step is to convert the RGB image to the grayscale image. Each component of an image in RGB color space is in the range of 0-255. The conversion of the RGB image to grayscale image is carried out using the following formula:

$$Im(x,y) = 0.2989R(x,y) + 0.5870G(x,y) + 0.1140B(x,y) \quad (3\text{-}2)$$

Where $Im(x,y)$ is the intensity value at the (x,y) coordinate of a grayscale image and $R(x,y)$, $G(x,y)$, and $B(x,y)$ are the intensity value at the (x,y) coordinate of red, green, and blue channels in an RGB image, respectively.

Concrete surface is consistent with roughness and other non-cracked abnormalities. In the proposed method, a median filter is designed to reduce the effect of the surface roughness on the crack detection algorithm. A 5 by 5 median filter is applied on the image in the second step. Median filtering is a nonlinear operation, which is generally used in image processing to reduce "salt and pepper" noise and smooth the surface. The median filter can be applied as a mask with a convolution operation on the image. Considering a 5 by 5 matrix, the center of this matrix (the median filter) is moved on all of the image intensities. In other words, the center of the filter is placed on the intensity, overlapping with 25 pixels. The median value of 25 pixels is calculated, and replaced with the original intensity. For border intensities, intensities with zero value are defined, which is called zero padding.

Crack pixels in a grayscale image show a sudden change of intensities compared to their neighborhood. Hence, finding the gradient of the image will emphasize on the cracks. Sobel edge detectors are applied on the image separately. The derivatives in each major direction, i.e. x and y, are defined as in Eq. (2-31).

They are applied on the grayscale image using the convolution operation with zero padding. Then the below formula is used to get the edge image:

$$I(x,y) = |I_x(x,y)| + |I_y(x,y)| \quad (3\text{-}3)$$

Where $I_x(x,y)$ and $I_y(x,y)$ are filtered results by applying G_x and G_y on the grayscale image, respectively.

The next step is to define a threshold using Otsu's method. The Otsu method divides the image (e.g., $I(x,y)$ obtained in the step 4) into two groups: Target pixels and background pixels. This method chooses the thresholding value based on the minimization of intra-class variance (the variance within the class). Minimizing the intra-class variance is the same as maximizing the inter-class variance (Otsu 1979). The intra-class variance is defined as:

$$\sigma_w^2(T) = \omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T) \quad (3\text{-}4)$$

Where ω_0 and ω_1 are the probabilities of the two classes separated by the threshold T, and σ^2_0 and σ^2_1 are variances for these two classes. The probabilities are computed from L histogram:

$$\omega_0(T) = \sum_{i=0}^{T-1} p(i)\,;\ \omega_1(T) = \sum_{i=T}^{L-1} p(i) \quad (3\text{-}5)$$

The class mean value can be calculated as:

$$\mu_0(T) = \sum_{i=0}^{T-1} ip(i)/\omega_0\,;\ \mu_1(T) = \sum_{i=T}^{L-1} ip(i)/\omega_1 \quad (3\text{-}6)$$

And the individual class variances can be obtained using:

$$\sigma_0^2(T) = \sum_{i=0}^{T-1}[i-\mu_0(T)]^2 \frac{p(i)}{\omega_0(T)} \quad (3\text{-}7)$$

$$\sigma_1^2(T) = \sum_{i=T}^{L-1}[i-\mu_1(T)]^2 \frac{p(i)}{\omega_1(T)} \quad (3\text{-}8)$$

A number between 0 and 255 should be assigned to T and the corresponding intra-class variances are computed using the above formulas. The value of T corresponding with the maximum intra-class variance is the thresholding value called Otsu's thresholding value. This proposed method uses 20% of the Otsu threshold to construct a binary image using the following formula:

$$B(x,y) = \begin{cases} 0 & if\ I(x,y) < 0.2T \\ 1 & if\ I(x,y) \geq 0.2T \end{cases} \quad (3\text{-}9)$$

Where $B(x,y)$ is the binary image pixel at the location of (x,y) and $I(x,y)$ is the image intensity after applying Sobel edge detectors at the (x,y) location.

The image achieved to this point usually has small objects, which appear due to surface unevenness. To remove these objects, an area threshold is applied on the binary image. The number of pixels for each connected component (object) is calculated and the area of each object is estimated. The objects with an area less than 200 pixels will be added to the background pixels. The image attained in this step will be referred to as Post-Processed images 1 later in this report.

One of the major problems in automatic crack detection is classification between cracks and other linear objects on concrete, such as member edges, background lines, scenery lines, paint or oil stains, and so on. In this report, multiple techniques are proposed and implemented to separate cracks from irrelevant objects.

Cracks do not appear as a straight line on a concrete surface. On the other hand, edges of a concrete member, such as a column or a beam, are usually straight lines. In this step, the orientation of the connected components (objects) in the Post-Processed image 1 is calculated. For each object an equivalent ellipse with the equal second moment of the object is defined. The angle between the x-axis of the major axis of the ellipse (longer diameter), varying in the range of -90 to 90 degrees, is reported as the orientation of each object. Thresholding operation will be conducted based on the orientation. The objects with the orientation 0, -90, and 90 degrees are identified, and added to the background. An offset of ±1 degree is considered since it is improbable to have absolutely vertical and horizontal objects. The image after applying this threshold is referred to as Post-Processed image 2 later in this report.

Morphological operation is implemented to make the objects continuous to the next step. The objects (cracks) could be discretized after earlier procedures. Notice while carrying morphological operation out, non-cracked irrelevant pixels might get connected to the cracked pixels, which will reduce the ability of the program to detect cracks significantly. To address this issue, we apply morphological operation "majority" on Post-Processed Image 2. 3×3 structure element is defined to go through the image. If five or more pixels in the 3×3 structure element have the value of 1 (target pixel), the pixel in the center of the mask is set to 1. Otherwise, the program sets the value to 0 (background pixel). This majority operation, is repeated until the current resultant image is the same as the previous resultant image. The image made in this step is called Post-Processed Image 3.

Post-Processed Image 3 may have small objects yielded from the morphological operation of the previous step. Objects with the number of pixels less than 50 are detected and removed from the image. This threshold value is determined manually according to the database used in this report. The image obtained in this step is referred to as Post-Processed Image 4.

To this point, all of the crack detection techniques have been carried out on the grayscale image. The next step is an innovative method proposed based on the properties of the image in the color space. The original input image is an RGB image, where all pixel values are defined as a percentage of three basic colors including red, green, and blue (Figure 22). However, this is not the only color space presenting a color image. HSV color space is defined with hue, saturation, and the value of a color (Figure 23.). Hue of color (H) is the pure color resemblance, which is described by a number between 0 and 1, specifying the position of the pure color on the color wheel. For instance, the Hue value 0 is for red, 1/6 is for yellow, and 1/3 is for green. The saturation value (S) of a color stands for how white the color is. For instance, a pure red has the saturation of 1, while the white has the saturation of 0. The value of a color (V) represents the lightness of the

color for instance a pure black has V value of 0. The cracks showed larger values in the S component of the color image. This characteristic is used to categorize cracks from irrelevant objects that have not been cleared from the image yet.

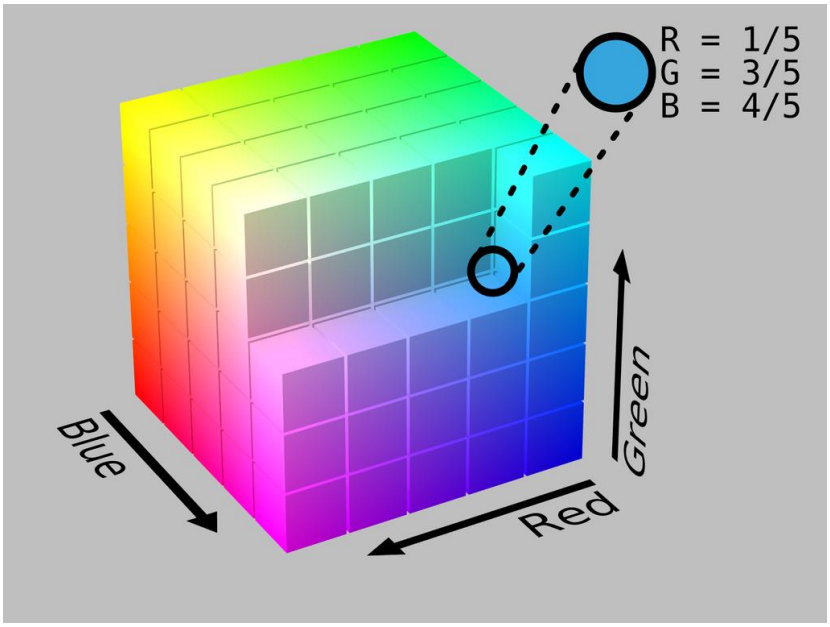


Figure 22. The cube representing the RGB color space (Wikipedia, The Free Encyclopedia 2016)

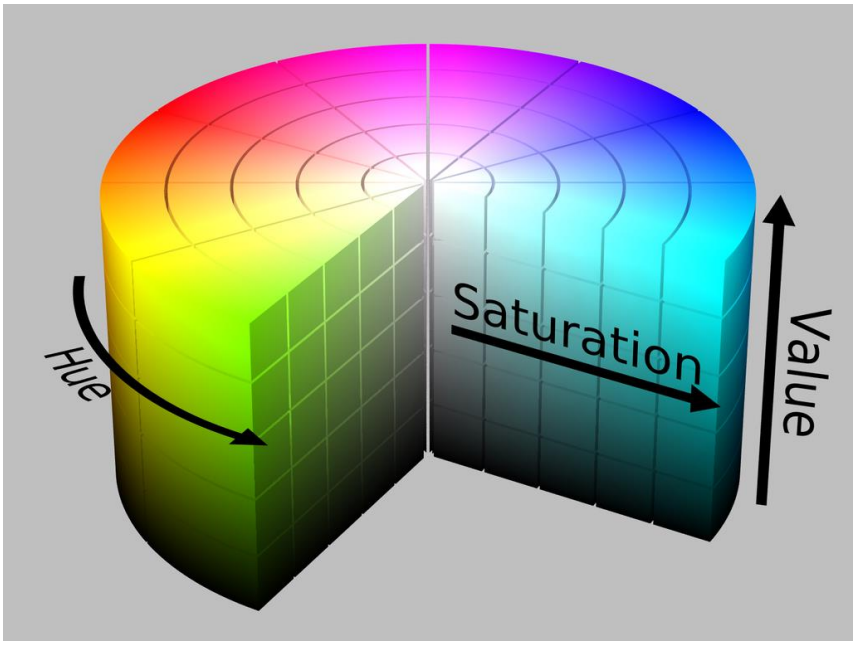


Figure 23. The cylinder representing the HSV color space (Wikipedia, The Free Encyclopedia 2016)

The relationship between the HSV color space and the RGB color space is given by Eq. (2-23) through Eq. (2-27). The original RGB image is converted to the HSV color space using

these formulas. Then, the S values of Post-Processed Image 4 are extracted. A thresholding value is calculated with Eq. (3-1). If the S value corresponding to a pixel in Post-Processed Image 4 is less than the threshold, the pixel is added to the background. Otherwise, the pixel is kept intact. The image obtained in this step is called Post-Processed Image 5.

Figure 24 shows the crack detection procedure proposed in this report with intermediate results displayed in the raster scan order. The original image, crack detection results and the reconstructed image with cracks and statistics are shown in Figure 25 for the same input image. Each sub-image in Figure 24 is obtained as the following:

1. Grayscale Image is the image attained in step 2 using Eq. (3-2).
2. Median Filtered is the image attained in step 3 after applying a 5 by 5 median filter.
3. Edge Image is the image attained in step 4 by applying filtering operation with the masks defined in Eq. (3-3) on the Median Filtered image and combining the two filtered results using Eq. (3-4).
4. Threshold Image is the image attained in step 5 using Eq. (3-10)
5. Post-Processed Image 1 is the result of the step 6.
6. Post-Processed Image 2 is the result of the step 7.
7. Post-Processed Image 3 is the result of the step 8.
8. Post-Processed Image 4 is the result of the step 9.
9. Post-Processed Image 5 is the final result of the steps 10 through 12

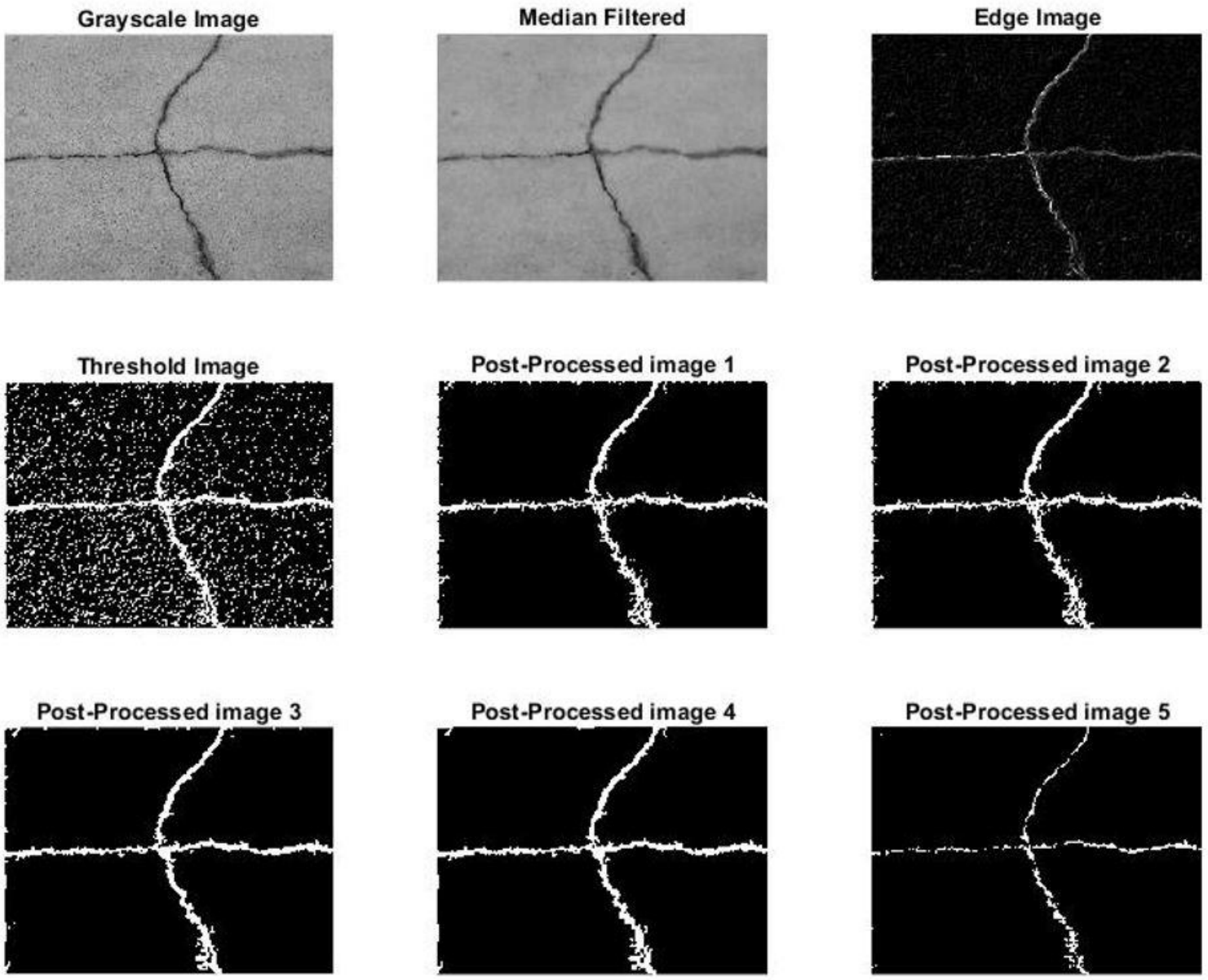


Figure 24. Crack detection results using the proposed algorithm

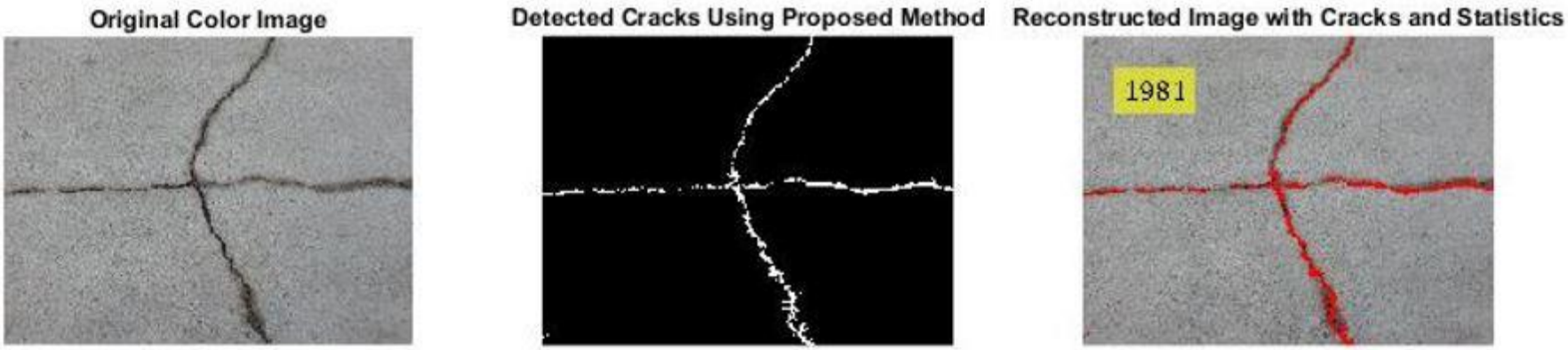


Figure 25. The crack detection results and the reconstructed color image with cracks and statistic, using the proposed method

4. EXPERIMENTAL RESULTS

The experimental results of the proposed method will be discussed in this section. The advantages and shortcomings associated with this method will be discussed and compared with the method proposed by Talab (Talab, et al. 2015). The efficiency of any automated unmanned inspection method is assessed in the following terms:

1. True Positive (TP): means successfully identified and reported a true particular defect that exists on the structure.
2. True Negative (TN): means successfully identified that a particular defect does not exist on the structure.
3. False Positive (FP): means identified or reported a particular defect that does not exist on the structure.
4. False Negative (FN): means unsuccessfully identified or reported a particular defect that exists on the structure.

These factors are usually reported as the ratio of TP, TN, FP, and FN and the number of images in the database, respectively.

4.1 EXPERIMENTAL RESULTS OF THE PROPOSED METHOD

A higher rate of TP indicates better efficiency of the method. Figure 26 shows a TP report for surface crack detection using the proposed method on another sample image. As seen in the original image, there is a horizontal crack and stain on the surface. The Median Filtered image has less surface roughness, which aids the program to be more efficient to detect cracks. Edge Image shows all of the image edges. Using thresholding value, Threshold Image is constructed. In Post-Processed Image 1, all small objects are removed. Since there are no vertical or horizontal objects

in the original image, Post-Processed Image 2 is the same as Post-Processed Image 1. The effect of the morphological operation can clearly be observed in Post-Processed Image 3, connecting the remaining target pixels together. In Post-Processed Image 4 another filter is applied to get rid of small objects in the target pixels. Finally, in Post-Processed Image 5, target pixels have been classified as cracked and non-cracked using the properties of cracked pixels in the HSV color space. The non-cracked pixels have been added to the background. Figure 27 shows the reconstructed image with superimposed cracked pixels and statistics.

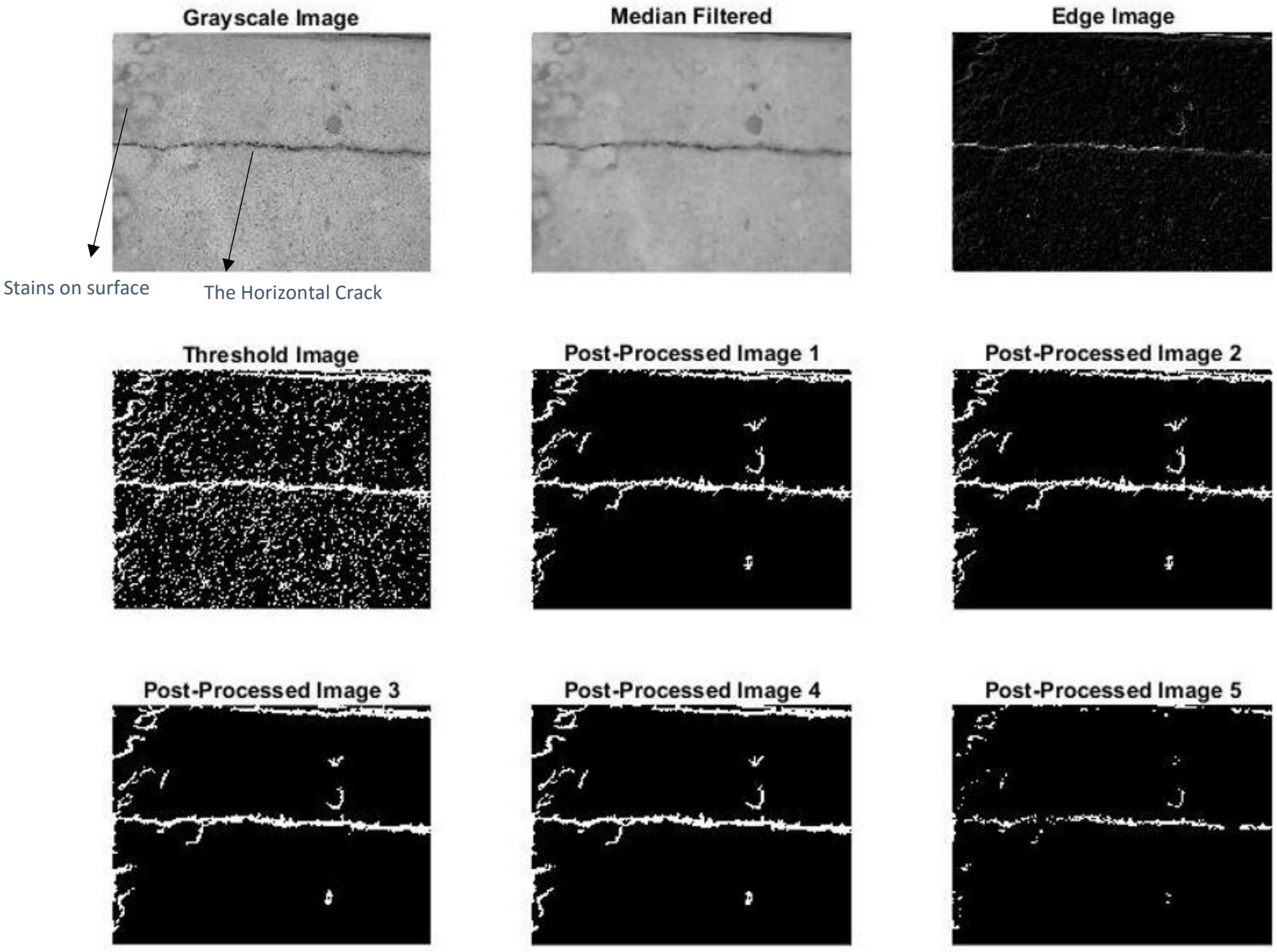


Figure 26. An example of TP using the proposed method

Figure 28 shows another sample image with no cracks on the surface. The program reported 0 cracked pixels, which is in accordance with the ground truth. This is a case of a TN. However, considering the intermediate results in Figure 29, the left boundary of the image was detected as cracked pixels until Post-Processed Image 2. Using the morphological operation which yields Post-Processed Image 3, removal of small objects which yields Post-Processed Image 4, and HSV properties of the original image and target pixels which yields Post-Processed Image 5, the boundary has been completely removed. Thus, the program did not detect any cracks on a crack-free surface.

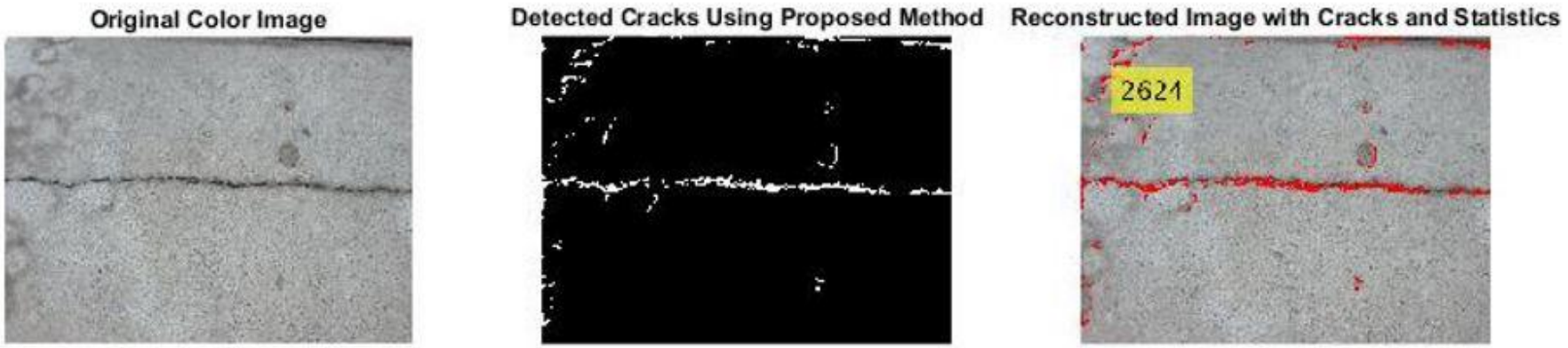


Figure 27. The crack detection result and the reconstructed color image with cracks and statistics for a TP case using the proposed method

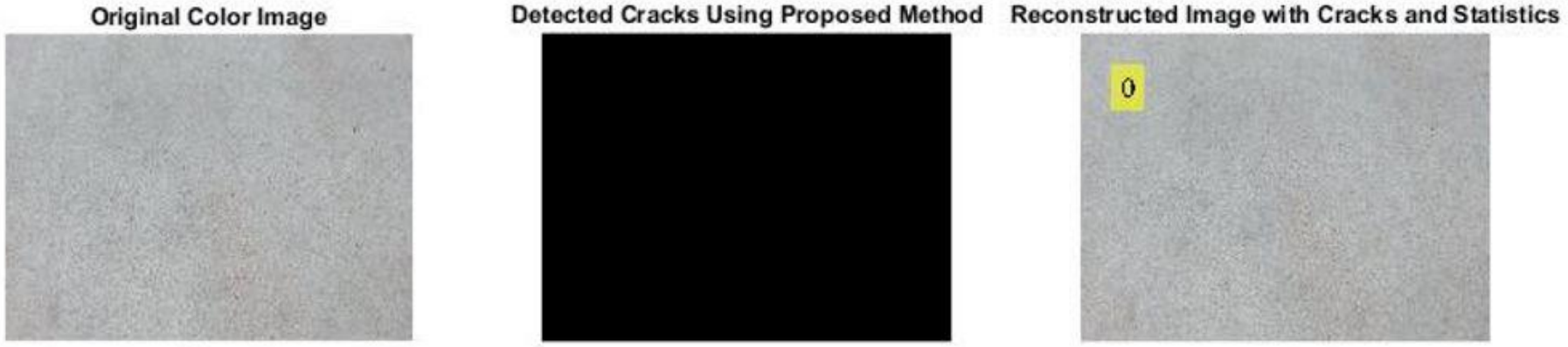


Figure 28. The crack detection report and the reconstructed color image with cracks and statistics of a TN case using the proposed method

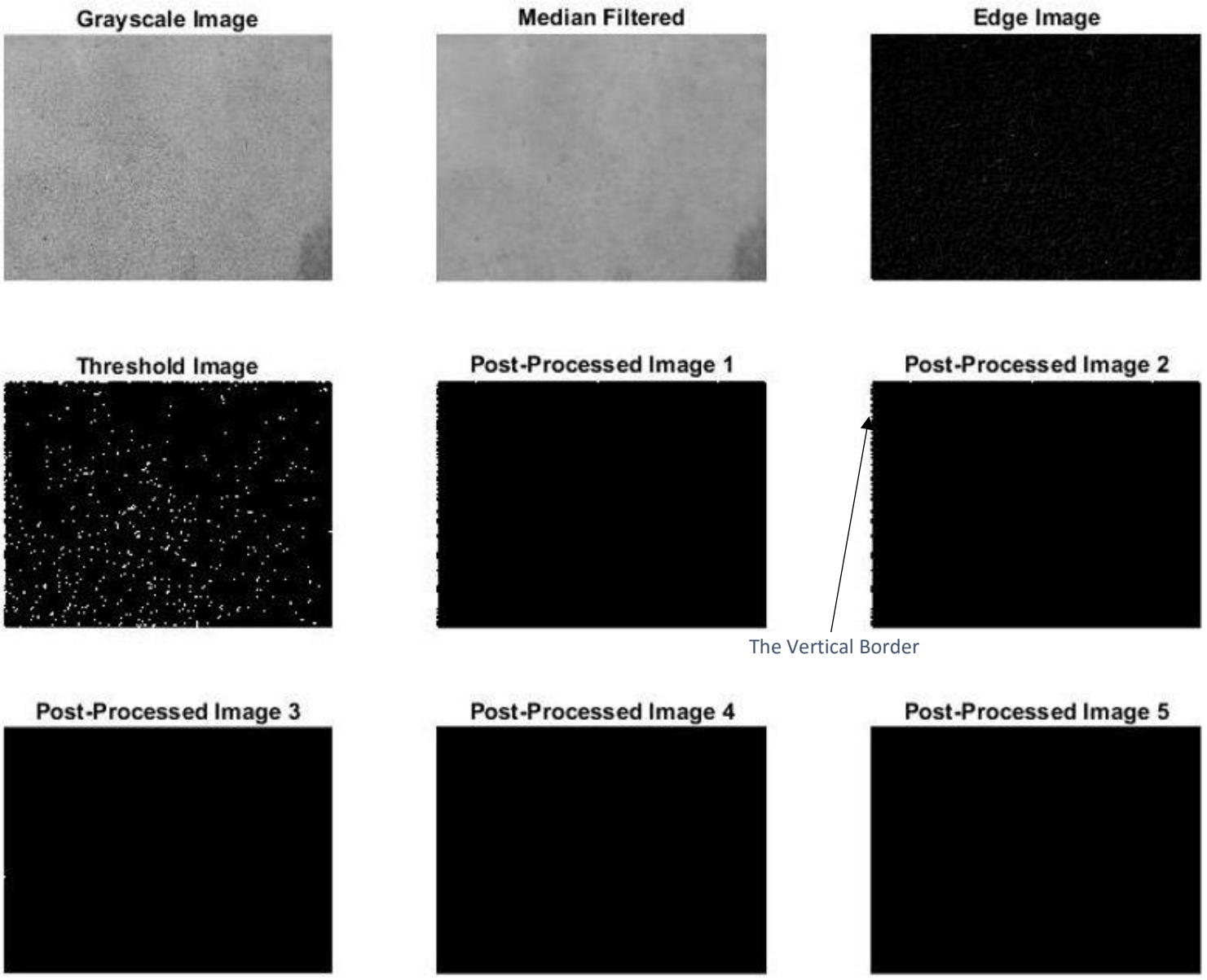


Figure 29. An example of TN using the proposed method

FP and FN reports should be minimized in order to have more reliable automatic crack detection methods. Although different techniques have been used in this report to minimize the amount of false reports, there are cases where the proposed method failed. An example of FP detection is shown in Figure 30. The concrete column investigated in the original image has a visible edge, a vertical form line, and form holes. The techniques applied for this image were not completely effective (Figure 30). HSV thresholding technique in step 10 (Post-Processed Image 5) removed some of the non-cracked pixels from the image. But 1502 cracked pixels were reported in the Reconstructed Image with Cracks and Statistics (Figure 31). Some of the cracked pixels are associated with the form holes, which are desirable since they can provide a route for water and corrosive materials to penetrate the concrete. Other crack pixels picked up by the program should have been classified as non-cracked, and added to the background.

FN reports only appeared in three cases for all images in the dataset. The program was unable to detect a horizontal crack shown in Figure 32. Although the crack was big, the pixel count of the final is only 32. Furthermore, it is not associated with the main crack. For a more precise scrutiny, the intermediate results are shown in Figure 33. It seems the crack has been identified in the Edge Image. After applying step 7 (removal of the vertical and horizontal objects), almost all the cracked pixels were removed. The crack itself seems to be very continuous in Post-Processed Image 1. However, the orientation of the crack obtained from the equivalent ellipse was near 0 degree. Since almost all of the crack pixels are connected, and recognized as a single object by the program, the operation in step 7 removed all of the pixels associated with that object (i.e. the big crack). More images processed with this proposed method can be found in Appendix A.

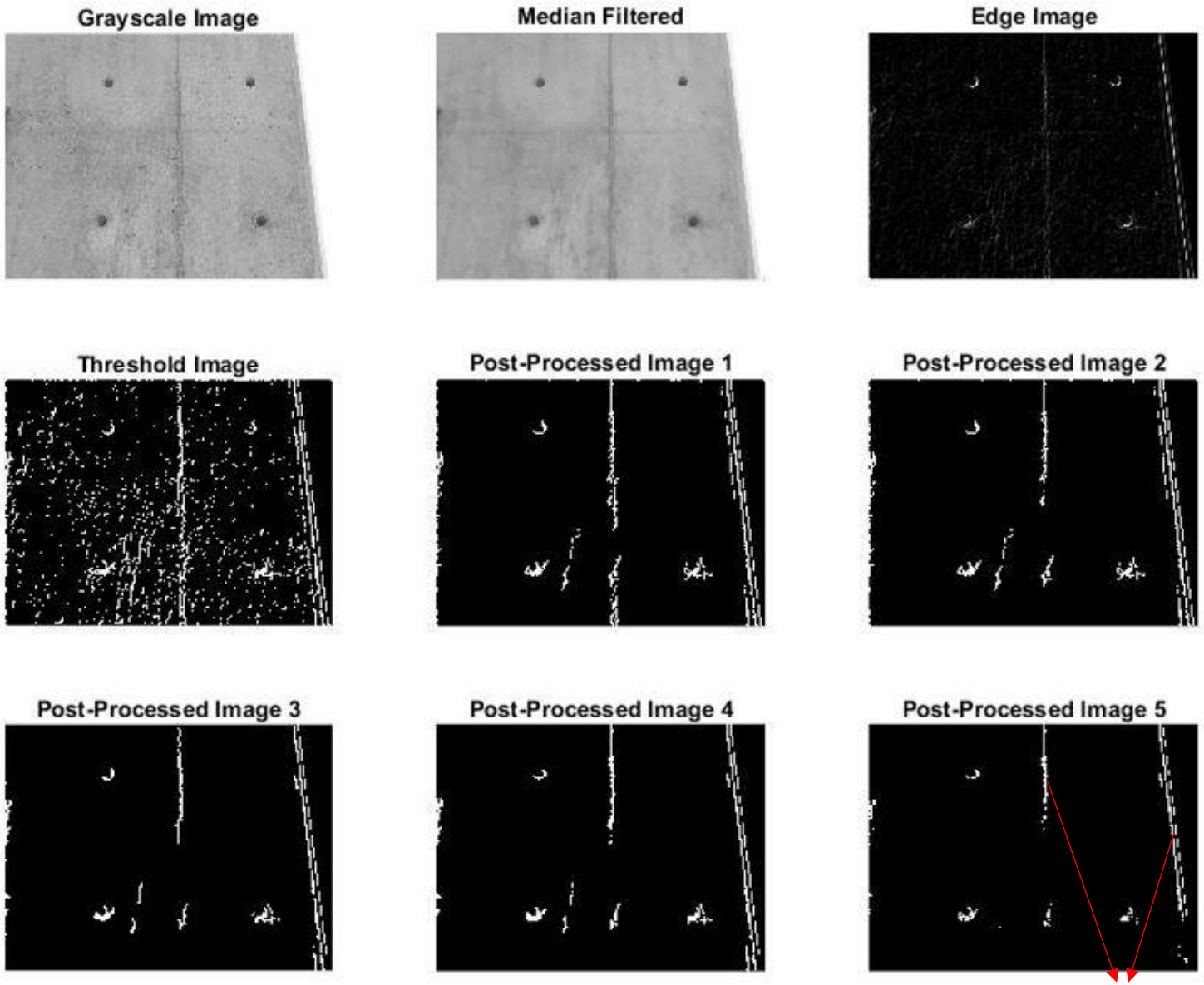


Figure 30. An example of FP using the proposed method

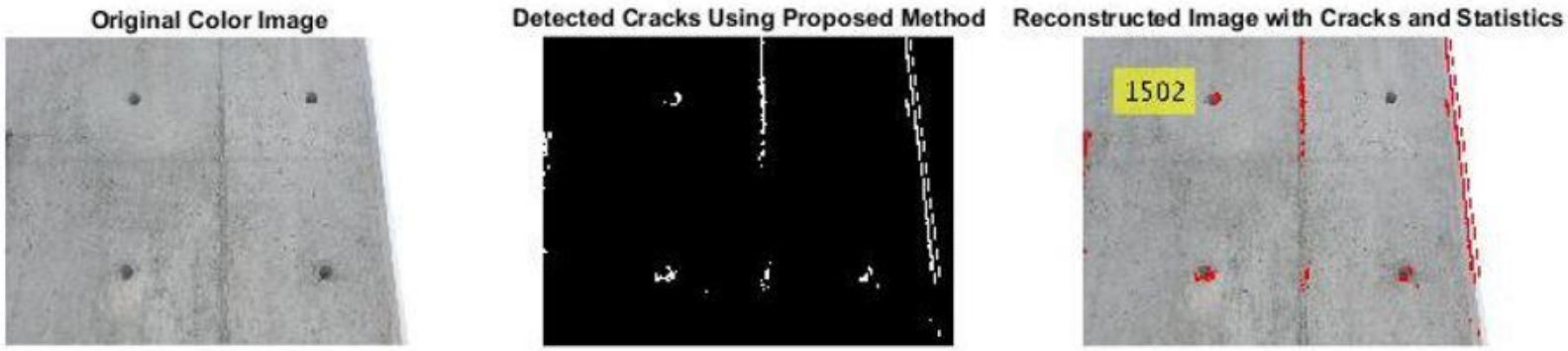


Figure 31. The crack detection result and the reconstructed color image with cracks and statistics of a FP case using the proposed method

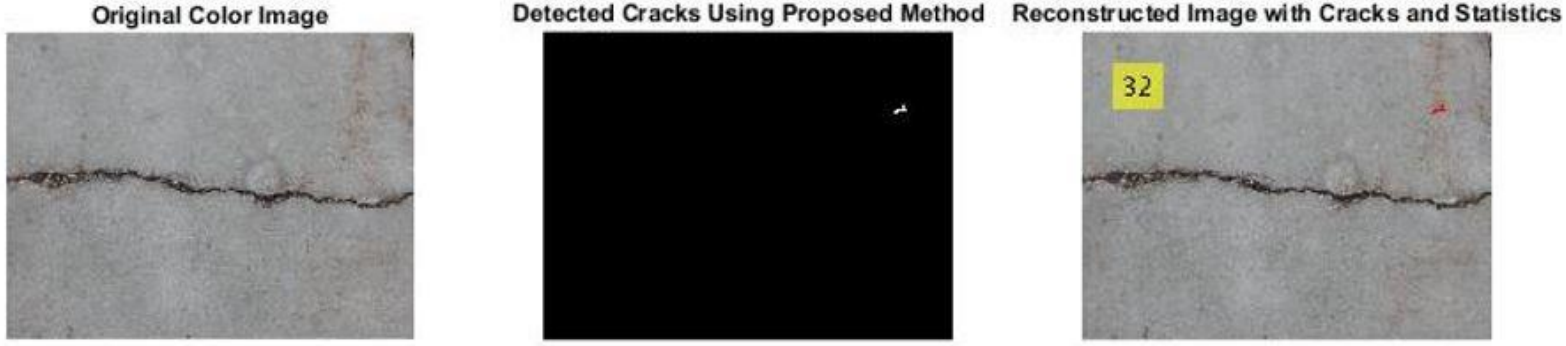


Figure 32. The crack detection result and the reconstructed color image with cracks and statistics of a FN case using the proposed method

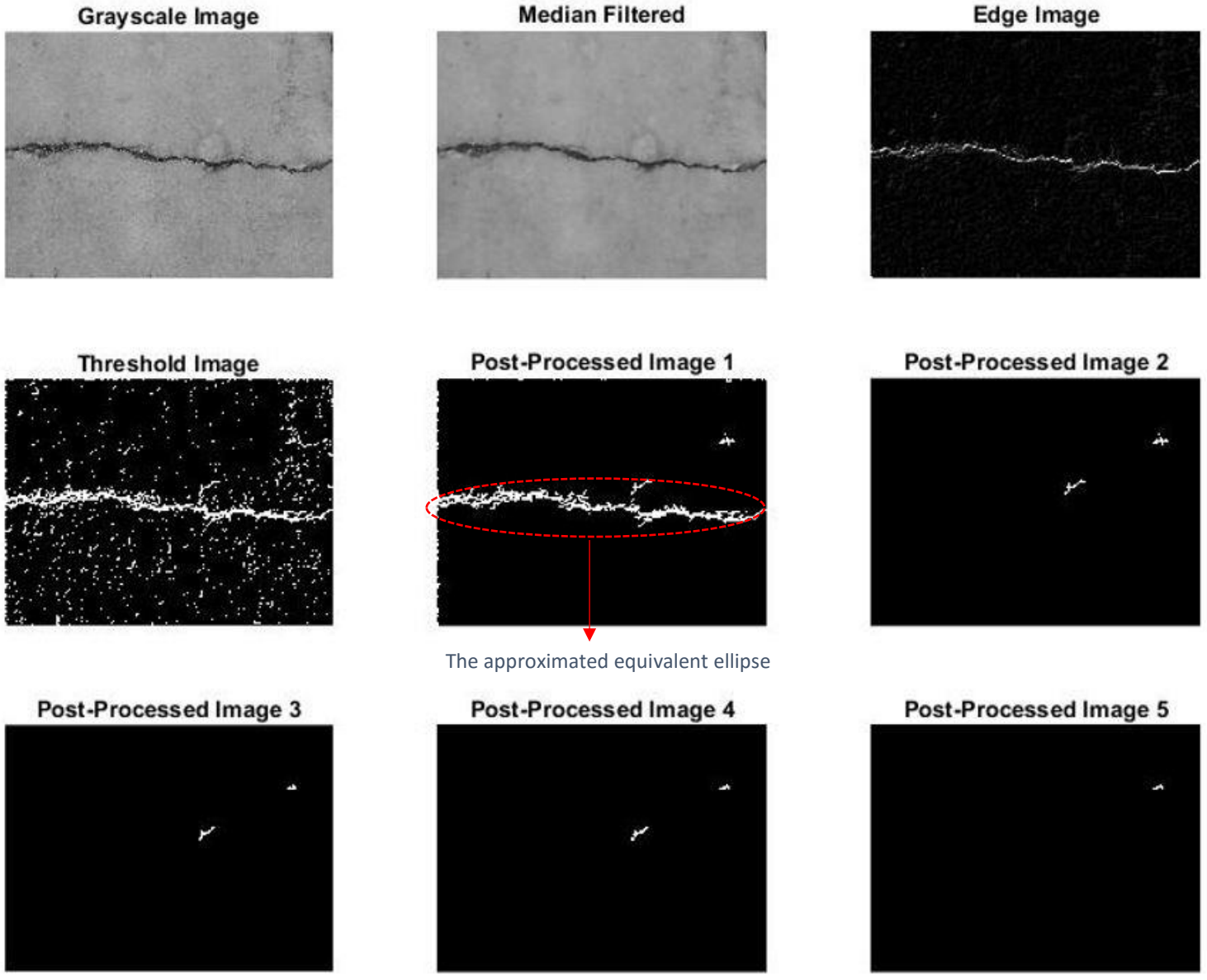


Figure 33. An example of FN using the proposed method

4.2 THE ADVANTAGES OF THE PROPOSED METHOD

The proposed method was built on the method proposed by Talab (Talab, et al. 2015). However, the proposed method has clear advantages over Talab's work. Talab's methodology has been written in Matlab, and tested with the same database experimental with the proposed method. The steps to detect cracks using Talab's method are as follows:

1. Convert RGB image to grayscale using Eq. (3-2).
2. Use the Sobel edge detector to find the edges with masks defined by Eq. (3-3). Get the value of the edge image using Eq. (3-4).
3. Use the Otsu's thresholding value to obtain the binary image by applying Eq. (3-5) through Eq. (3-9). Let T be the threshold value obtained from Otsu's method. The thresholding operation is defined as follows:

$$B(x,y) = \begin{cases} 0 & if\ I(x,y) < T \\ 1 & if\ I(x,y) \geq T \end{cases} \quad (4\text{-}1)$$

   Where, $B(x,y)$ is the binary image value at the (x,y) coordinate and $I(x,y)$ is the intensity value at the (x,y) coordinate for the edge image.

4. Find the connected components in the binary image with the area less than 30 pixels and add them to the background, i.e. removing small objects from the image.

For convenience, the images obtained in steps 1 through 4 are called Grayscale Image, Edge Image (Talab), Threshold Image (Talab), and Post-Processed Image (Talab), respectively. The intermediate results using Talab's method for a sample image shown in Figure 24 are presented in Figure 34. In addition, the reconstructed color image with cracks and statistics is shown in Figure 35. As it can be seen, Talab's method could not detect all the cracks in the original

image. According to Talab's method, the left border pixels of the image is reported as cracks while there are no cracks occurred in that area in the original image.

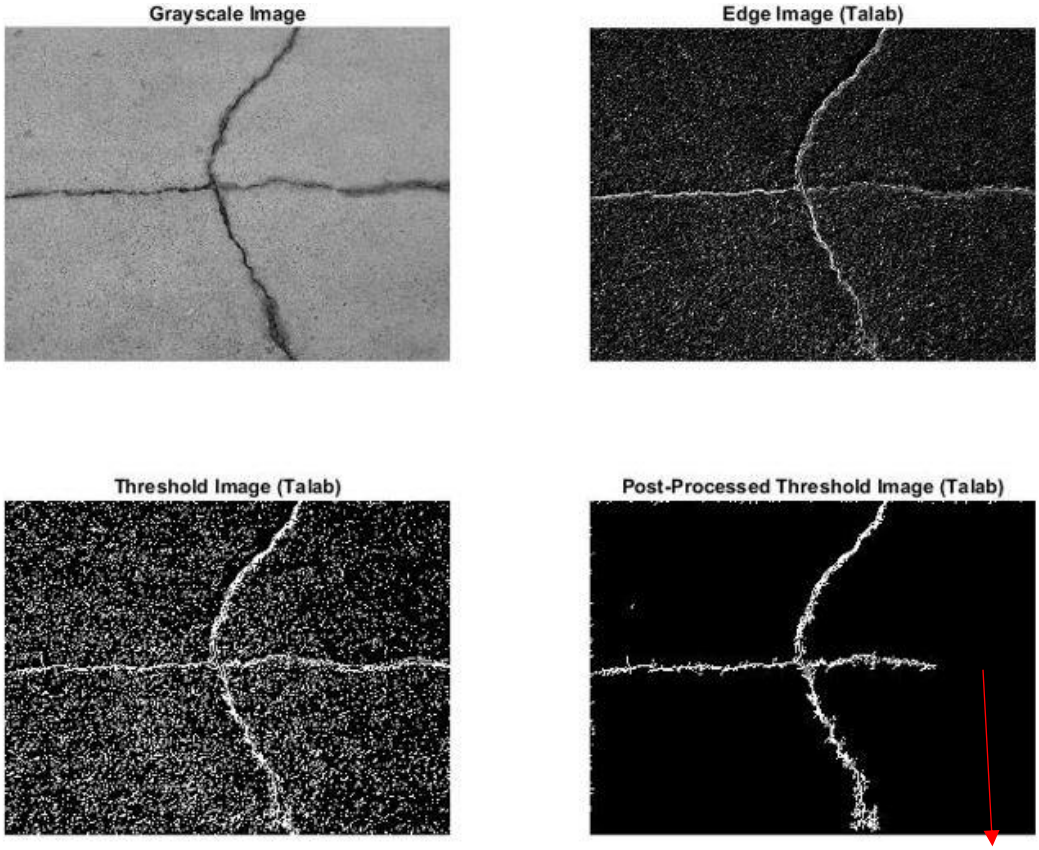


Figure 34. . Crack detection results using Talab's algorithm

Figure 36. shows a comparison between the proposed method and Talab's method. The crack pixels detected by the proposed method and Talab's method are displayed in Red and Blue, respectively. As it can be seen, the total number of cracked pixels identified by the proposed method is less than the total number of cracked pixels identified by the Talab's method. However, the proposed method was able to completely detect both vertical and horizontal cracks without involving lots of noisy pixels.

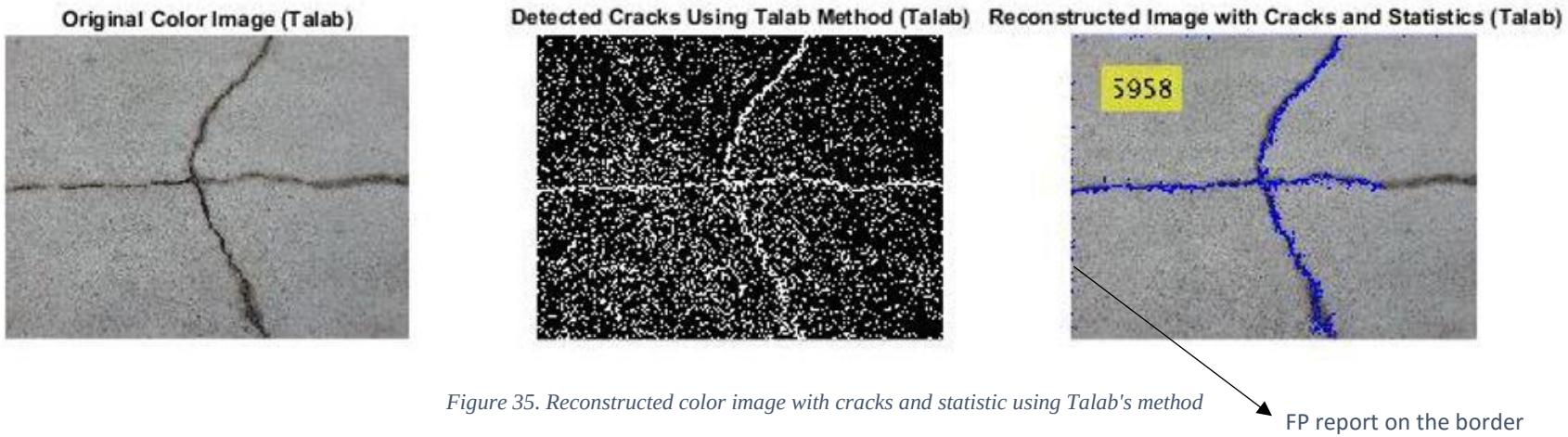


Figure 35. Reconstructed color image with cracks and statistic using Talab's method

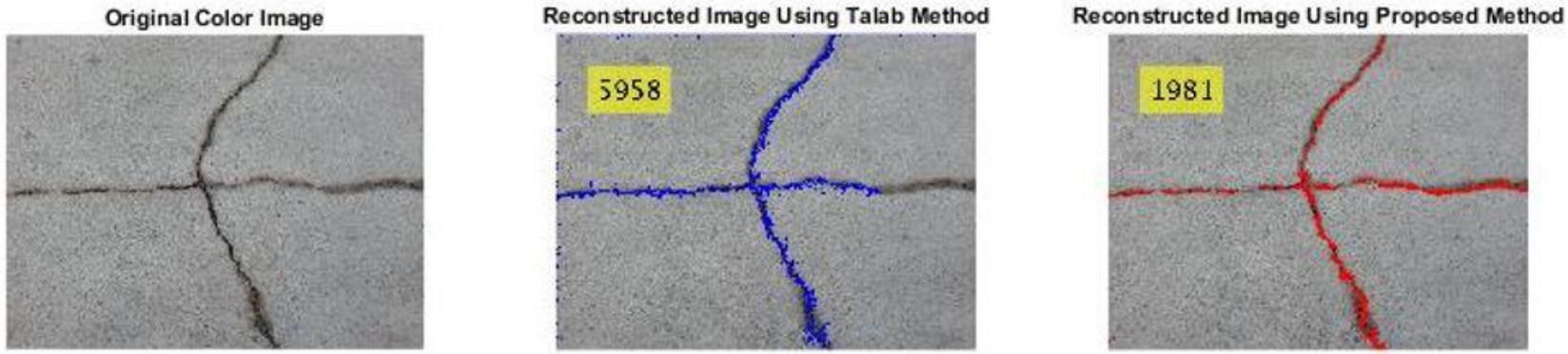


Figure 36. Comparison results using Talab's method and obtained by the proposed method

As mentioned before, there are four indicators used to assess any automated unmanned methods by researchers and engineers: TP, TN, FP, and FN. Methods with higher rates of TN and TF and lower rates of FP and FP are more desirable and effective in the field.

A comparison can be made based on the input image processed in Figure 27 for TP report. The reconstructed color images using Talab's methodology and the proposed method are shown in Figure 37. The proposed method more accurately identifies the horizontal crack in the input image. Even though the crack statistics for both methods are almost equal, 2695 pixels for Talab's method and 2621 pixels for the proposed method, Talab's method only detected roughly one fourth of the crack's length. It means most of the target pixels in the Talab's method are not associated the crack. Figure 37 indicates that the proposed method detects cracks more effectively than Talab's method. In other words, the proposed method provides a higher rate of TP. Twenty-five images with surface defects (cracks and form holes) have been used in this study. Talab's method completely or partially identified 13 images to be defected (an example of partial detection can be seen in Figure 36). The proposed method showed more promising TP reports with 22 completely or partially defect detection. More comparison results can be found in Appendix B. Considering the definition of FN, the remaining images which had not been picked up from defected database are the FN rate. Thus, Talab's methodology reported 12 images while the proposed method reported only 3 images as FN. Figure 38 shows the effectiveness comparison of both methods to detect a crack. The Talab's method reported False Positive while the proposed method successfully detected the crack. The target pixels produced by the Talab's method were not associated with defects.

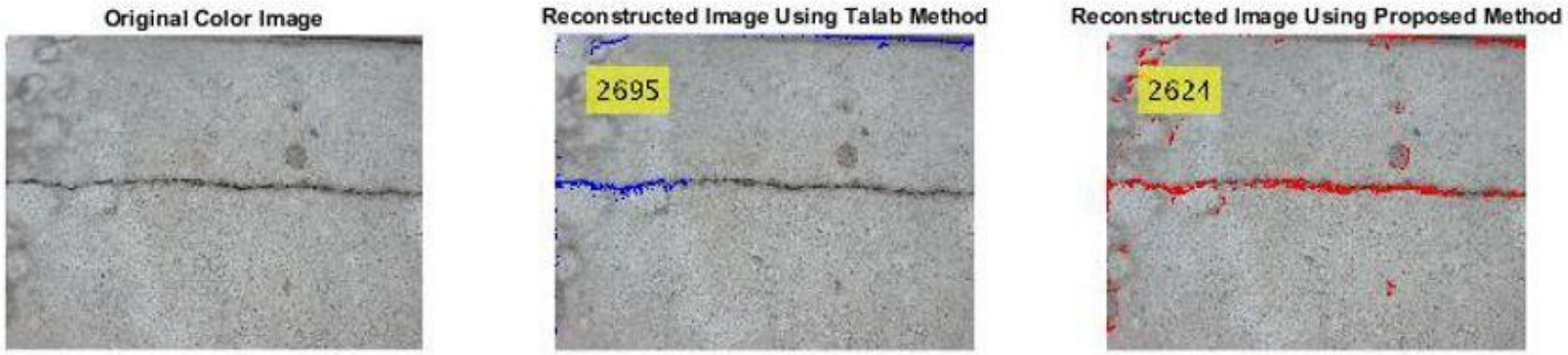


Figure 37. The comparison between two methods reporting TP for the same input image

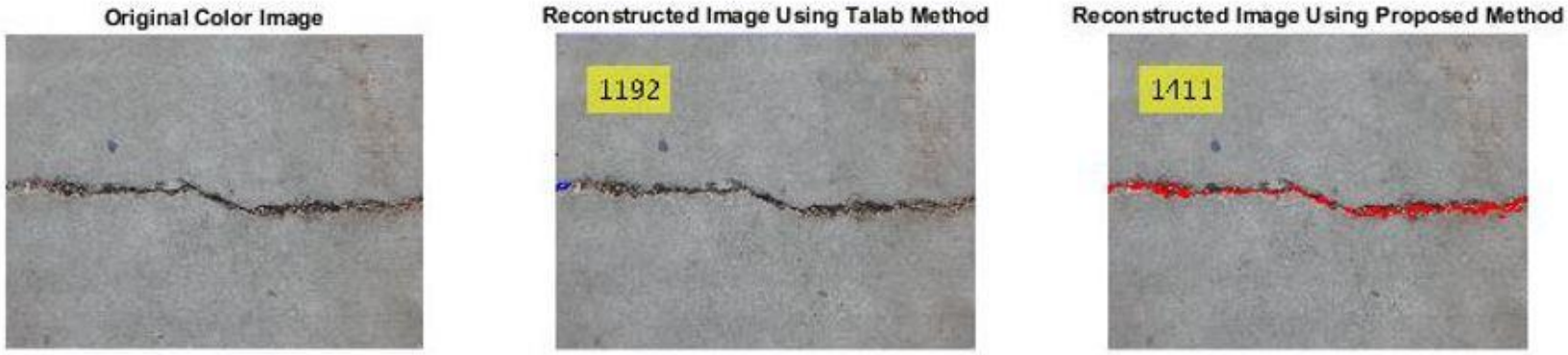


Figure 38. The crack identified by the proposed method compared with the FN report of Talab's work

The number of TN and FP reports is calculated based on the images with no cracks (16 images). Some of these images contain background lines, scenery lines, water marks, and edges of the concrete. The proposed method reported 11 images as crack-free images (TN) whereas Talab's method reported 6 images as crack-free images. In Figure 39, an image with no defects has been investigated by Talab's method and the proposed method. Since there are no cracks in the original color image, the expected output is a reconstructed, superimposed color image with 0 cracked pixels. The reconstructed image obtained by Talab's algorithm ended up reporting 1527 cracked pixels, which is an example of FP report. The proposed method, on the other hand, reported 0 cracked pixels which is an example of TN. The number of FP for Talab's method and the proposed method was 10 and 5, respectively. Even though the results obtained by the proposed method are significantly better than the results obtained by Talab's method, the number of FP reports are still high. Scenery lines, edges, and watermarks are the main culprits for the high rate of FP reports. In Figure 40, a concrete column is shown with the edges and background scenery. Both methods failed to separate cracks from other objects. The proposed method reported 3090 cracked pixels and Talab's method reported 3081 cracked pixels, while there are no cracked pixels in the original image. The comparison between Talab's method and the proposed method is tabulated in Table 1.

Table 1. Comparison between Talab's method and the proposed method.

	True Positive	False Negative	True Negative	False Positive
Talab's method	52%	48%	37.5%	62.5%
Proposed method	88%	12%	68.75%	31.25%

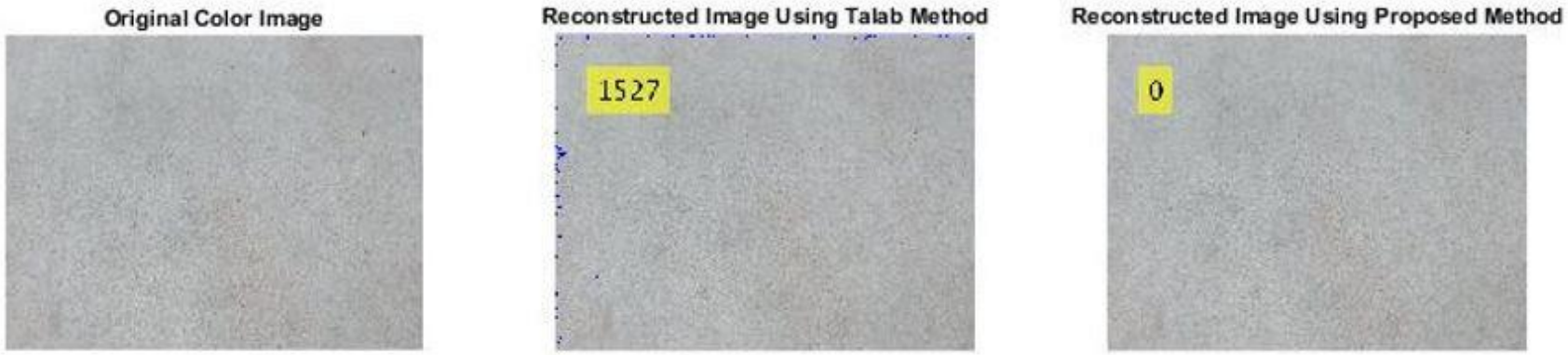


Figure 39. The comparison between two methods reporting TN

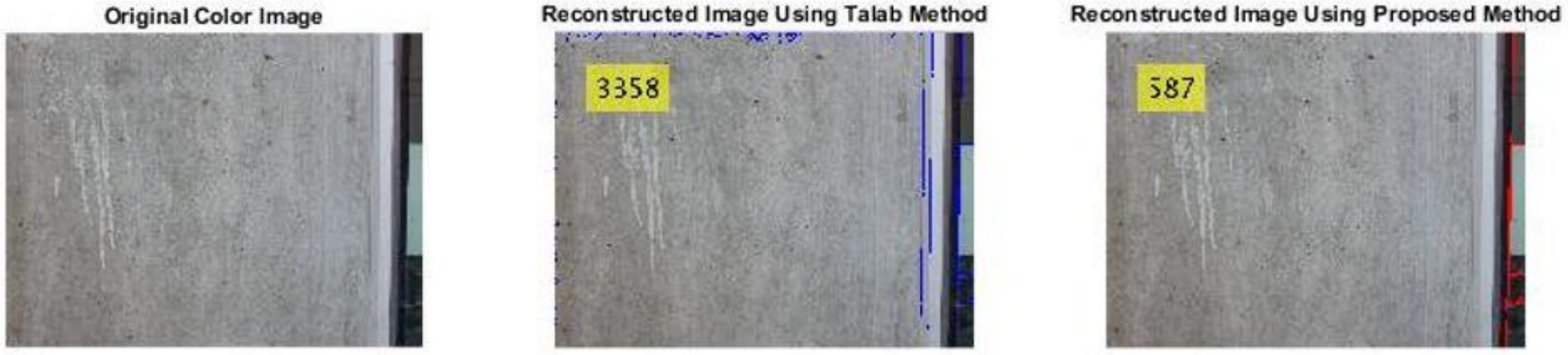


Figure 40. The FP report by two methods

The average execution time per image for the proposed method and Talab's method was 0.0859 and 0.0739 seconds, respectively. Since multilevel computational steps are employed in the proposed method to enhance its performance, the increase of computational time is reasonable.

4.3 EFFECTIVENESS OF THE PROPOSED METHOD

The proposed method achieved promising results over the Talab's method. However, there are some drawbacks, which will be discussed in this section.

The proposed method is developed based on the gradient of the pixel's intensities in the converted greyscale image. It is straight forward and computationally efficient and the algorithm can be directly applied on the raw images without training. However, the program is unable to detect small cracks. When a small crack is processed using the proposed method the change of the pixel's intensities is not tangible. The same argument can be made for Talab's method since both methods are developed based on the Sobel edge detector. Figure 41 shows a small crack on the concrete surface. The corresponding pixels with the small crack in the original image have not been identified at the very beginning of the proposed method (i.e., in Edge Image). Applying the median filter before Sobel edge detector masks has smoothed the image, and eliminated most of the surface roughness. However, smoothing the surface almost made the crack disappear (Median Filtered Image in Figure 41). Processing the same image with Talab's algorithm, the crack was not detected after using Otsu's thresholding operation. Removing objects with an area less than 30 pixels in the final step added the remaining transparent pixels associated with the crack to the background. Thus, the final output images obtained from both methods reported FN (Figure 43).

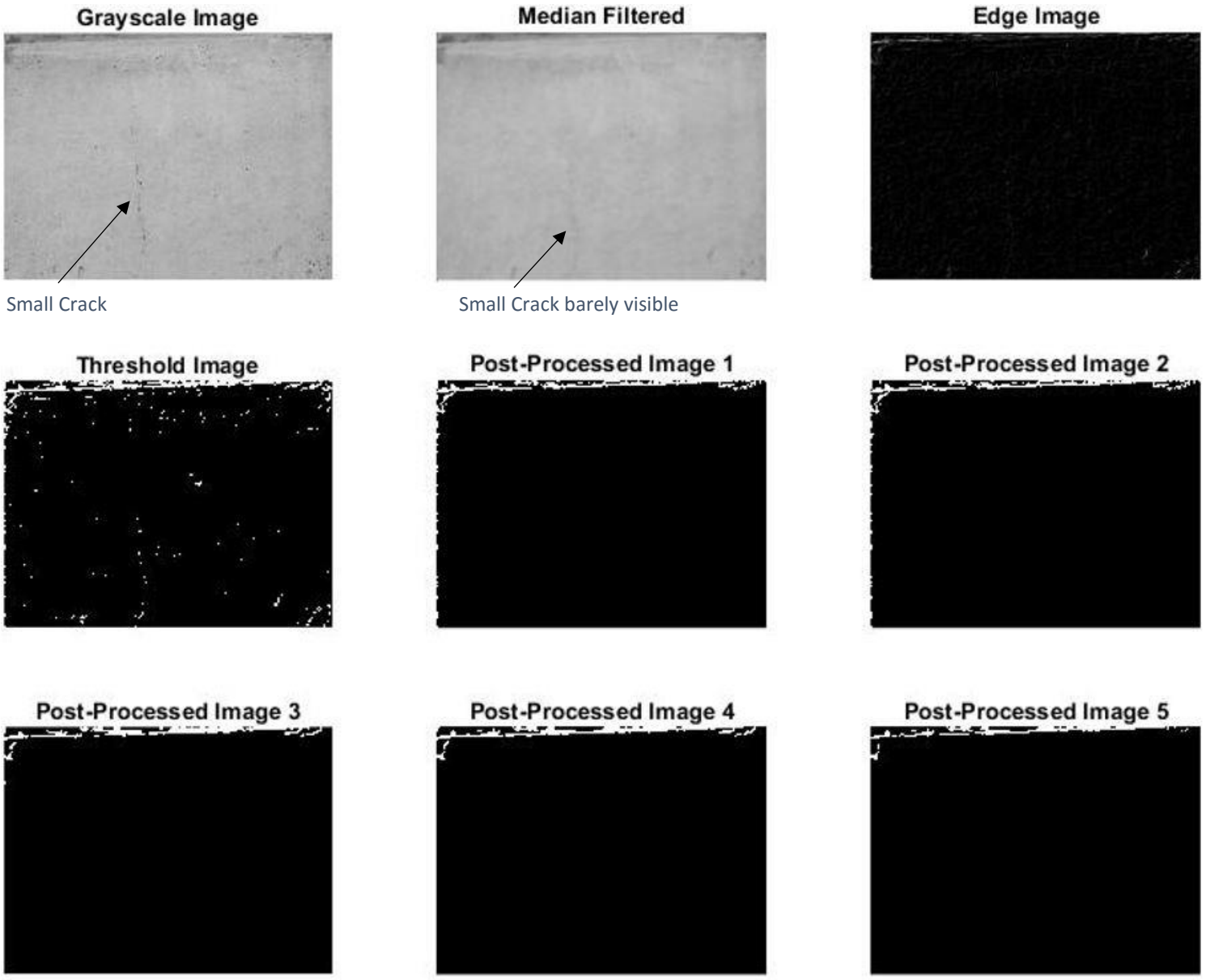


Figure 41. Unidentified small crack using the proposed method

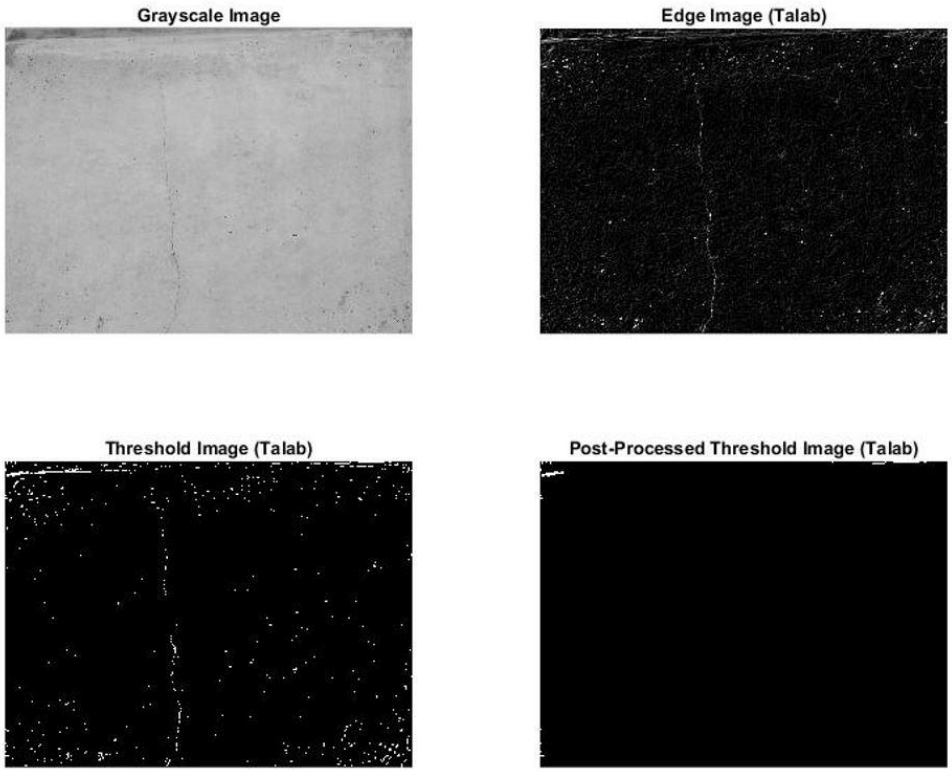


Figure 42. Unidentified small crack using Talab's method

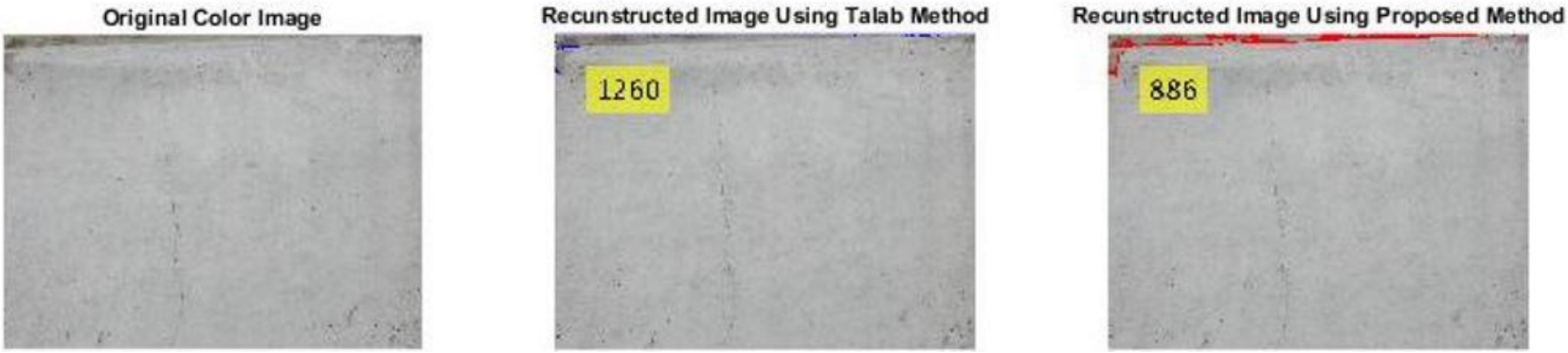


Figure 43. Comparing the proposed method and Talab's method to identify small cracks

Despite the attempts, the proposed method was not able to completely classify cracks, scenery lines, edges, water marks, and stains. Most of the FP reports are associated with the scenery lines, edges, water marks, and stains. However, each of the attempts partially improved the proposed method efficiency. Removing straight lines in Post-Processed Image 2 eliminated straight objects from the image. Figure 44 shows an example of how the operation in Post-Processed Image 2 removed 1 vertical edge corresponding to the concrete column. The reason for this operation to be unsuccessful is the border objects. As it can be seen, there are false target pixels (non-cracked) at the borders of the image in the fourth step (Post-Processed Image 1). Since almost all the post processed operations are based on the connected components in the binary image, the program considers each connected component for orientation. The long vertical edges in Figure 44 are connected to the top border objects and the orientation for the entire connected component was not near either 0 or 90 degrees. As a result, some vertical lines are not removed.

The properties of the cracks in HSV color space were used to invent an index to separate cracks and other objects was by the. Applying a threshold value base on the minimum and the standard deviation of the cracked pixels in S component of the color image was partially effective. The edges of the concrete were almost completely removed by of the operation in HSV domain as shown in Post-Processed Image 5 in Figure 45. However, using crack properties in HSV color space is not always effective. This operation was based on the observation that the cracks have different intensity values than the intensity values of the other objects in S component. So, the thresholding value was defined to preserve the pixels with smaller values than the threshold defined by Eq. (3-1). Although this observation is true for this report's database, other objects can sometimes pass through the thresholding operation and be preserved as cracked pixels. The

scenery lines and concrete edges in Figure 44 are examples that the operation in Post-Processed Image 5 was not as effective as it was expected.

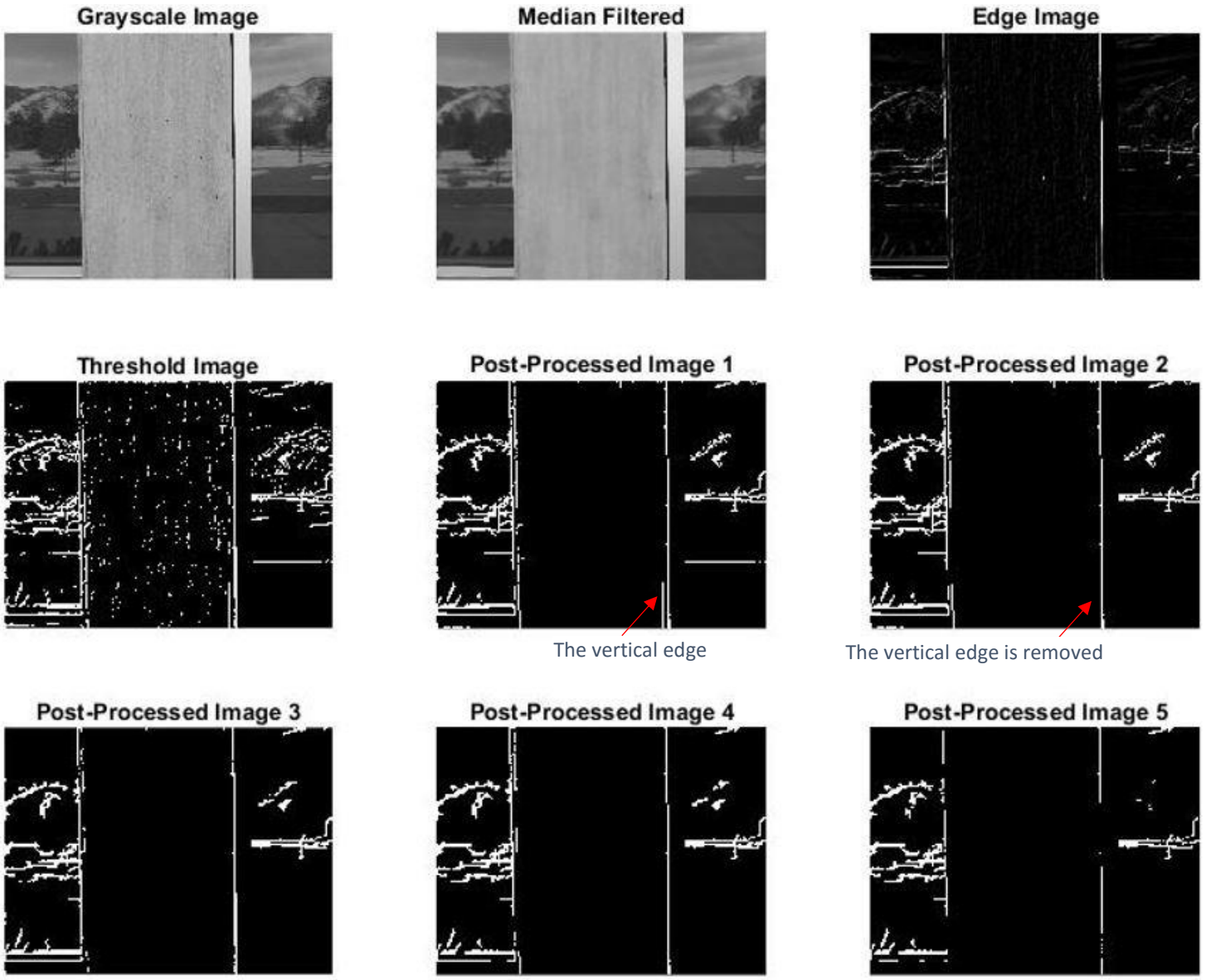


Figure 44. Partially removal of an object based on the orientation using the proposed method

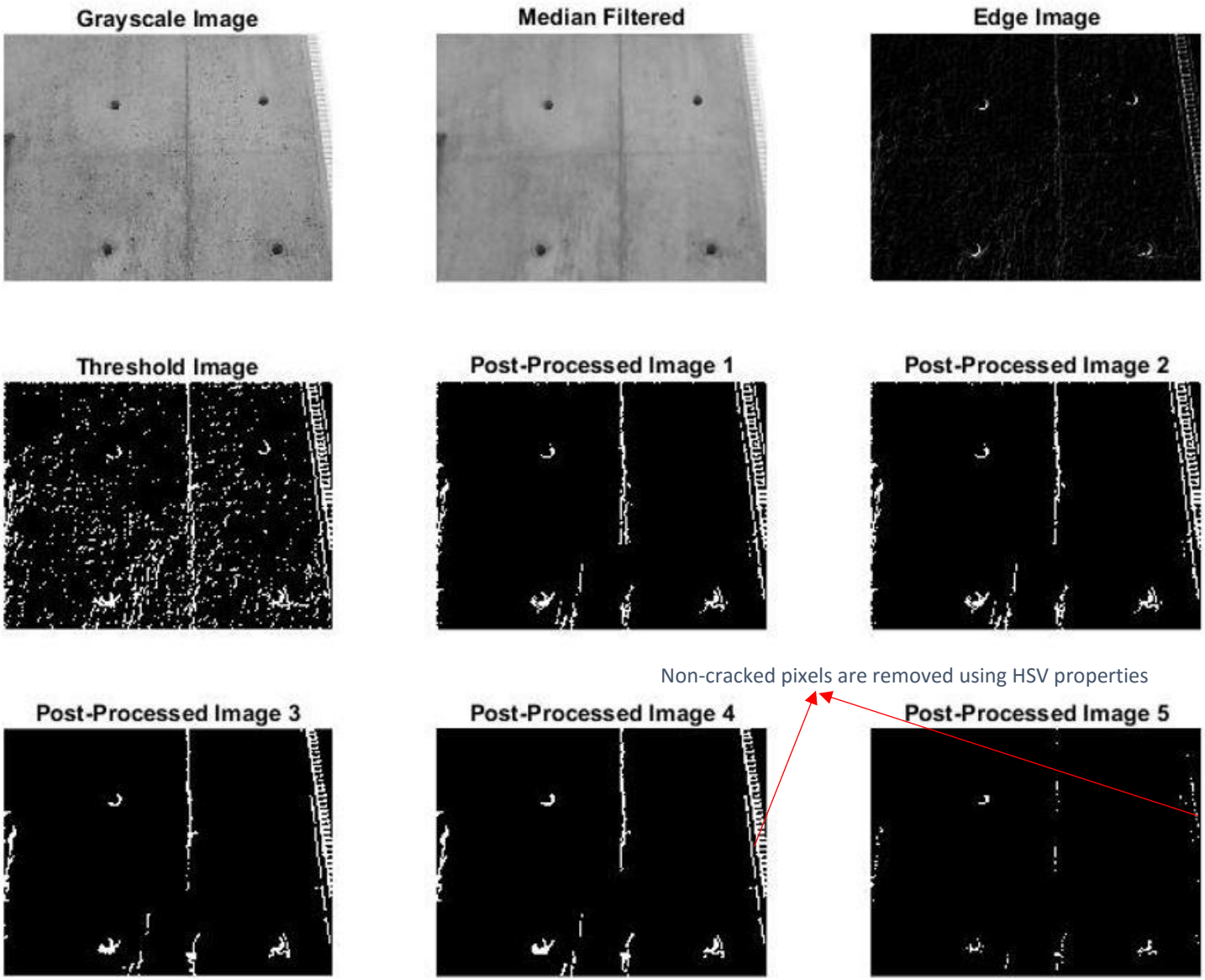


Figure 45. Edge removal of the concrete member using HSV operation in the proposed method

Form hole is a type of surface defect that can result in further damage to the structure. These holes are the side effect of the form work, and are inevitable in concrete construction. Bearing that in mind, these holes usually are benign but as time passes the structure experiences erratic temperature change and ice cycling. The form holes can get bigger and deeper by this time, and allow the moistures or corrosive materials to penetrate through concrete. Thus, inspection of the form holes as another surface defect on concrete structures is important in structural health monitoring. The proposed method was able to detect the form holes as Figure 46 demonstrates that, 9 out of 9 form holes in the original image have been detected by the proposed method. In addition, the vertical and horizontal edge in the image were removed in Post-Processed Image 5. The same input image was processed by Talab's method and the result is shown in Figure 47. None of the form holes were detected, and the horizontal edge was reported as crack pixels in the Post-Processed Threshold Image (Talab). The final results of the two compared methods for the same input image are shown in Figure 48.

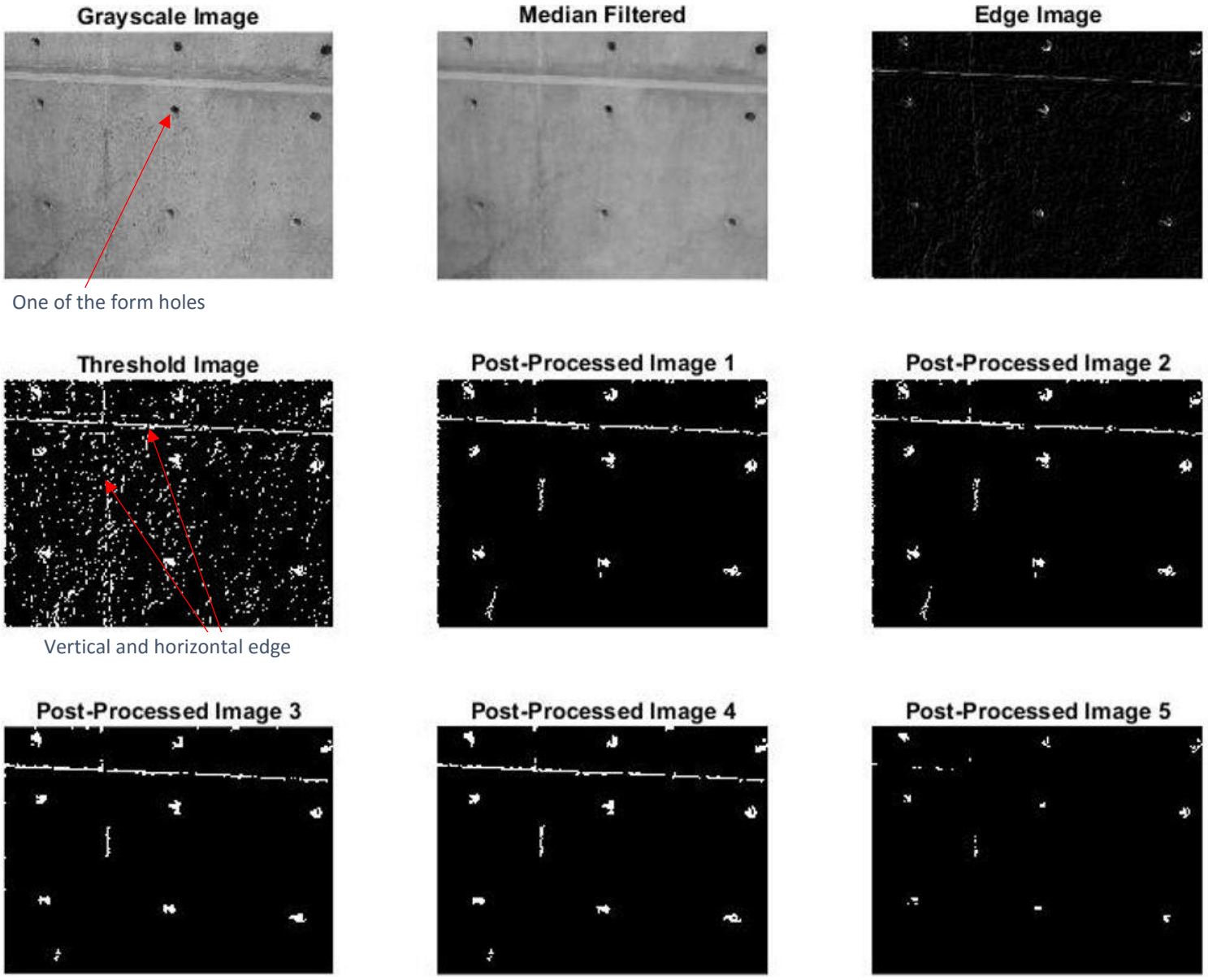


Figure 46. Form hole detection using the proposed method

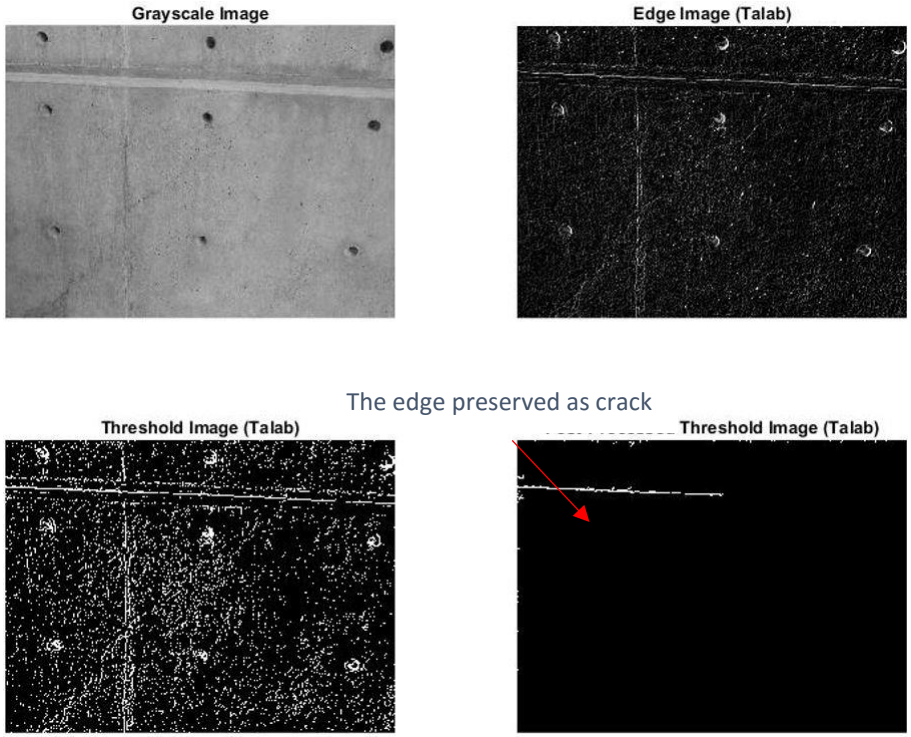


Figure 47. Form hole detection using Talab's method

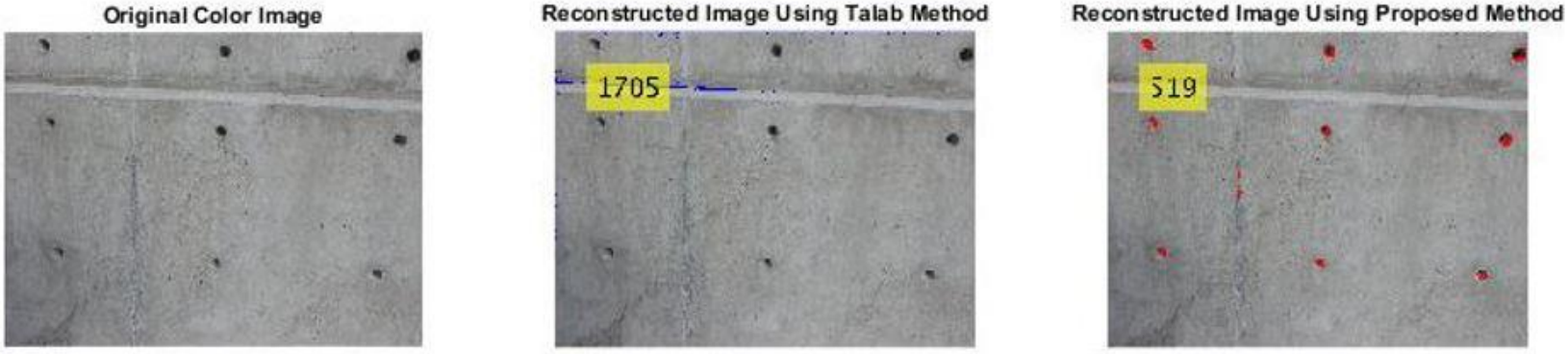


Figure 48. The comparison between Talab's method and the proposed method in form hole detection

The four indicators (i.e. TP, TN, FP, and FN reports) mentioned in part 4.2 of this section proved that the proposed method can detect surface cracks in a more efficient manner. The quality of the crack detection using the proposed method and Talab's method can also be compared. There were more cases where the proposed crack identification method was more effective.

Figure 49 shows the process of the crack detection by the proposed method. The original image had a horizontal crack with watermarks on the surface. Although some of the pixels corresponding to the watermarks were identified as cracks at the final stage, the crack was detected clearly. Using the same input image with Talab's method, there are no signs of cracks or watermarks. Instead of crack detection, the Talab's method identified target pixels on the top and left border of the Post-Processed Threshold Image (Talab) as crack pixels as shown in Figure 50. The border pixels are obviously not consistent with the crack in the original image. Figure 51 shows the comparison of the two methods in finding cracks on the same input image.

Figure 52 shows the intermediate results of the proposed method for an image involving a cross crack (one vertical and one horizontal). All of the cracks in the original image have been identified, but the right side of the horizontal crack is discontinuous. This discontinuity occurred in Post-Processed Image 5, after applying the HSV properties. The crack is small, and there are pixels whose intensities are similar to the intensities of the concrete surface after applying the median filter (Median Filtered). The results of applying Talab's method on the same image is shown in Figure 53. As it can be seen, the discontinuity for the right side of the horizontal crack is no longer an issue, but the algorithm proposed by Talab completely ignored the lower half of the vertical crack. In addition, the border pixels are detected as cracks in the final image. In Figure 54, the comparison results of both methods are shown. More examples can be found in Appendix B of this report.

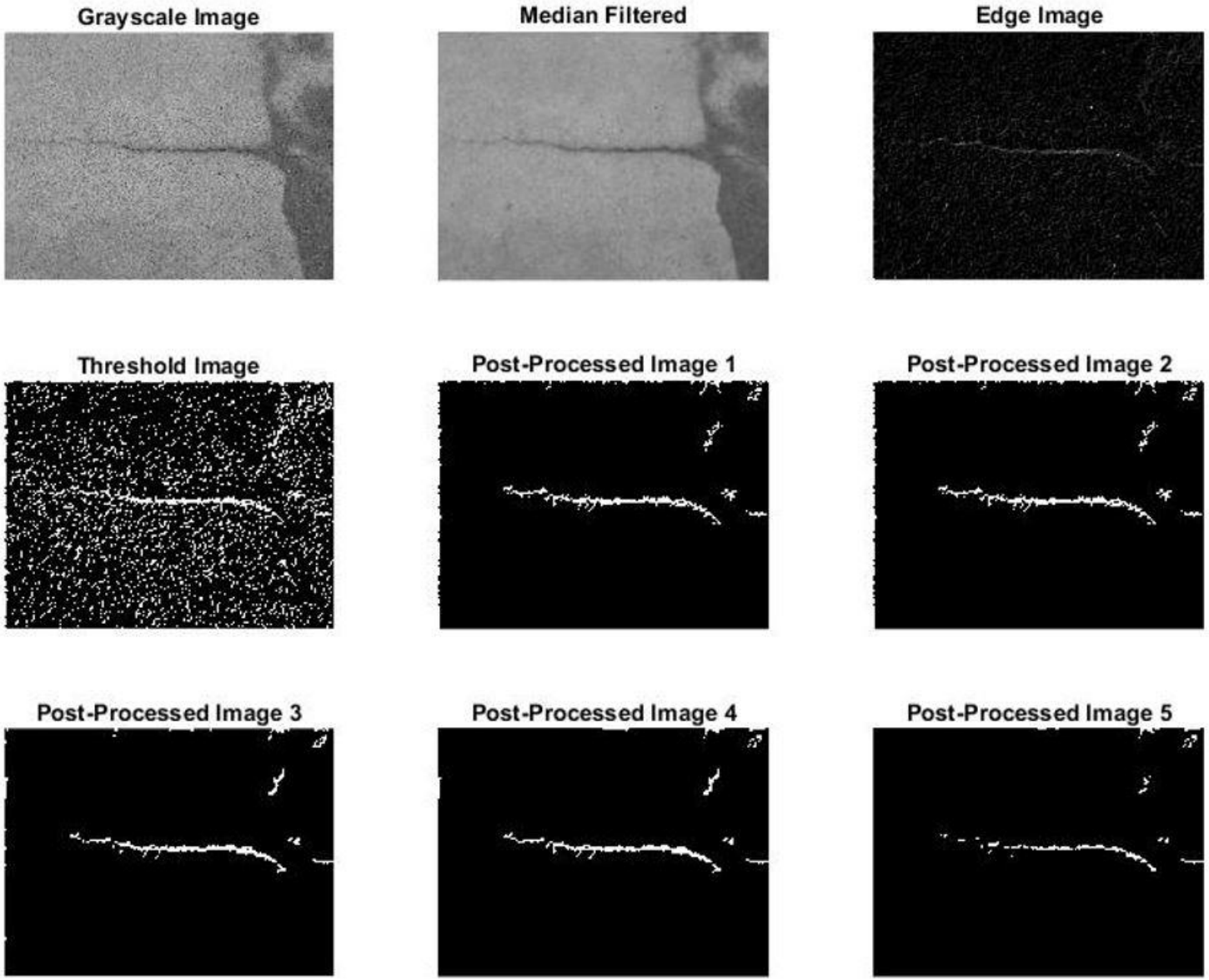


Figure 49. Performance of the proposed method on an image with crack and water marks

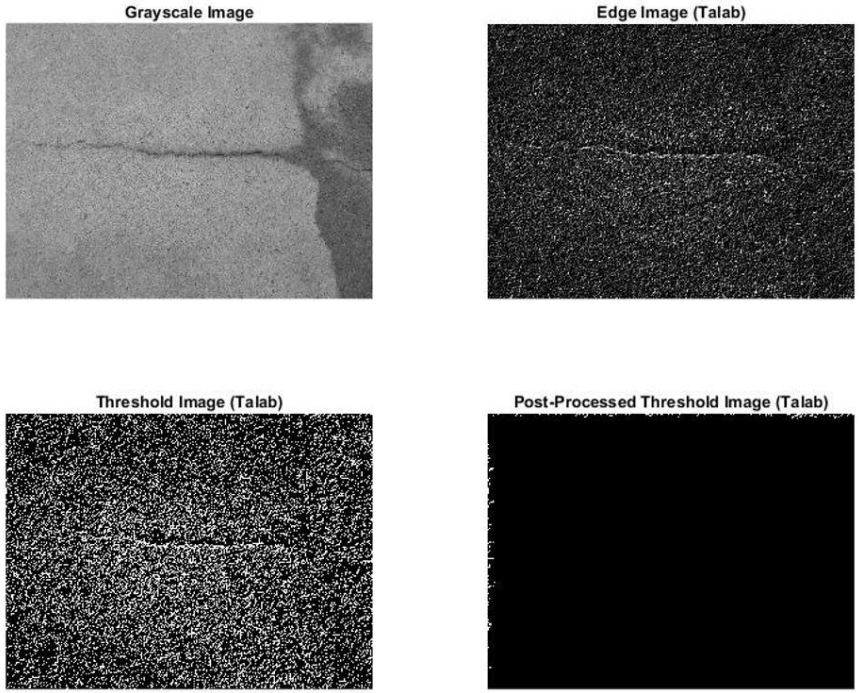


Figure 50. Performance of Talab's method on an image with cracks and water marks

Figure 51. Comparison of performances of the both methods on an image with cracks and water marks

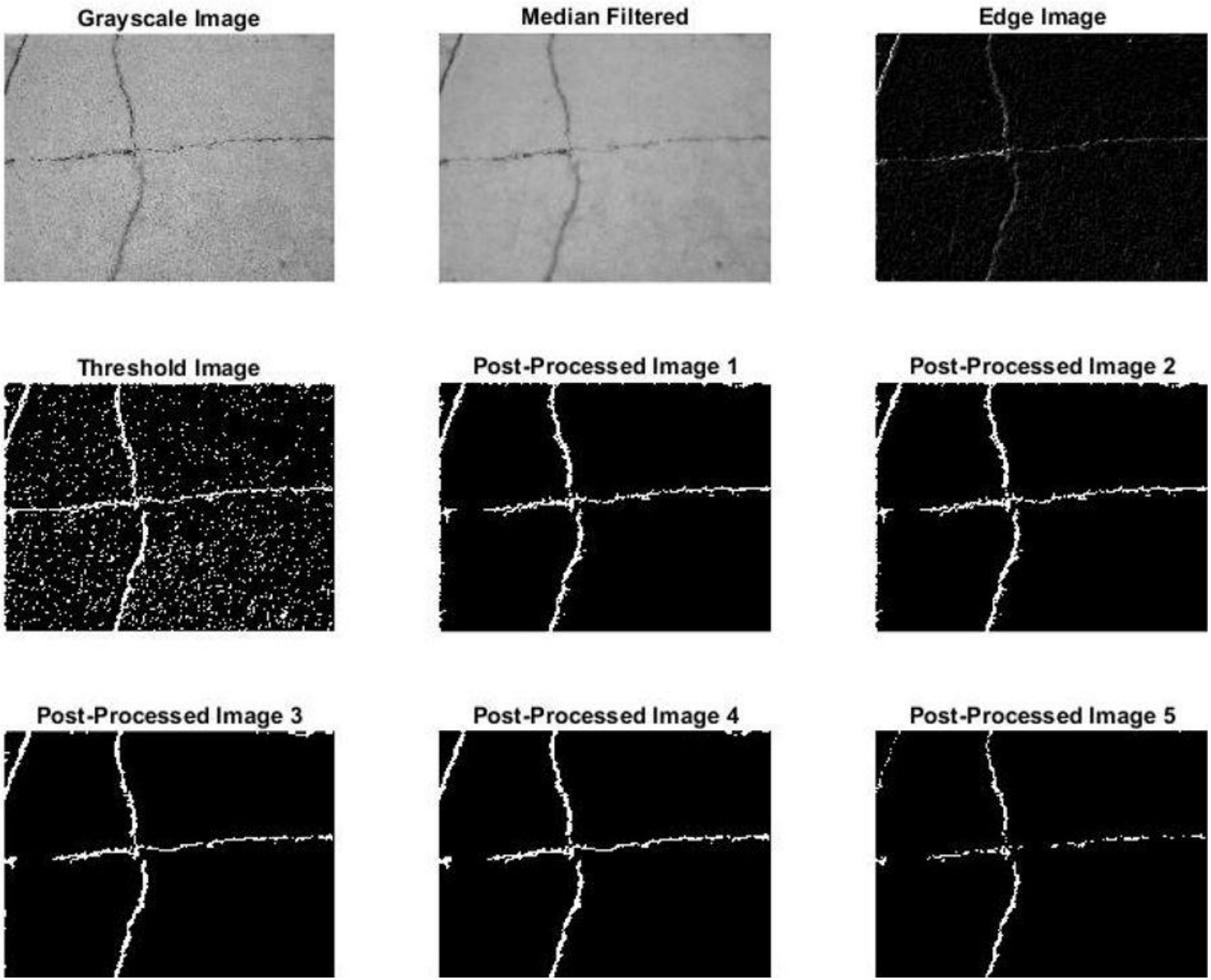


Figure 52. Discontinuity in crack detection using the proposed method

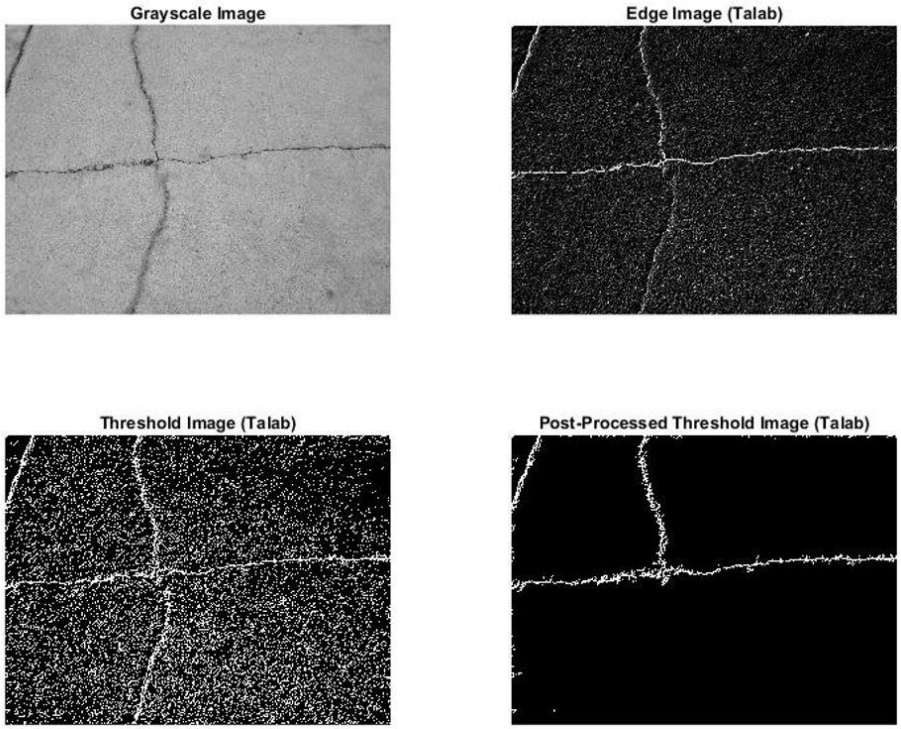


Figure 53. Partially effective crack detection using Talab's method

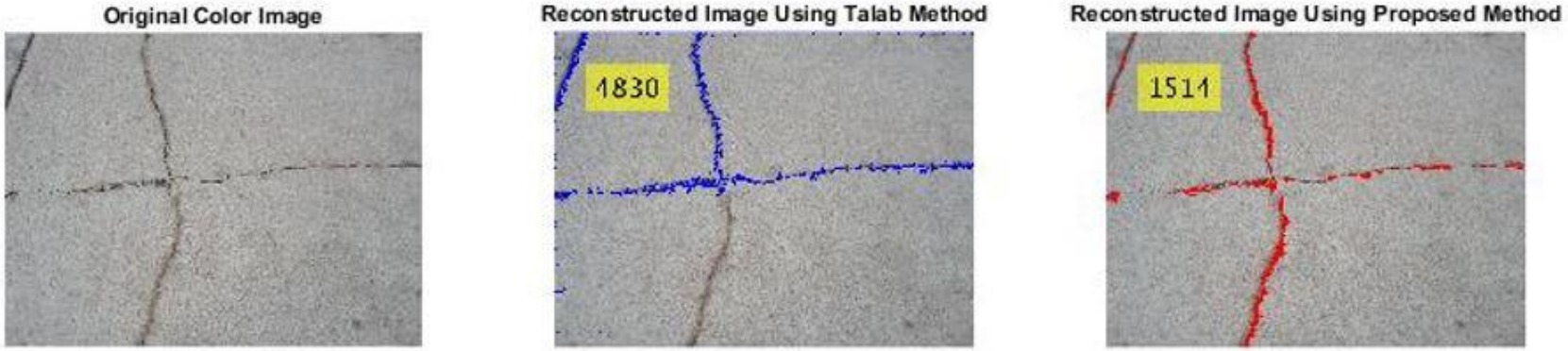


Figure 54. Comparison between Talab's method and the proposed method in cracks detection quality

5. CONCLUSIONS AND FUTURE WORK

5.1 SUMMARY

In this report, an effective method is proposed for automatic surface crack detection in concrete structures. The proposed method was built on the method proposed by Talab, (Talab, et al. 2015) and categorized as a method in the spatial domain without training. Since it is in the spatial domain, the results of each step were more tangible to the user and the algorithm is more user friendly. The proposed algorithm detects and separates cracks based on their properties using a multilevel operation. In addition, the proposed method is able to directly identify cracks while most of the current crack detection techniques use training databases and trained classificatory such as decision lines. The proposed method is compatible with different databases and it is less time consuming.

The algorithm proposed by Talab has the following steps:

- Convert RGB image to grayscale
- Use the Sobel edge detector to find edges (including cracks)
- Use the Otsu's thresholding value to obtain the binary image
- Find the connected components in binary image with an area less than 30 pixels and add them to the background, i.e. removing them from the picture

The following steps are required to find cracks in the proposed method:

- The original RGB image is read directly from the camera
- The grayscale image is converted from the original image
- A median filter is applied on the grayscale image to smooth the surface
- Sobel edge detectors are applied to magnify edges in the image

- Otsu's thresholding method is applied to obtain the binary image
- Connected components with an area less than 200 pixels are identified and removed
- Connected components with an orientation degree of 0 and 90 are identified and removed
- Morphological operation 'majority' is applied to connect the objects and fill the holes in them
- Objects with total pixels less than 50 are detected and removed
- The components of the original image in the HSV color space are calculated
- Pixels within the connected components is the HSV color space are kept as candidate crack pixels
- A new thresholding value is calculated based on Eq. (3-1)
- If a candidate crack pixel's S value is less than the threshold value calculated by Eq. (3-1), the pixel is added to the background (non-cracked)
- If a candidate crack pixel's S value is equal or greater than the threshold value calculated by Eq. (3-1), the pixel is preserved in the binary image (cracked)
- The cracked pixels are superimposed on the original image with and the total number of crack pixels are computed.

In the proposed method, the original color image was converted to a grayscale image. A median filter was designed to smooth the image and reduce the false positive reports. However, in case of narrow cracks, the median filter can eliminate the crack, which will lead to a FN report. The Sobel edge detectors in both horizontal and vertical directions were applied on the grayscale image to generate two derivatives. Intermediate images which are combined to construct an image

with intensified edges. The Otsu's thresholding algorithm is used to convert the intensified edge image to a binary image. Objects with an area less than 200 pixels were removed. The connected components with the orientation of 0 and 90 degrees were then removed assuming the cracks are very unlikely to exist in shape of vertical and horizontal lines. This operation was able to remove some of the vertical and horizontal lines that were not connected to the other target pixels on the border. However, applying this operation can cause the removal of cracks with a vertical or horizontal ellipse. Morphological operation was applied to remove small obsolete objects from the binary image and connect the objects with more neighbor target pixels together. Objects with an area less than 50 pixels were eliminated. Using the crack properties in S component of the HSV color space, the algorithm separates the cracks from other surface irregularities such as scenery line, edges, and watermarks. Although this operation was not very successful regarding scenery lines, it showed promising results for member edges and formed lines on concrete surface. The results of the crack detection using the proposed method over 41 input images (25 images with defects and 16 images without defects) have been showed in Appendix A.

5.2 CONCLUSION

The proposed method achieves 88% and 12% of TP and FP rate, respectively. The TP rate of Talab's method (e.g., 52%) was significantly lower than the TP rate of the proposed method rate and the FN rate of Talab's method (e.g., 48%) was four times greater than the FN rate of the proposed method (e.g., 12%). On the other hand, the proposed method showed improved rates in terms of TN and FN with 69% and 31%, respectively. Talab's algorithm detected 10 images containing cracks out of 16 images, where there were no defects existing in the original image (i.e. 63% of FP report).

The execution time for Talab's method was 0.0739 seconds per image, which is slightly less than the execution time for the proposed method (e.g., 0.0859 second per image). However, both quantity and quality of crack detection using the proposed method were improved, considerably.

Comparing the proposed method and Talab's method, the following conclusions can be made from this report:

- The execution time of the proposed method was increased by 0.12 seconds per image
- TN reports increased by 35% over 25 images with cracks
- TN reports was doubled over 16 images without cracks
- The proposed method was able to separate the cracks from other surface irregularities to some extent
- The proposed method was able to find form holes

The final resultant images using both methods for all image in the databases can be found in Appendix B.

5.3 LIMITATIONS

Executing the proposed method and Talab's method on crack with significantly different widths and orientations can present some problems. Shadows, watermarks, background lines and paints with dark color are likely to be mistaken with the cracks, and often identified as such. In Figure 55, the watermark in the left top corner of the original picture were detected as cracks in both approaches. Both Talab's method and the proposed method were unable to eliminate background lines and some edges as seen in Figure 56.

Both methods assume that cracked pixels have considerably darker intensity. This assumption is often valid. However, when the crack is wide enough, the crack pixel values are not significantly less than the pixel values of the texture of the concrete, the crack pixels are not identified.

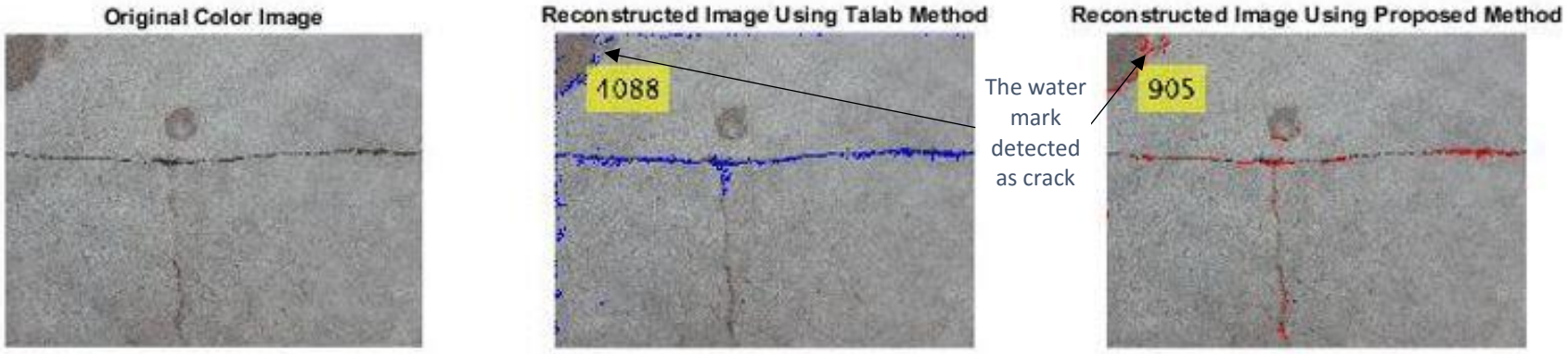


Figure 55. The limitation of both methods to separate water-marks from cracks

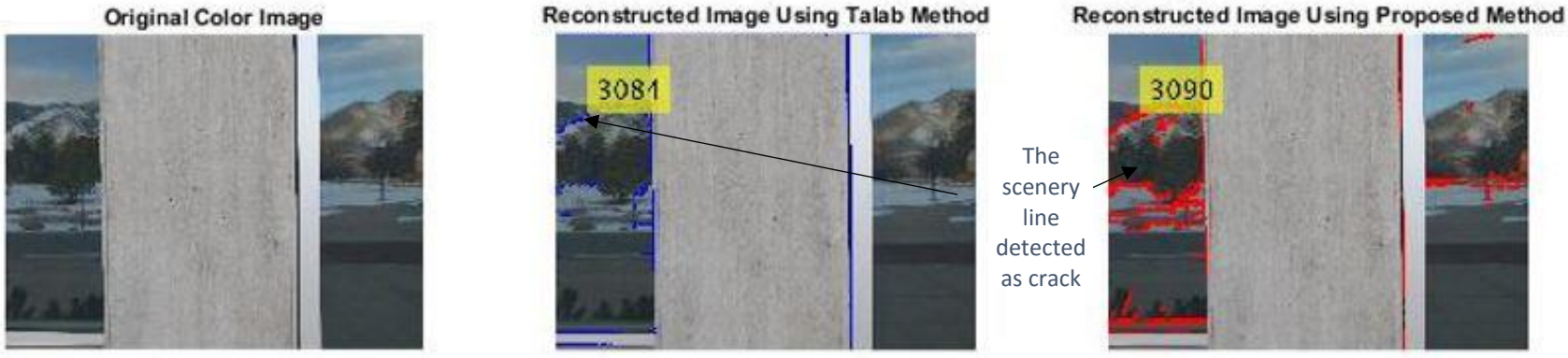


Figure 56. The limitation of both methods to separate background lines from cracks

5.4 FUTURE WORK

Automated, image-based crack detection methodologies explained here are relatively new to structural engineering, and has a vast potential for future studies. In the proposed method, the properties of crack pixels in HSV color space domain were investigated. A thresholding value was defined, which excludes the edges of the concrete from the cracks. Though this technique was partially successful, it seems to be possible to investigate the crack properties in LAB color space. A LAB color space is a color-opponent space with dimension L for lightness, and A and B for the color-opponent dimensions. In other words, if the crack pixels are found to have certain properties in HSV or LAB, the cracks and background line, or cracks and edges, could effectively be separated by the program. More work in this area can lead to a smart algorithm, with the ability to categorize diverse lines detected in gray-scale image and remove the pixels inconsistent with cracks.

The cut-off area thresholding can be improved to become more dynamic by using the intensities of the pixels in each image rather than having a static number as it has been implemented in this report. Both methods are incapable of recognizing small cracks that are significantly smaller than a pixel width, even though they are much longer than a pixel. Small cracks classified as fatigue cracks in concrete structures are very common, and can occur from repeated dynamic loading and unloading. Fatigue cracks play a major role in structural health monitoring both in steel and concrete structures. However, current automated algorithms are not effective at identifying them. Improvements in these areas will result in more widespread adoption of automated image processing based crack detection in the field of structural engineering.

6. REFERENCES

2016. http://www.had2know.com/technology/hsv-rgb-conversion-formula-calculator.html.

2016. March 16. https://en.wikipedia.org/wiki/HSL_and_HSV.

Ammouchea, A., D. Breysseb, H. Hornaina, O. Didryd, and J. Marchandc. 2000. "A new image analysis technique for the quantitative assessment of microcracks in cement-based materials." *Cement and Concrete Research* 25-35.

Aoki, S., and T. Nagao. 1999. "Automatic construction of treestructural image transformation." *International Conference on Image Processing.* Washington DC: IEEE Computer Society .

Ikhlas Abdel-Qader, P.E, P.E., M.ASCE Osama Abudayyeh, and and Michael E. Kelly. 2003. "Analysis of Edge-Detection Techniques for Crack Identification in Bridges." *JOURNAL OF COMPUTING IN CIVIL ENGINEERING* 255-263.

Kim, Jong-Woo, Sung-Bae Kim, Jeong-Cheon Park, and Jin-Won Nam. 2015. "Development of Crack Detection System with Unmanned Aerial Vehicles and Digital Image Processing." *Advances in structural engineering and mechanics (ASEM15).* Inchoen, Korea: I-ASEM.

Kittler, J., and J. Illingworth. 1986. "Minimum error thresholding." *Pattern Recognition* 41-47.

Kittler, J., R. Marik, M. Mirmehdi, M. Petrou, and J. Song. 1994. *DETECTION OF DEFECTS IN COLOUR TEXTURE SURFACES.* Guildford: University of Surrey.

Litorowicz, Agnieszka. 2006. "Identification and quantification of cracks in concrete by optical fluorescent microscopy." *Cement and Concrete Research* 1508-1515.

Matsumoto, Masato, Koji Mitani, and F. Necati Catbas. 2013. *BRIDGE ASSESSMENT METHODS USING IMAGE PROCESSING AND INFRARED THERMOGRAPHY TECHNOLOGY*. Orlando: University of central Florida.

Moon, Hyeong-Gyeong, and Jung-Hoon Kim. 2011. "Inteligent Crack Detecting Algorithm On The Concrete Crack Image Using Neural Network." *28th ISARC*. Seoul. 1461-1467.

Nishikawa, Takafumi, Junji Yoshida, Toshiyuki Sugiyama, and Yozo Fujino. 2012. "Concrete Crack Detection by Multiple Sequential Image Filtering." *Computer-Aided Civil and Infrastructure Engineering* 29-47.

Oliveira, Henrique, and Paulo L. Correia. 2013. "Automatic Road Crack Detection and Characterization." *IEEE Transactions on Intelligent Transportation Systems* 155-168.

Otsu, N. 1979. "A Threshold Selection Method from Gray-Level Histograms." *IEEE Transactions on Systems, Man, and Cybernetics* 62-66.

Pereira, Fábio Celestino, and Carlos Eduardo Pereira. 2015. "Embedded Image Processing Systems for Automatic Recognition of Cracks using UAVs." *IFAC*. PortoAlegre: ELSEVIER. 016-021.

Rimkus, Arvydas, Askoldas Podviezko, and Viktor Gribniak. 2015. "Processing digital images for crack localization in reinforced concrete members." *Procedia Engineering* 239-243.

Sankarasrinivasana, S., E. Balasubramaniana, K. Karthika, U. Chandrasekarb, and Rishi Gupta. 2015. "Health Monitoring of Civil Structures with Integrated UAV and Image Processing System." *Procedia Computer Science* 508-515.

Talab, Ahmed Mahgoub Ahmed, Zhangcan Huanga, Xi Fan, and HaiMing Liu. 2015. "Detection crack in image using Otsu method and multiple filtering inimage processing techniques." *Optik* 1-4.

Wang, Hui, Zhang Chen, and Lijun Sun. 2013. "Image Preprocessing Methods to Identify Micro-cracks of Road Pavement." *Optics and Photonics Journal* 99-102.

Yamaguchi, Tomoyuki, and Shuji Hashimoto. 2010. "Image-Based Crack Detection for Real Concrete Surfaces images using percolation-based image processing." *Machine Vision and Applications* (Waseda University) 797-809.

Yamaguchi, Tomoyuki, Shingo Nakamura, Ryo Saegusa, and Shuji Hashimoto. 2008. "Image-Based Crack Detection for Real Concrete Surfaces." *TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING* 128-135.

Zhang, Wenyu, Zhenjiang Zhang, Dapeng Qi, and Yun Liu. 2014. "Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring." *Sensors* 19307-19328.

Zheng, Paul, and Cristopher Moen. 2014. *CRACK DETECTION AND MEASUREMENT UTILIZING IMAGE-BASED RECONSTRUCTION*. Blacksburg: VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY.

7. APPENDIX A: RESULTS OF CRACK DETECTION-THE PROPOSED METHOD

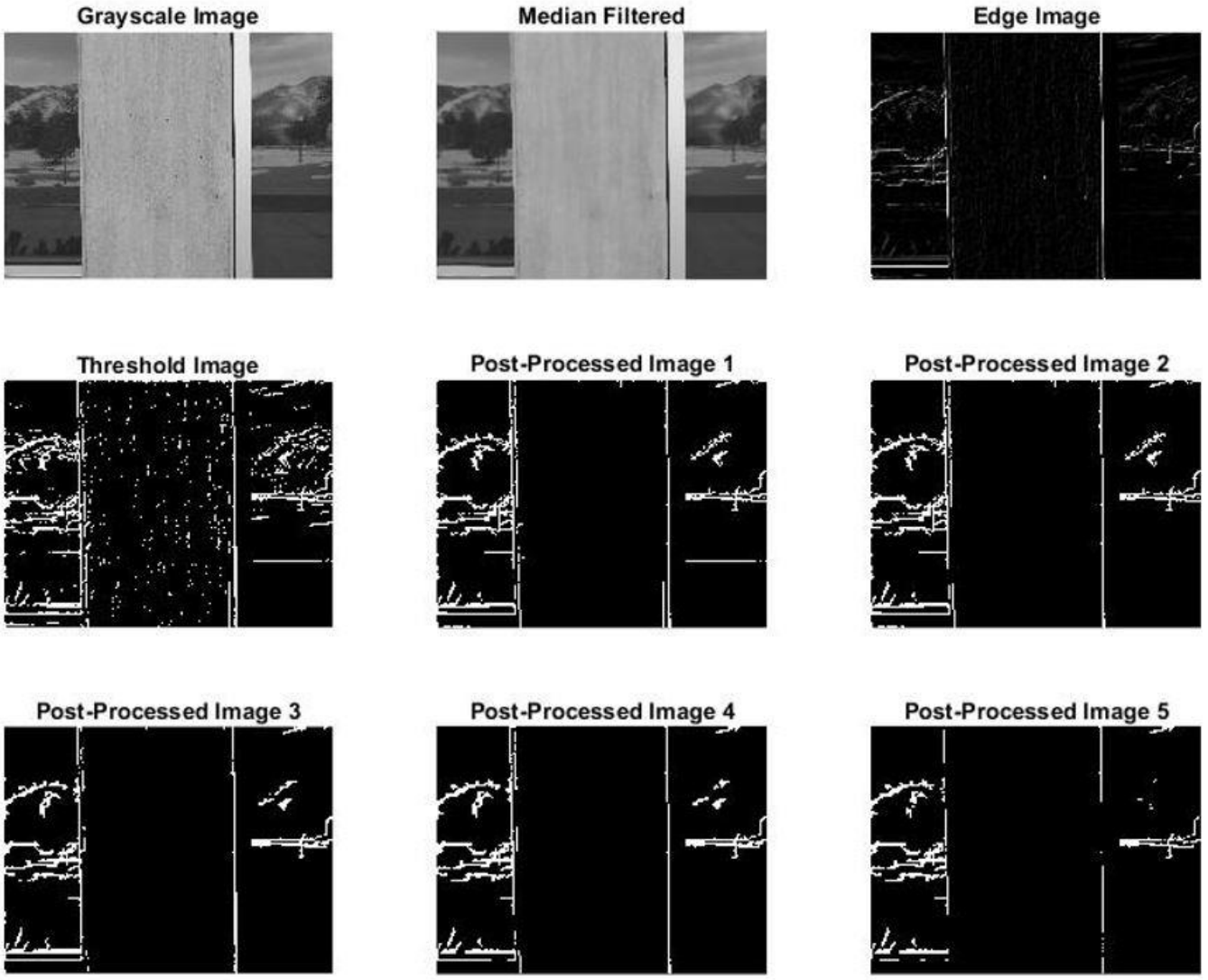


Figure A. 1. Crack detection results using the proposed method on database image #1

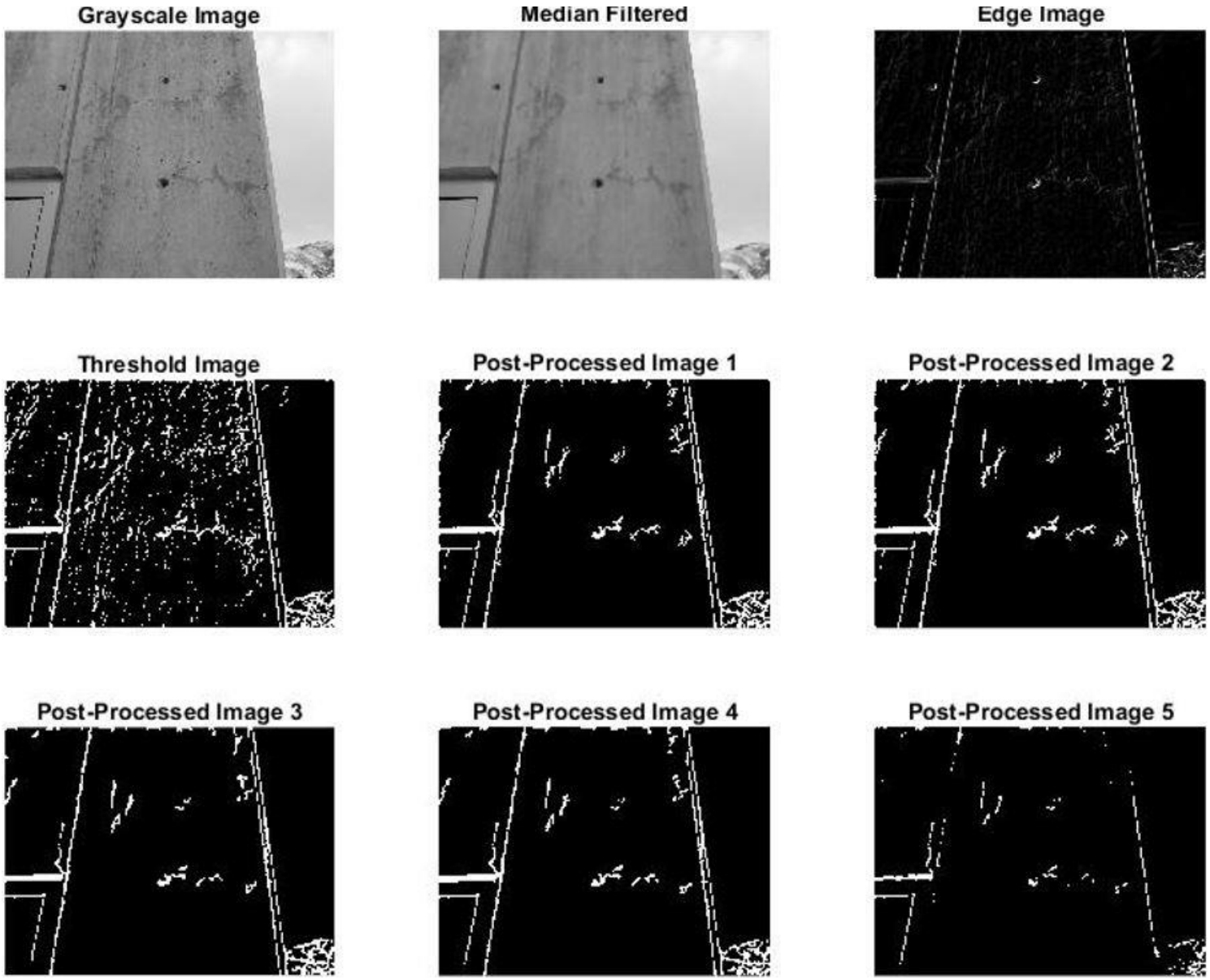


Figure A. 2. Crack detection results using the proposed method on database image #2

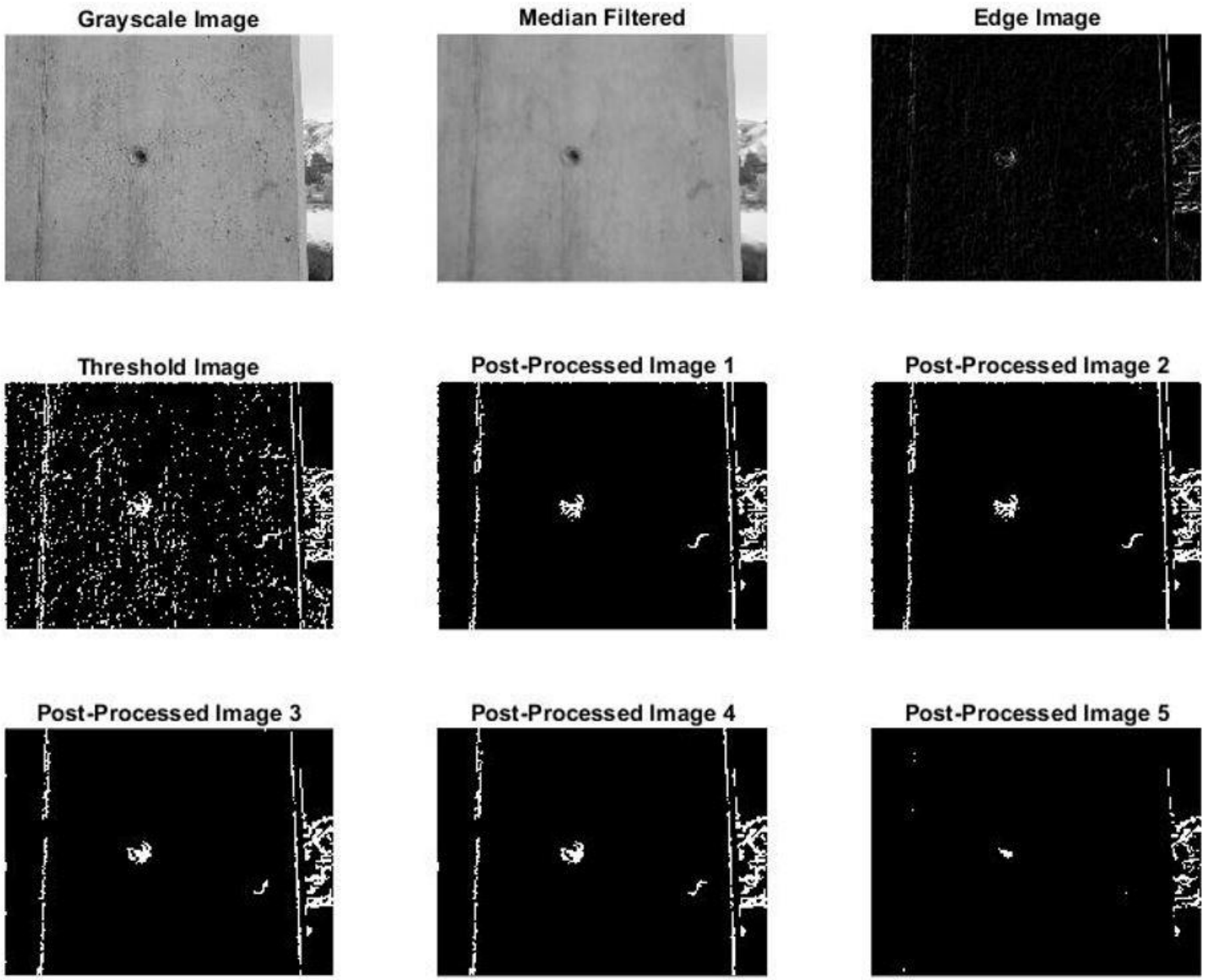


Figure A. 3. Crack detection results using the proposed method on database image #3

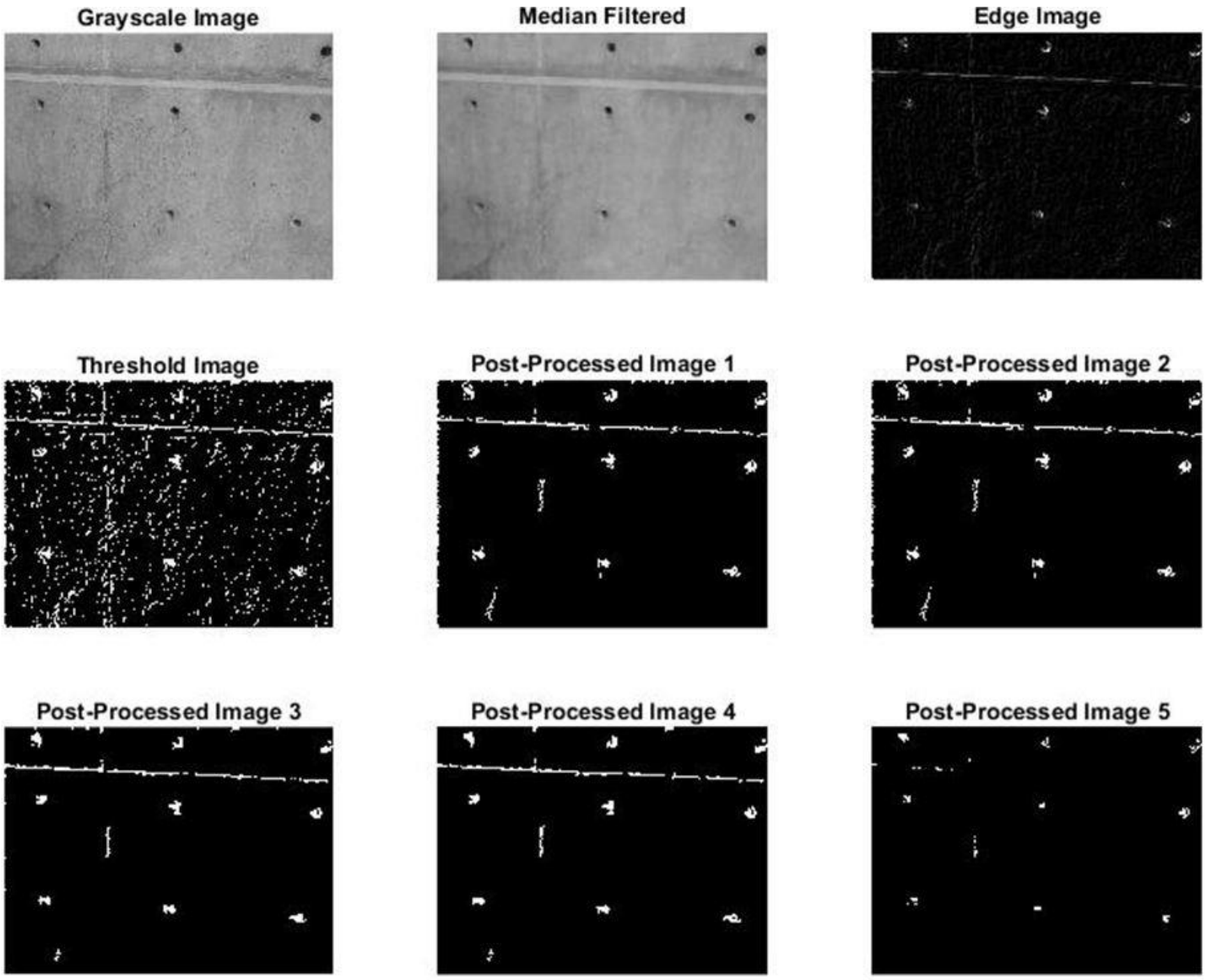


Figure A. 4. Crack detection results using the proposed method on database image #4

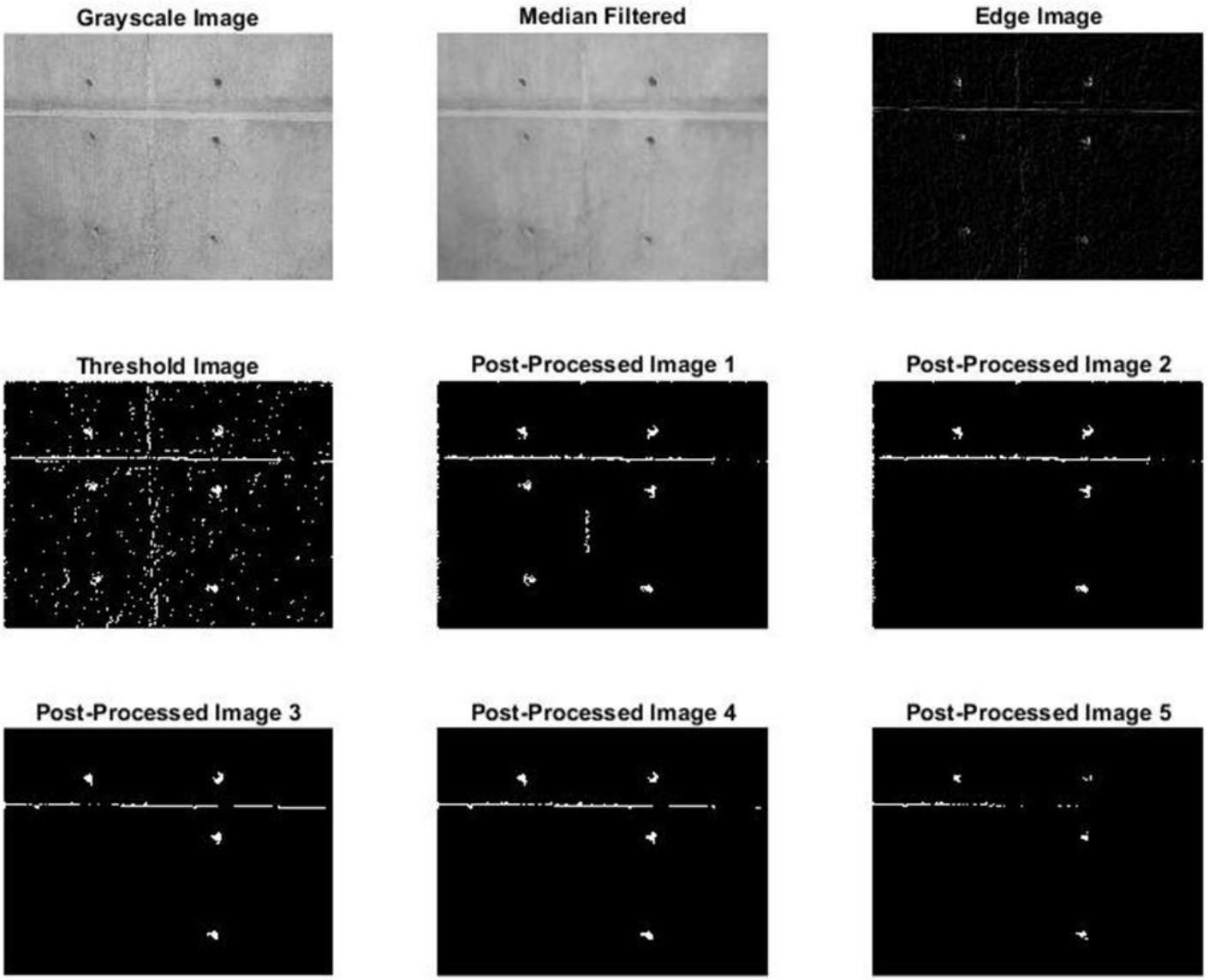


Figure A. 5. Crack detection results using the proposed method on database image #5

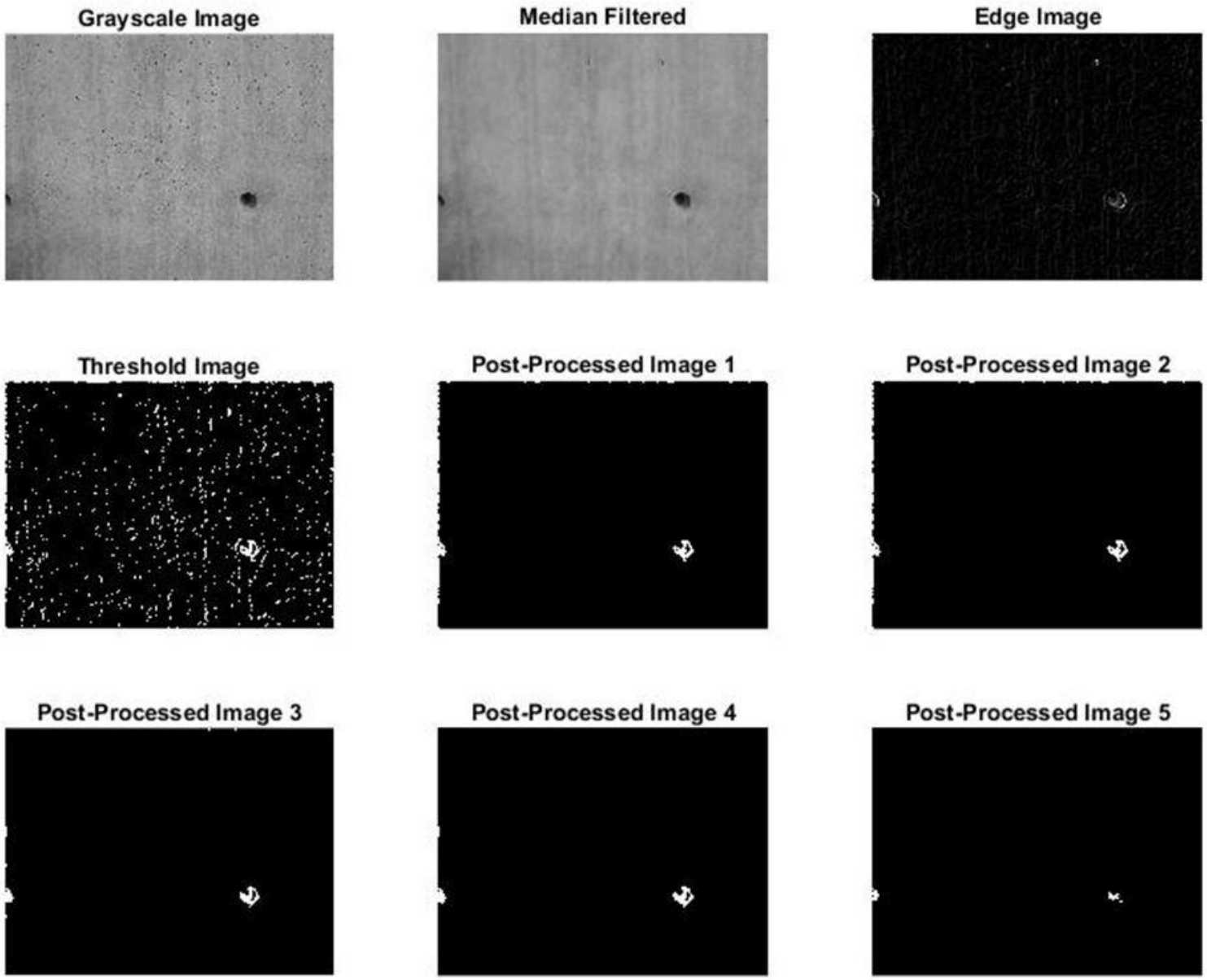


Figure A. 6. Crack detection results using the proposed method on database image #6

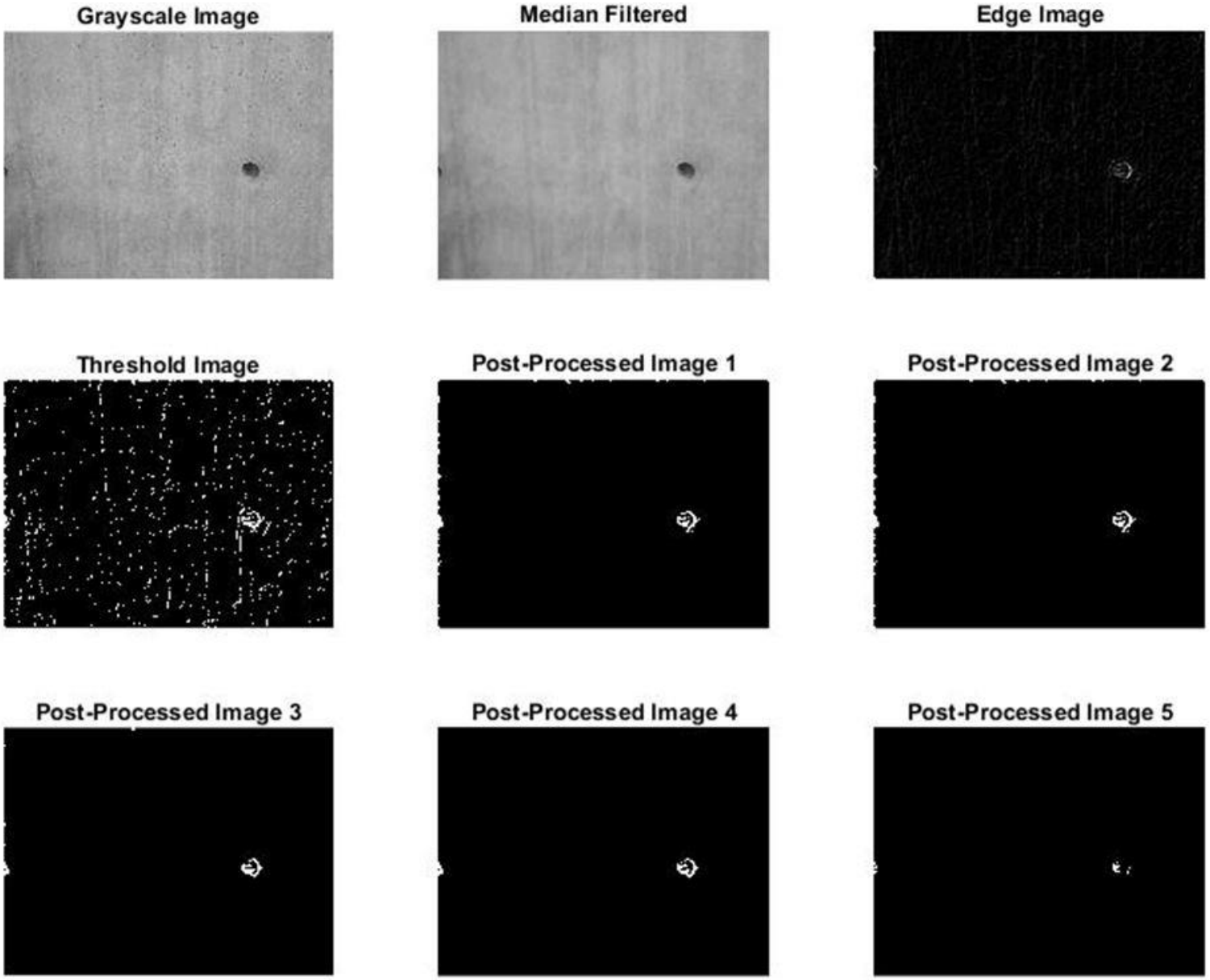


Figure A. 7. Crack detection results using the proposed method on database image #7

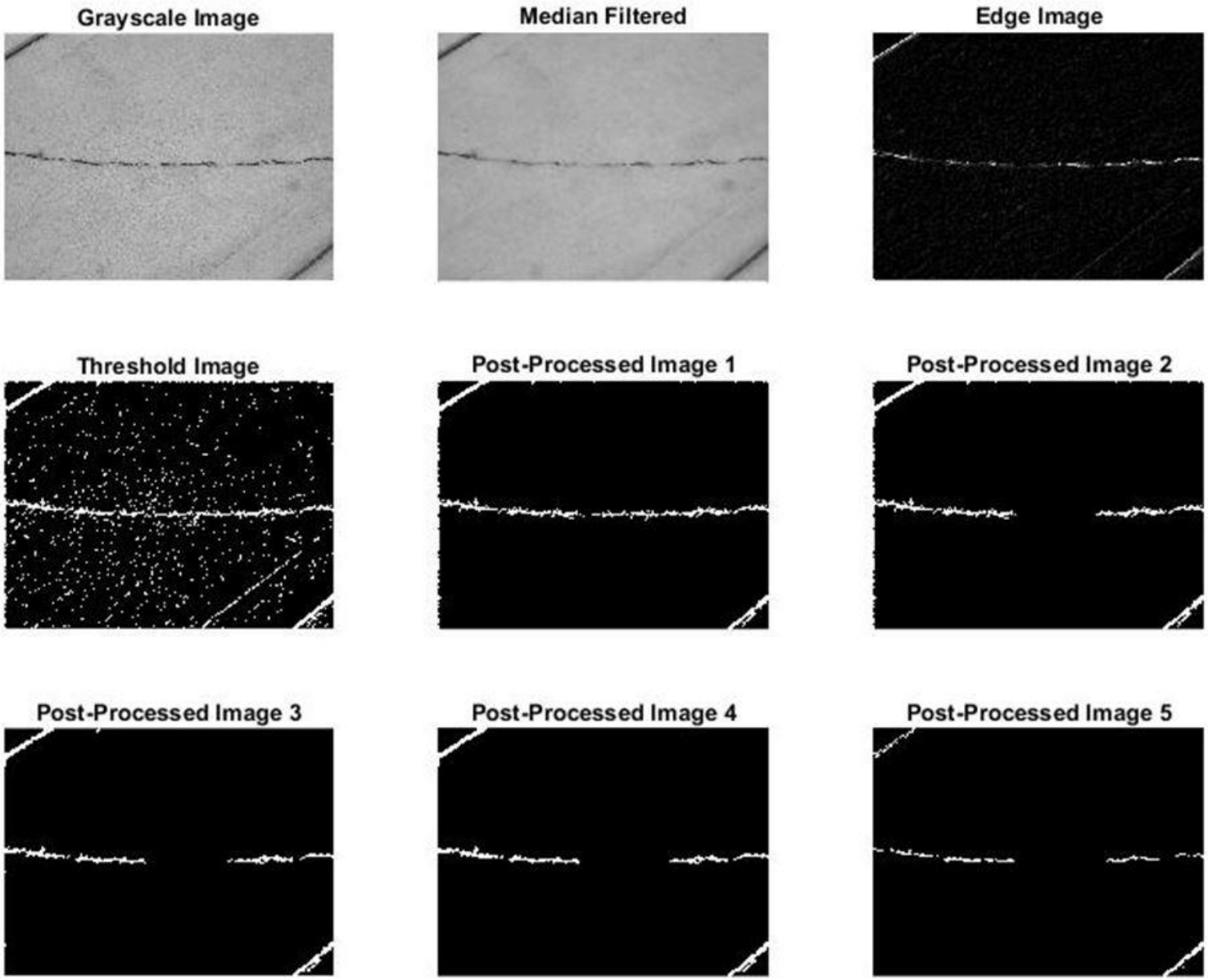


Figure A. 8. Crack detection results using the proposed method on database image #8

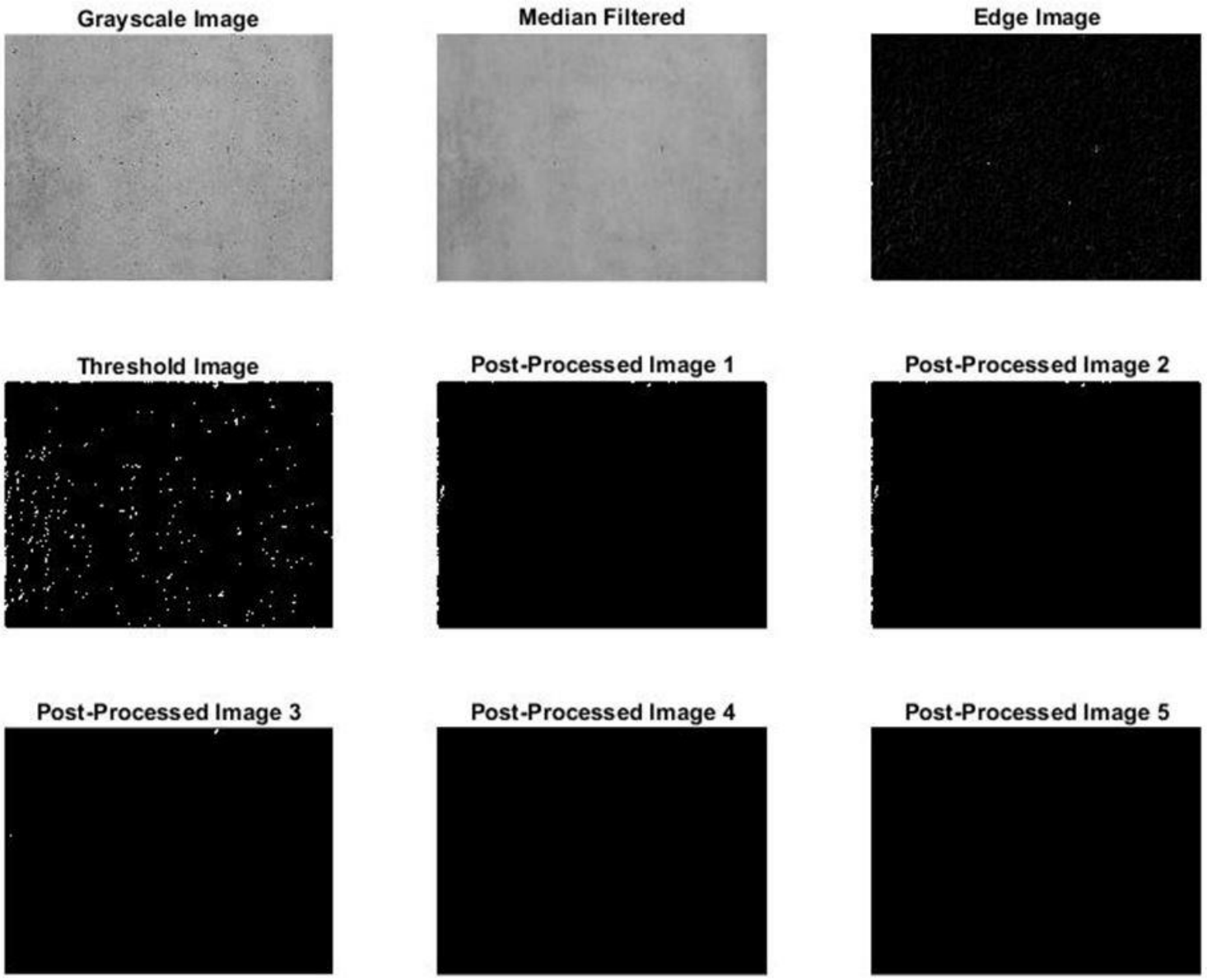


Figure A. 9. Crack detection results using the proposed method on database image #9

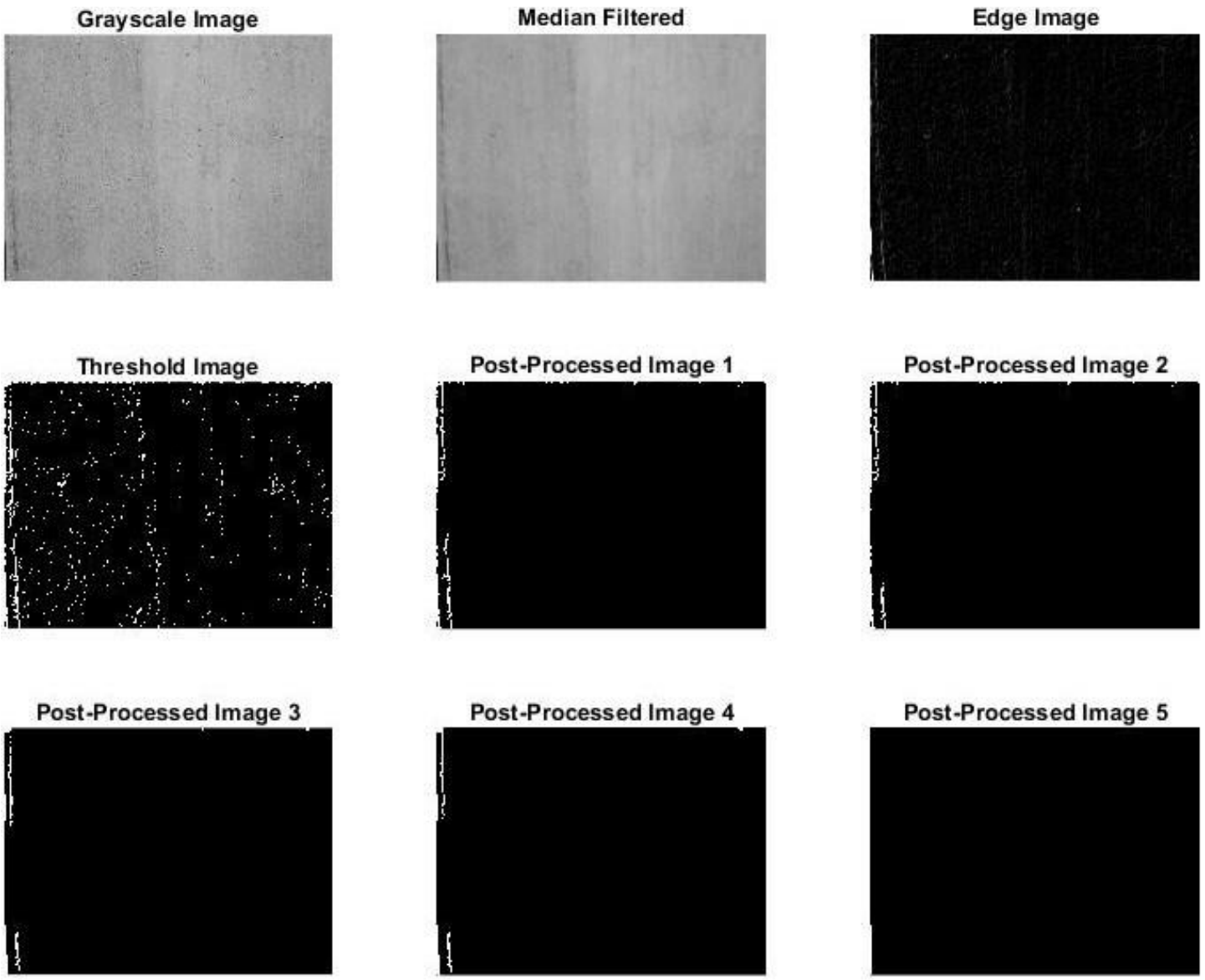


Figure A. 10. Crack detection results using the proposed method on database image #10

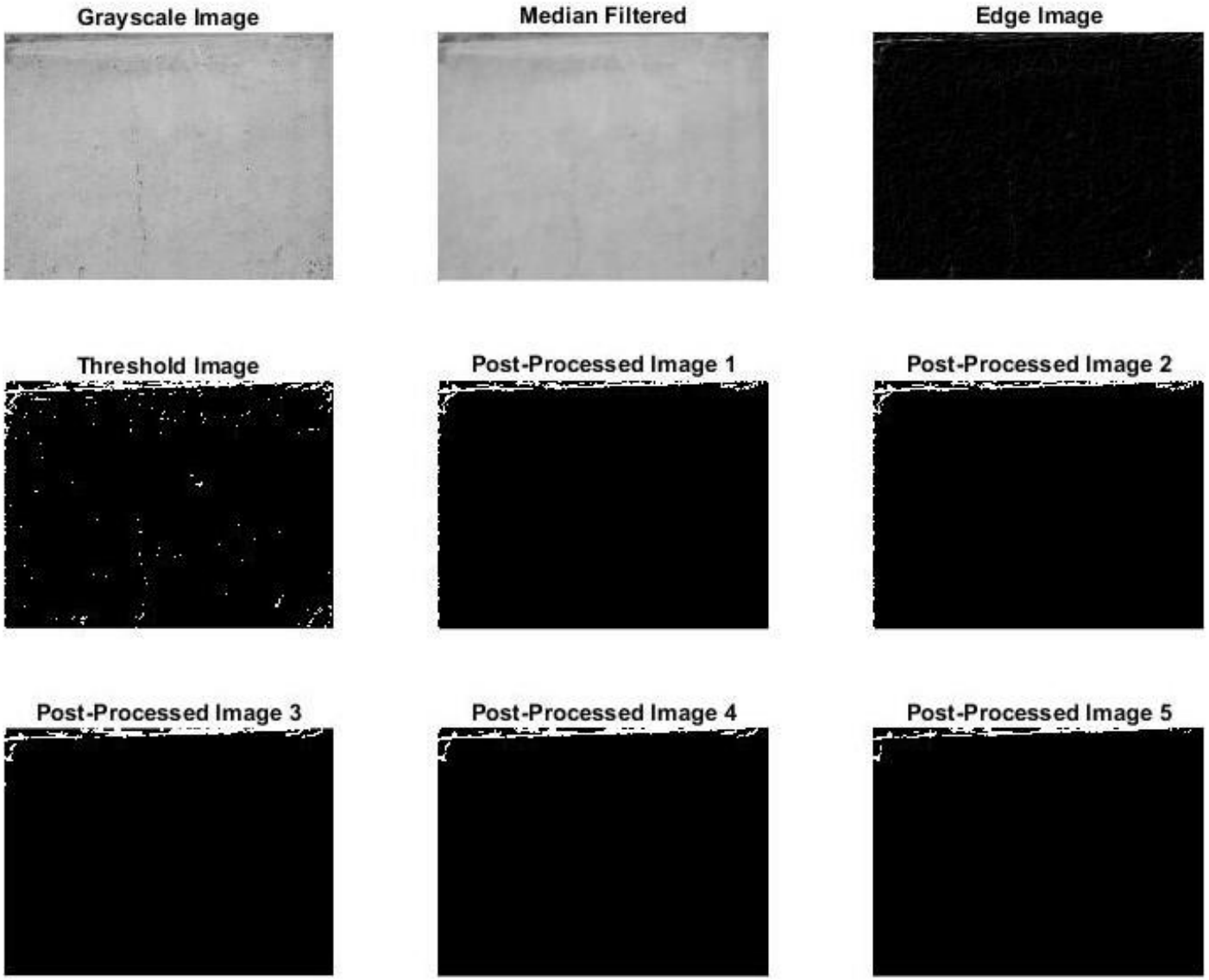


Figure A. 11. Crack detection results using the proposed method on database image #11

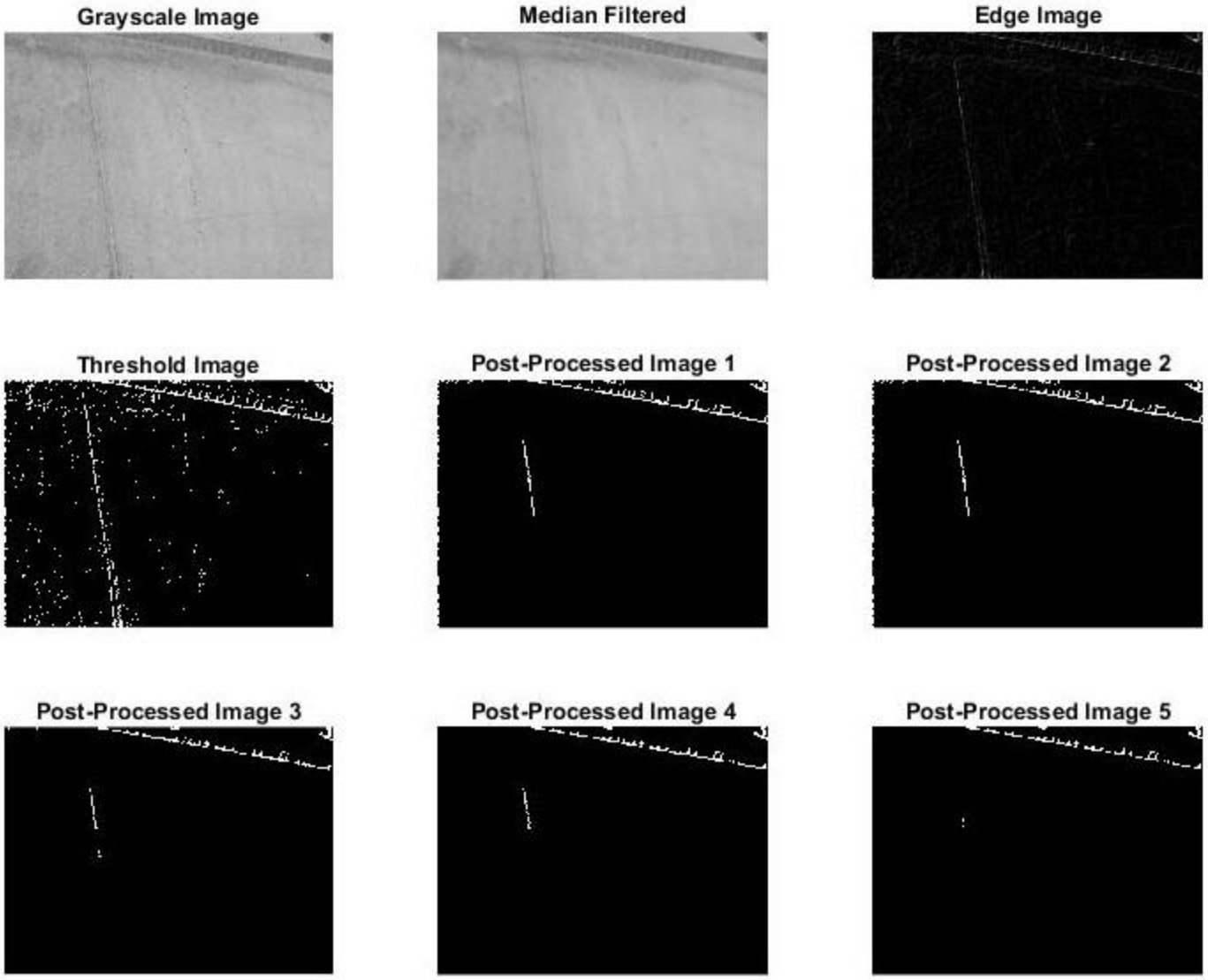


Figure A. 12. Crack detection results using the proposed method on database image #12

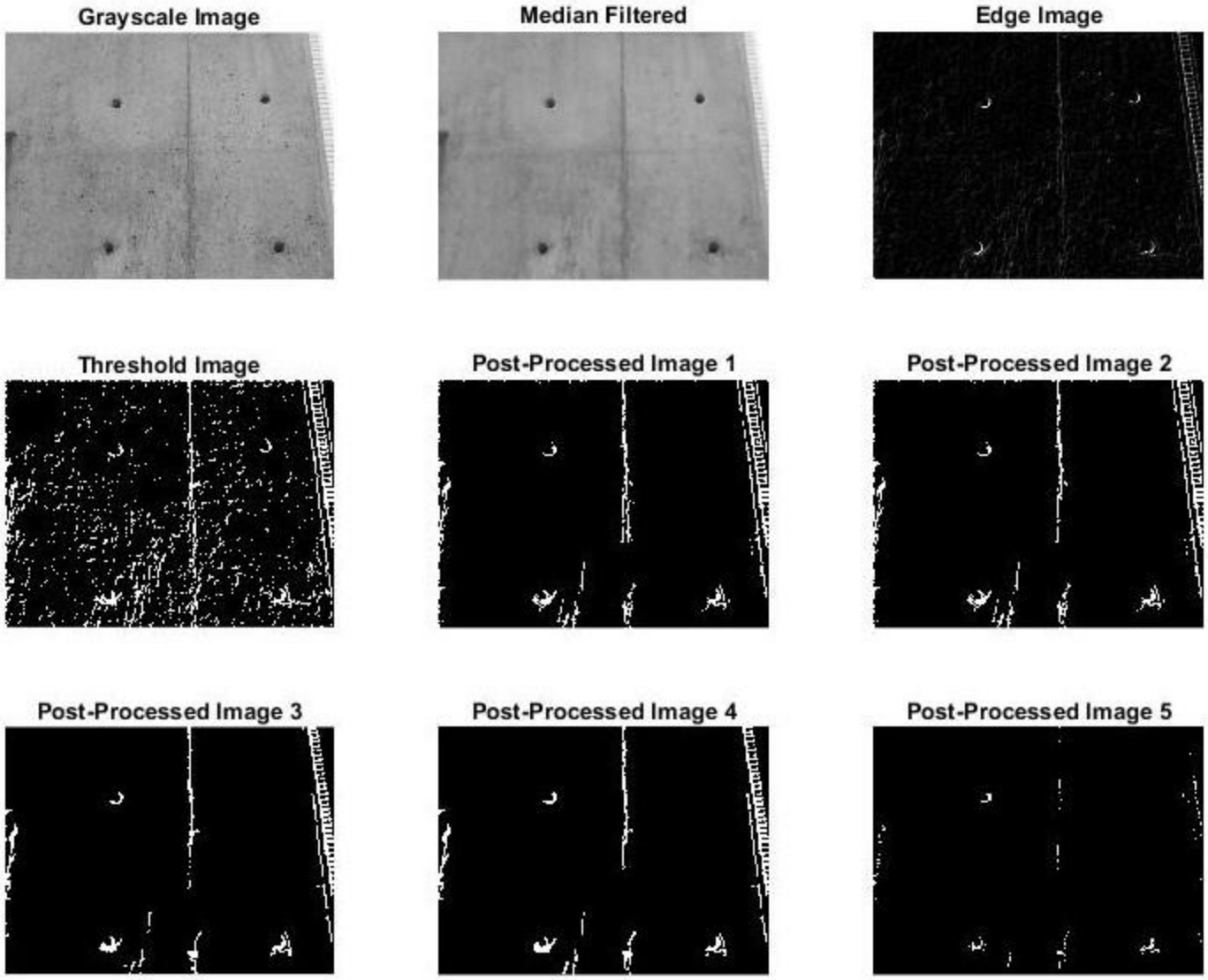


Figure A. 13. Crack detection results using the proposed method on database image #13

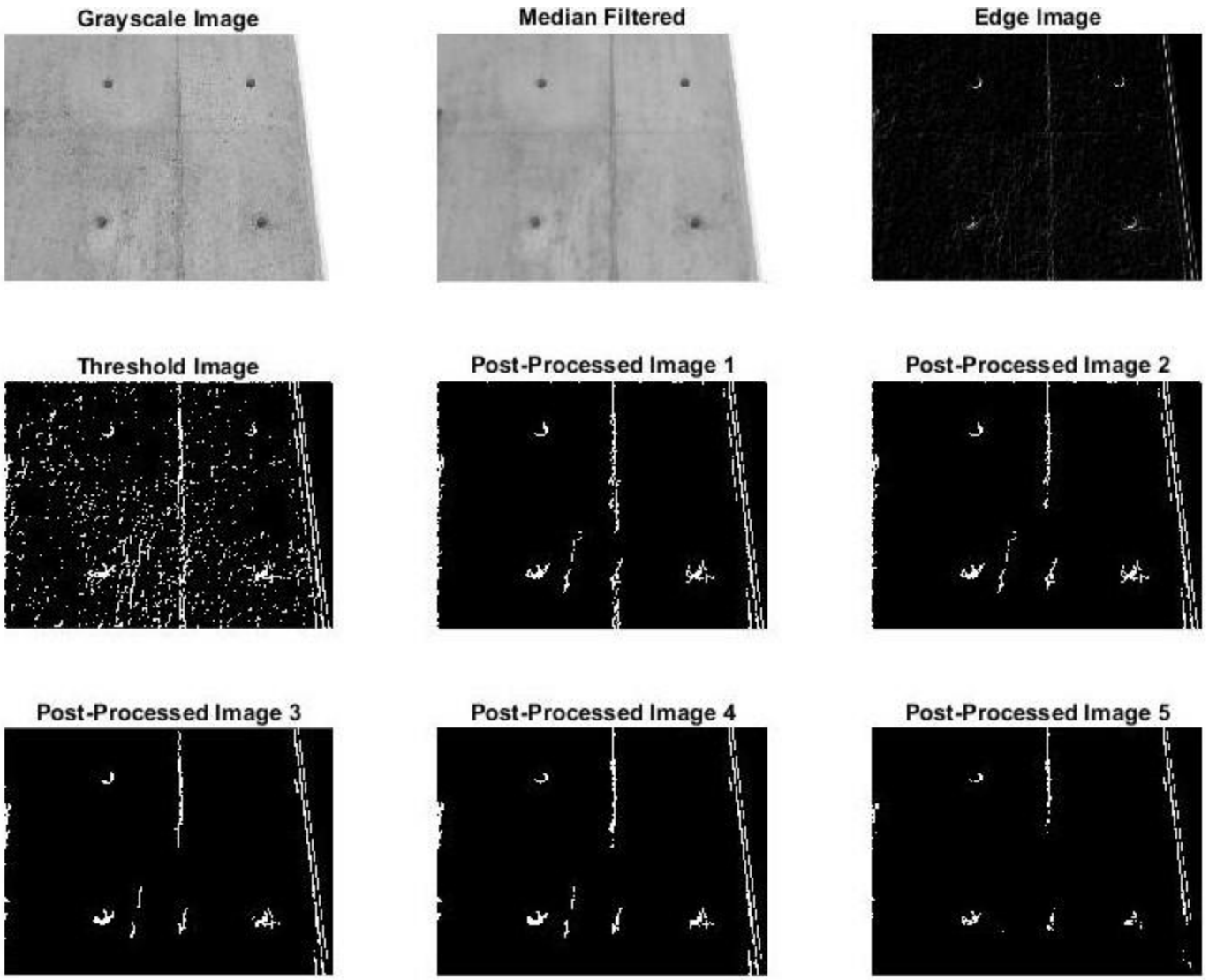


Figure A. 14. Crack detection results using the proposed method on database image #14

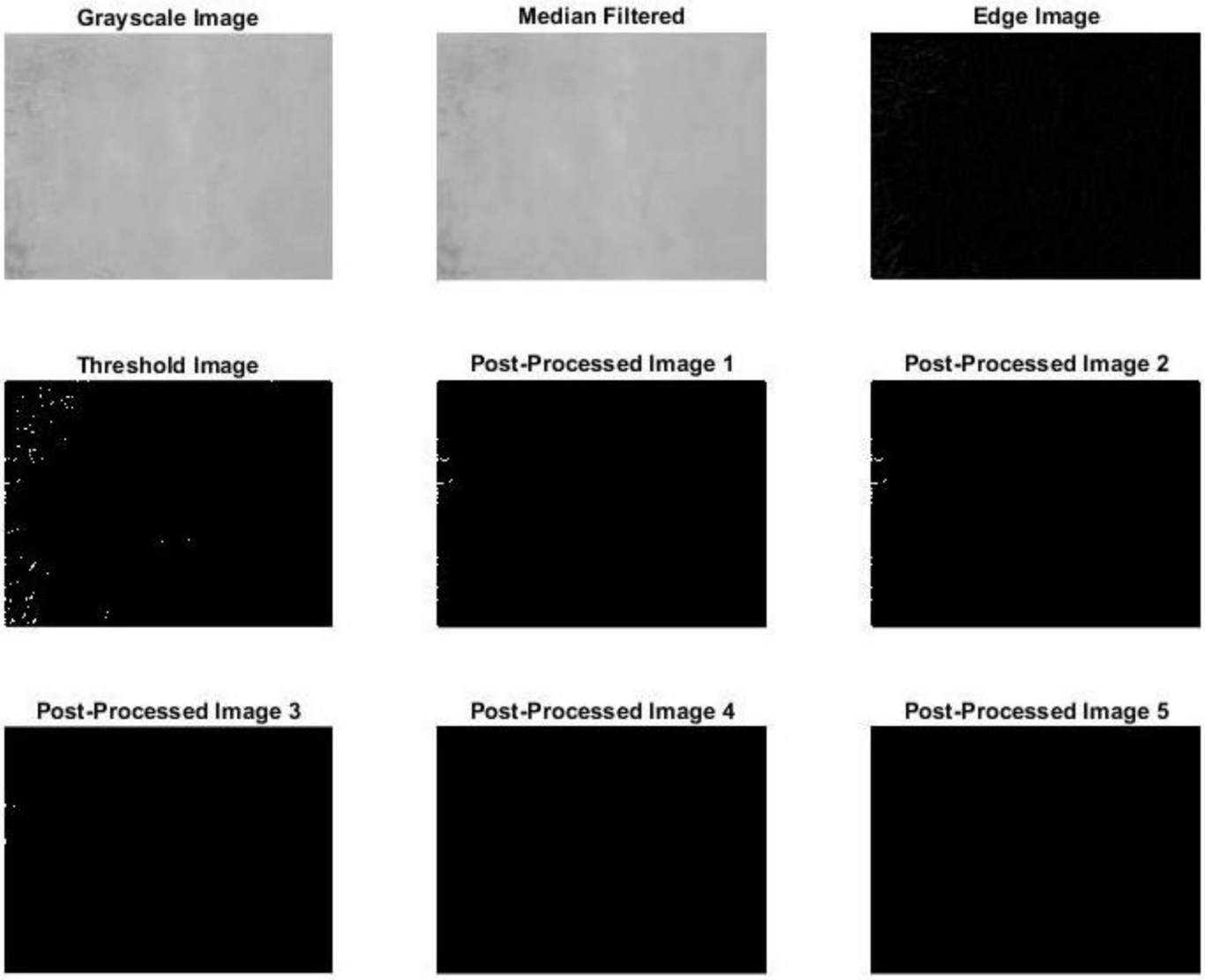


Figure A. 15. Crack detection results using the proposed method on database image #15

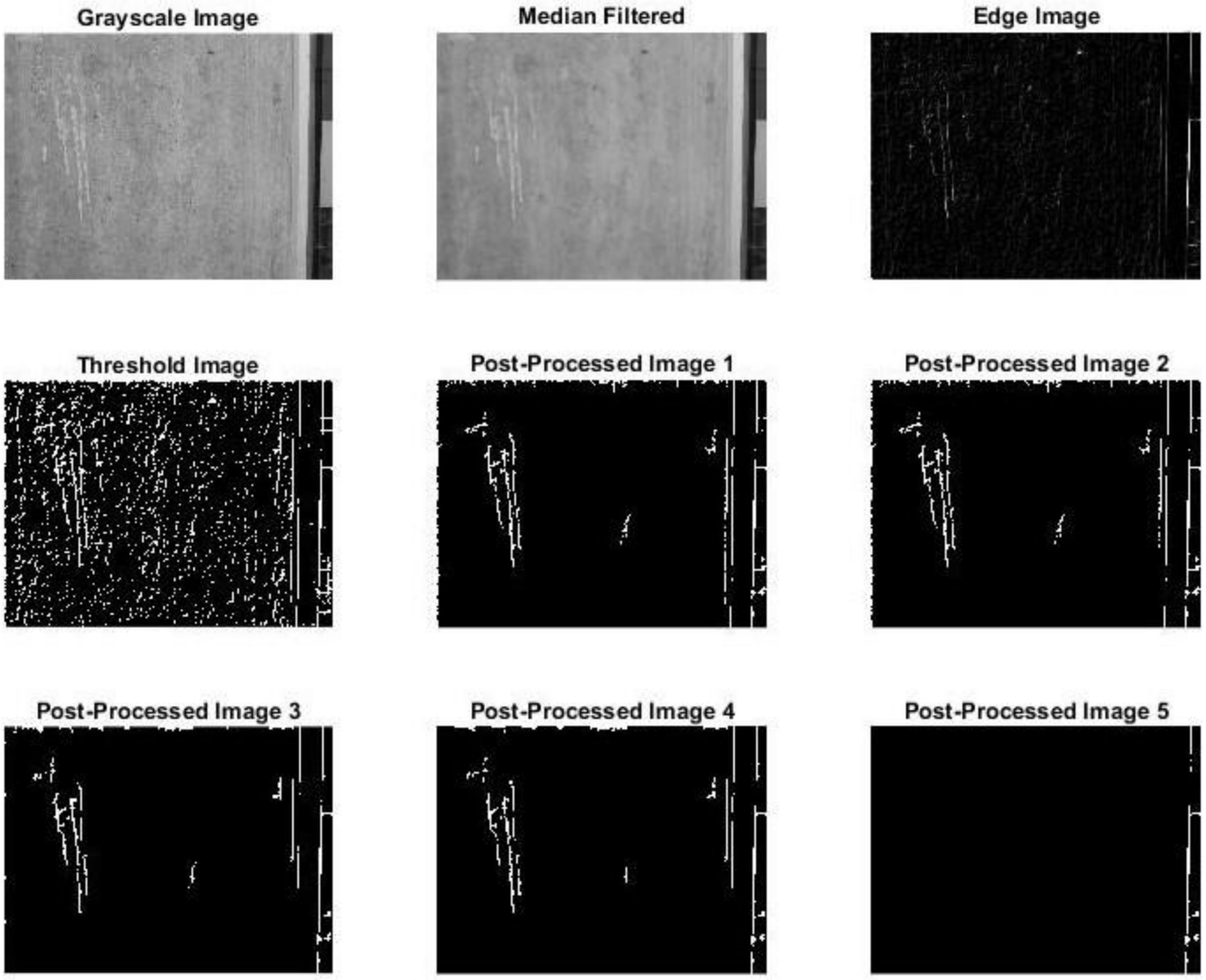


Figure A. 16. Crack detection results using the proposed method on database image #16

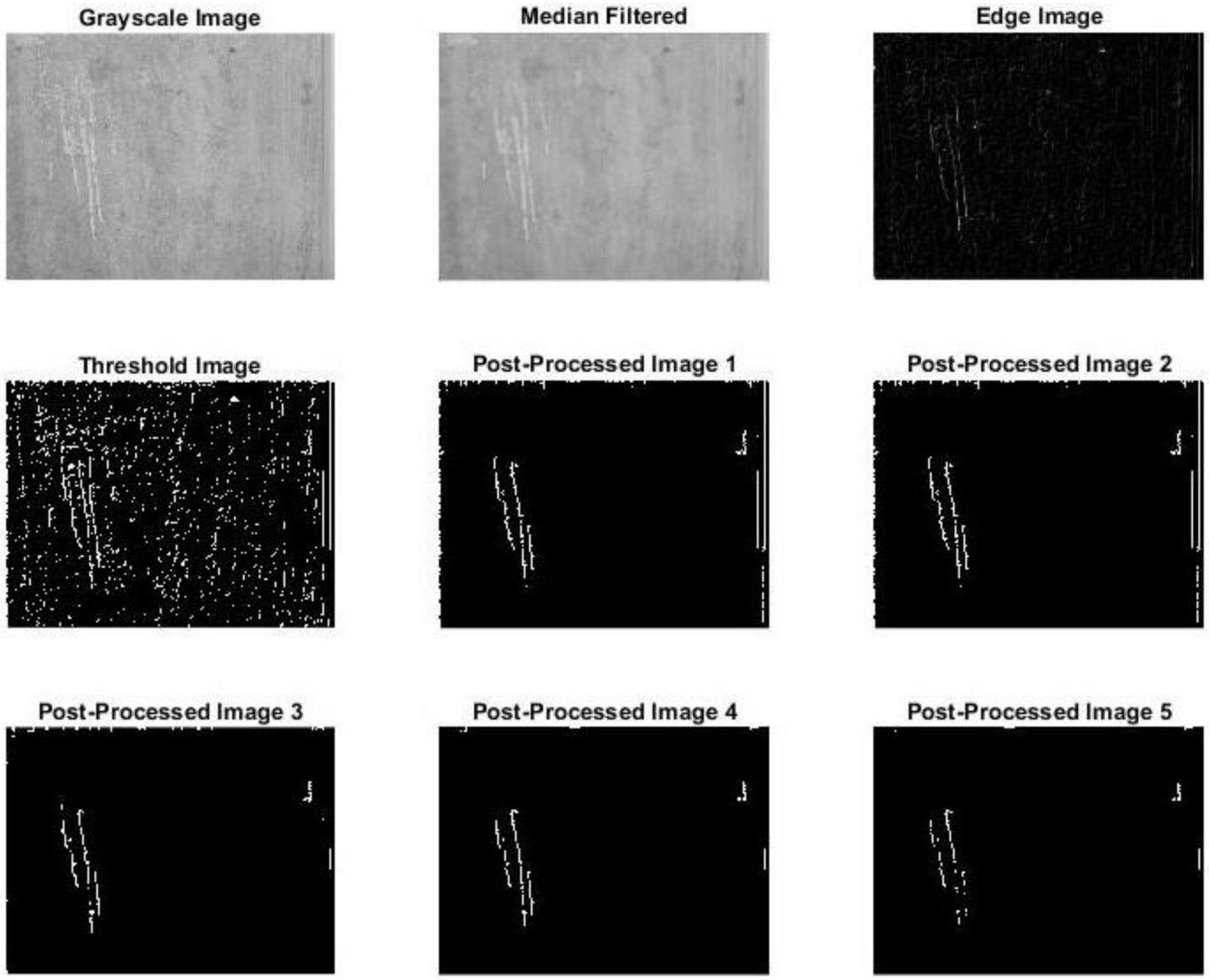


Figure A. 17. Crack detection results using the proposed method on database image #17

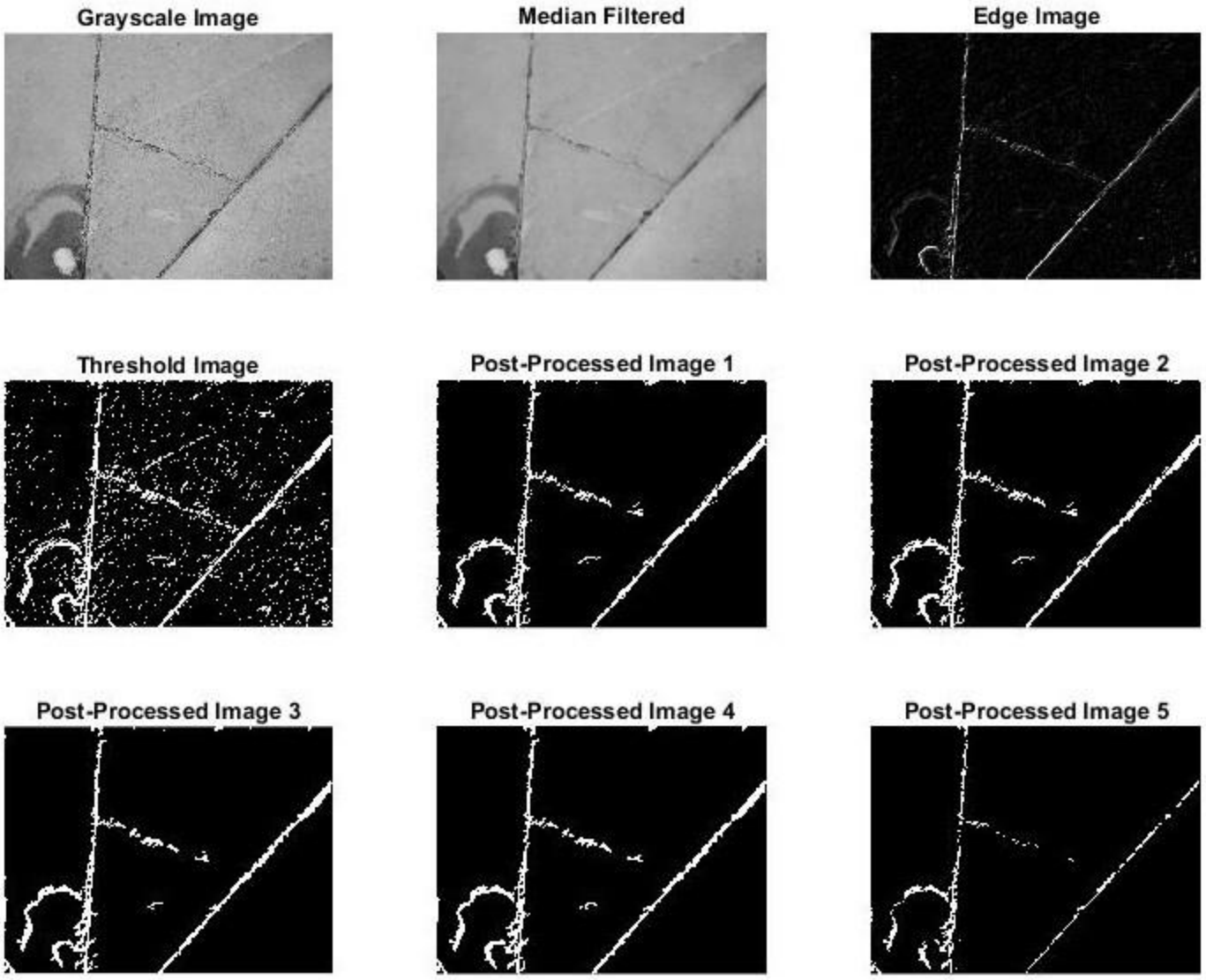


Figure A. 18. Crack detection results using the proposed method on database image #18

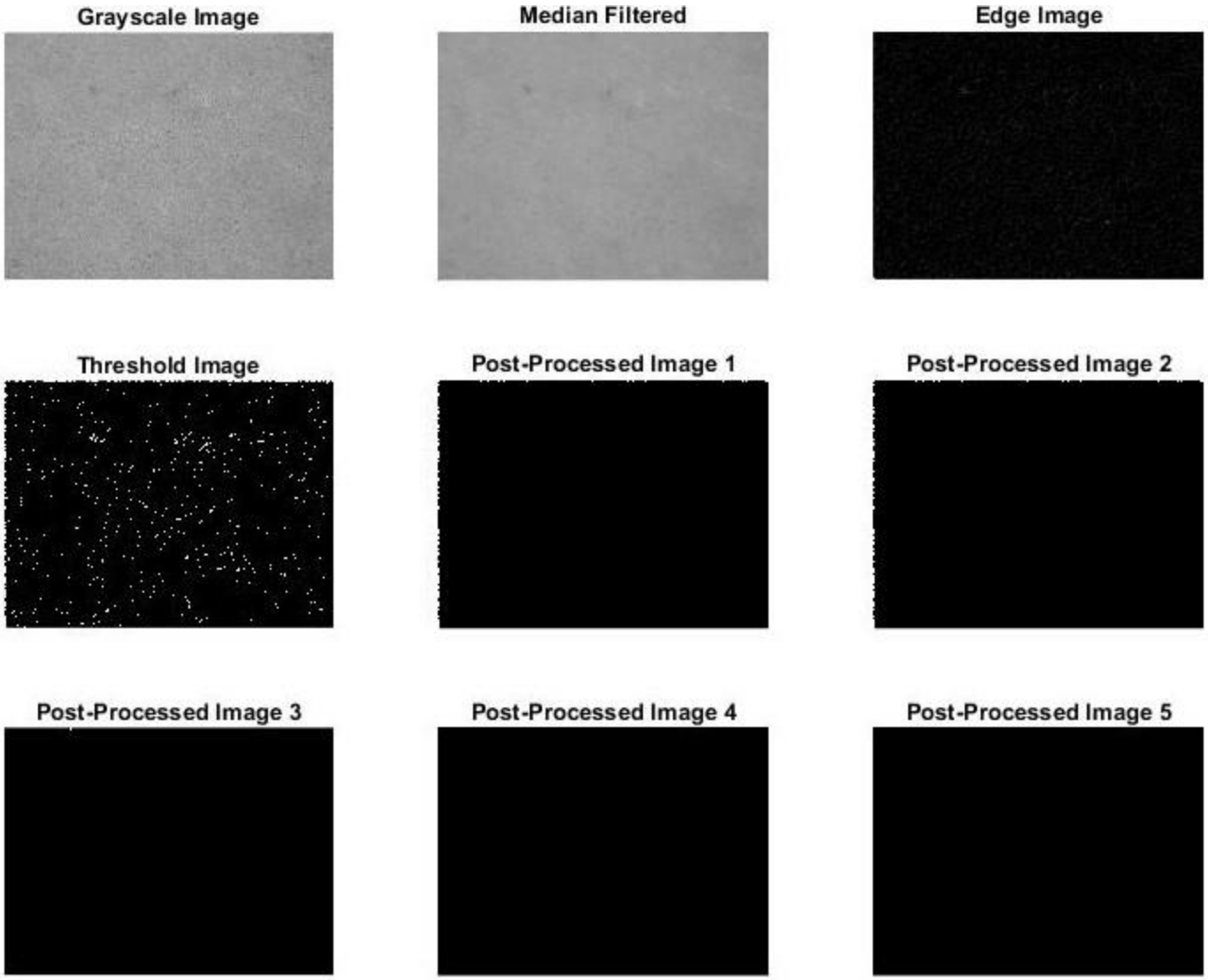


Figure A. 19. Crack detection results using the proposed method on database image #19

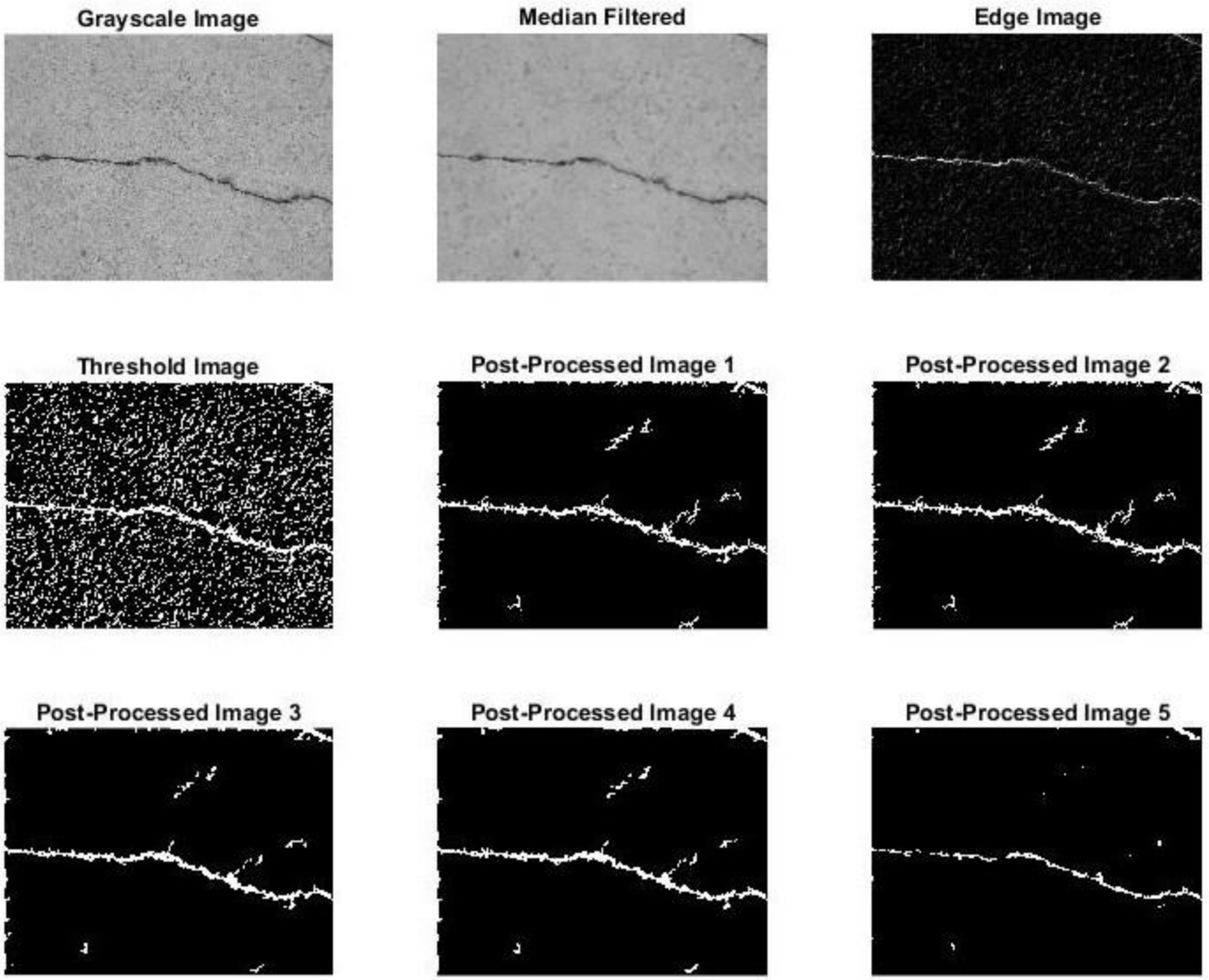


Figure A. 20. Crack detection results using the proposed method on database image #20

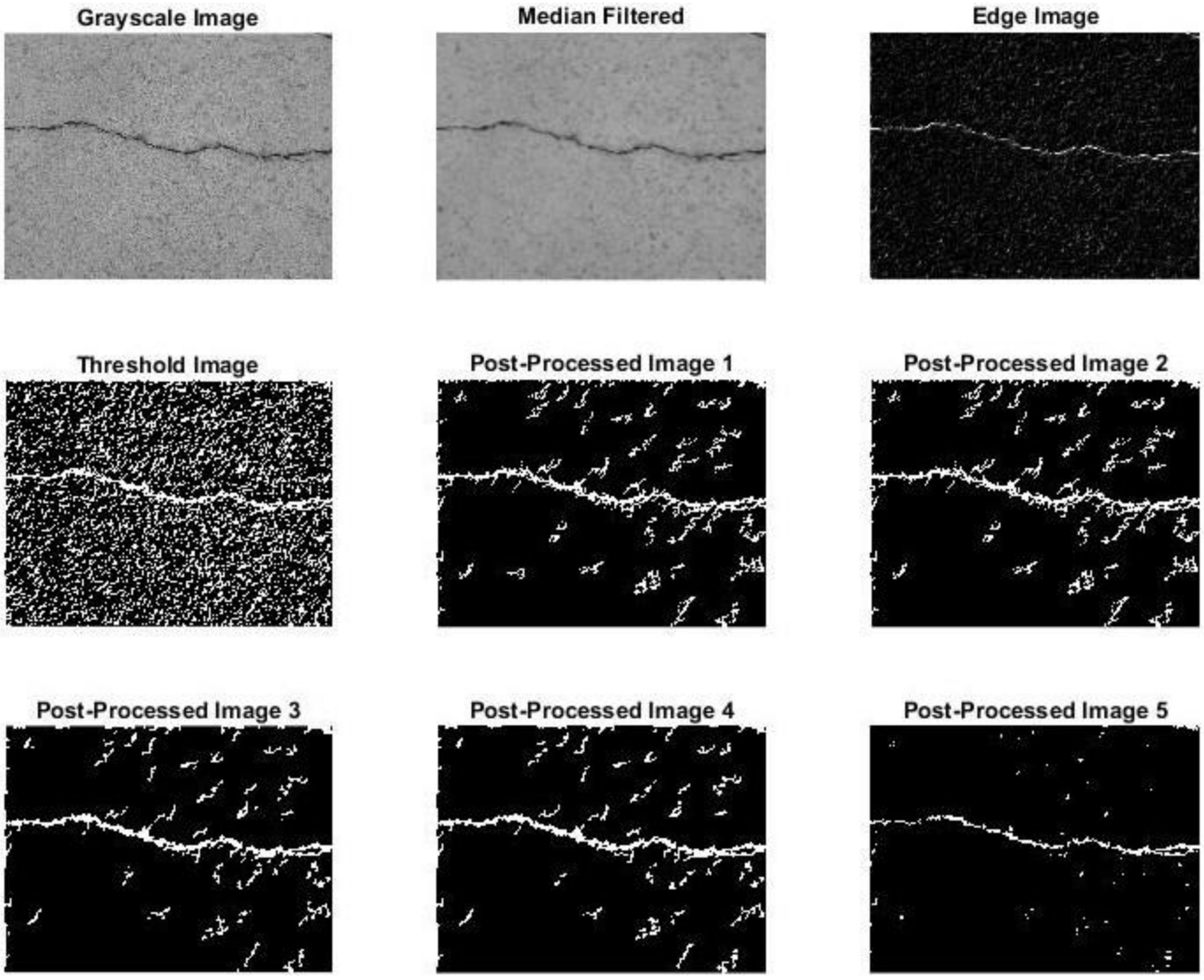


Figure A. 21. Crack detection results using the proposed method on database image #21

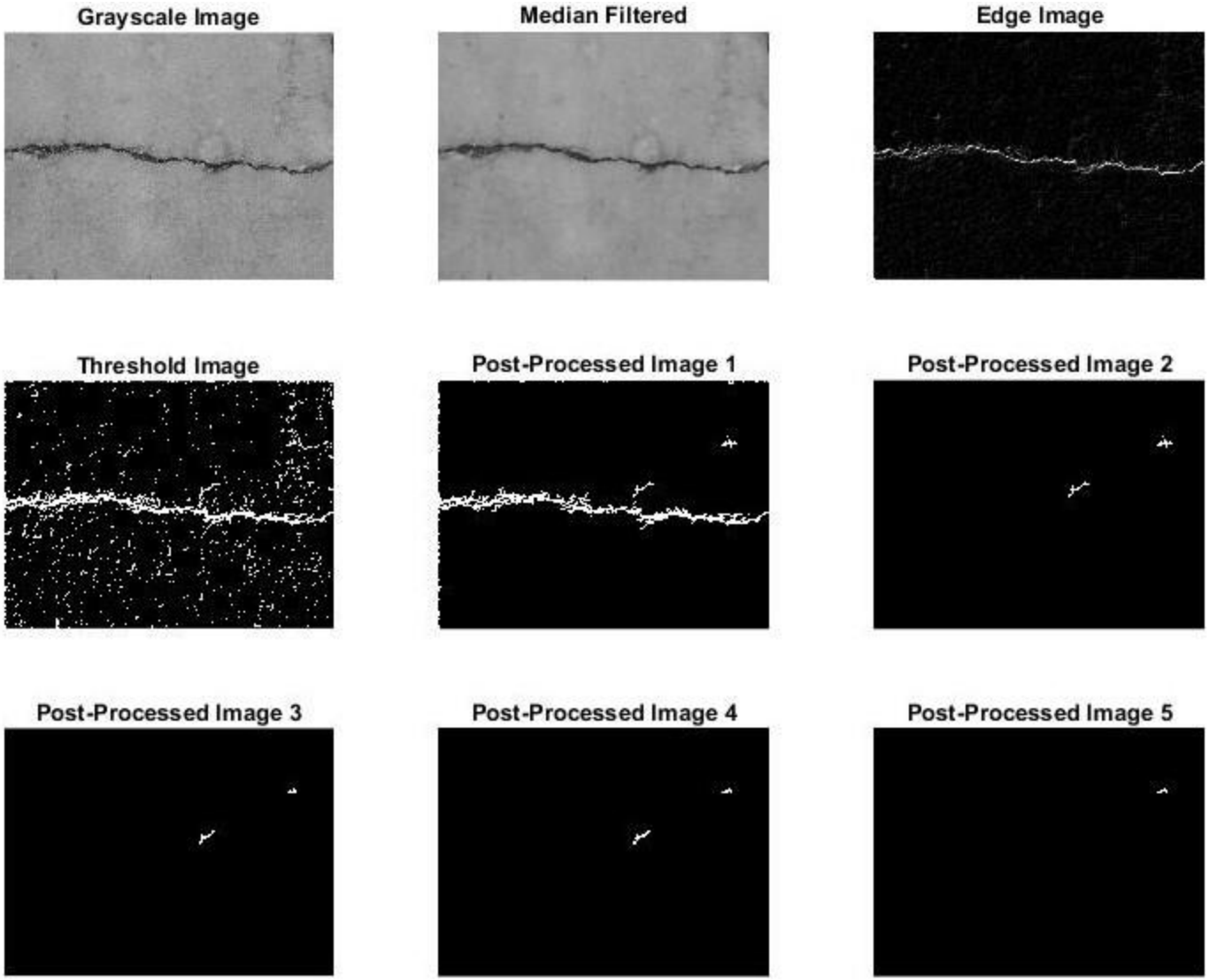


Figure A. 22. Crack detection results using the proposed method on database image #22

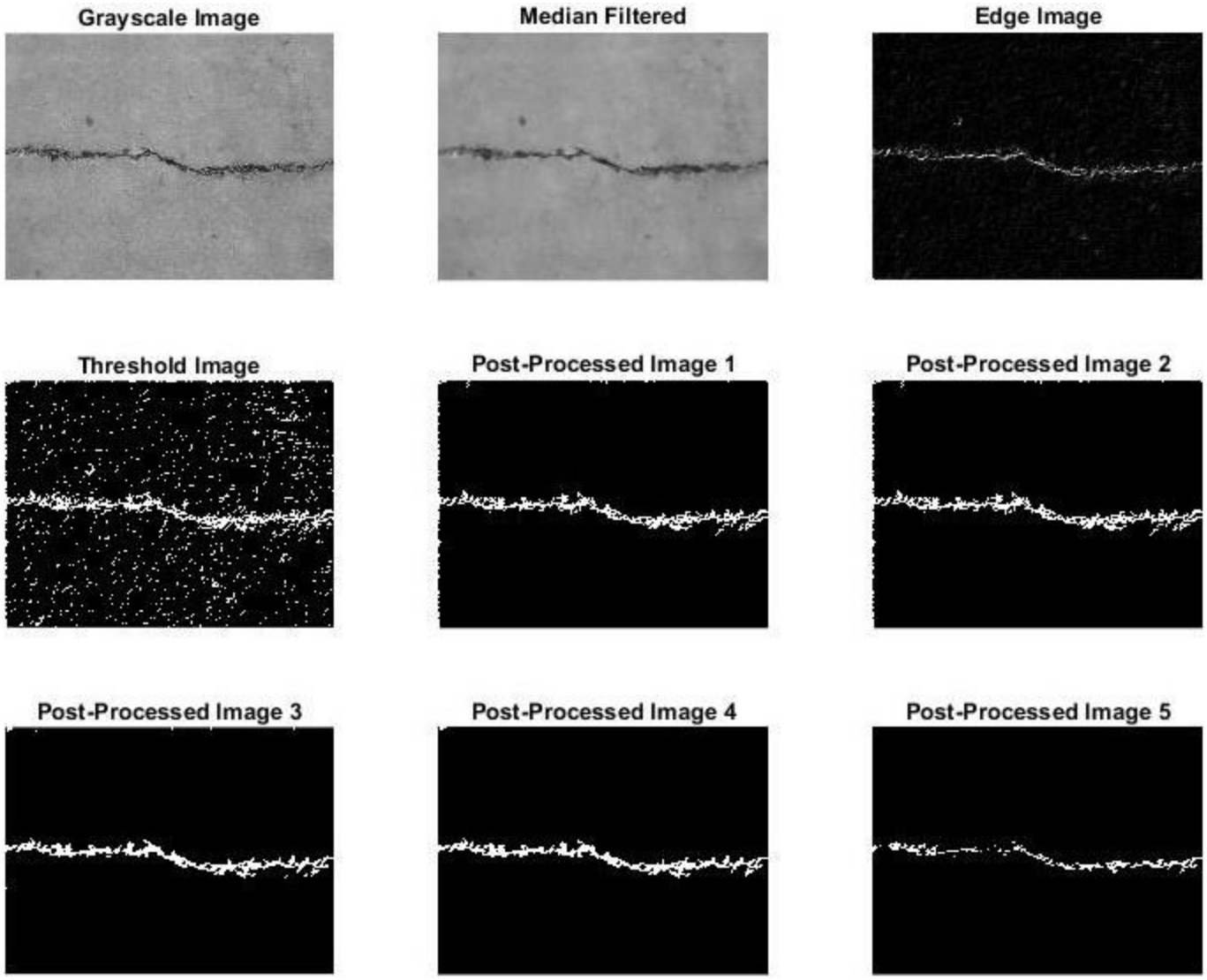


Figure A. 23. Crack detection results using the proposed method on database image #23

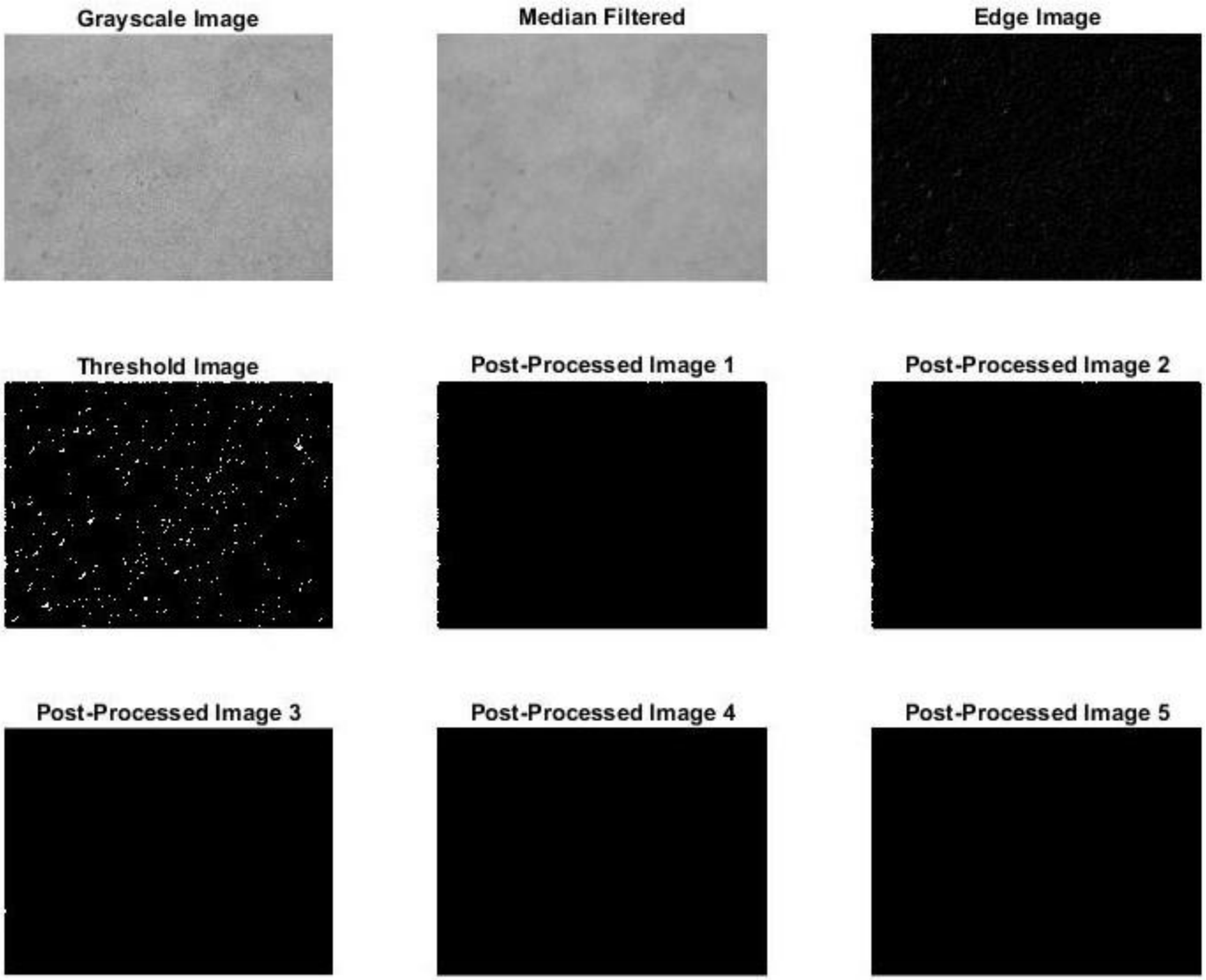


Figure A. 24. Crack detection results using the proposed method on database image #24

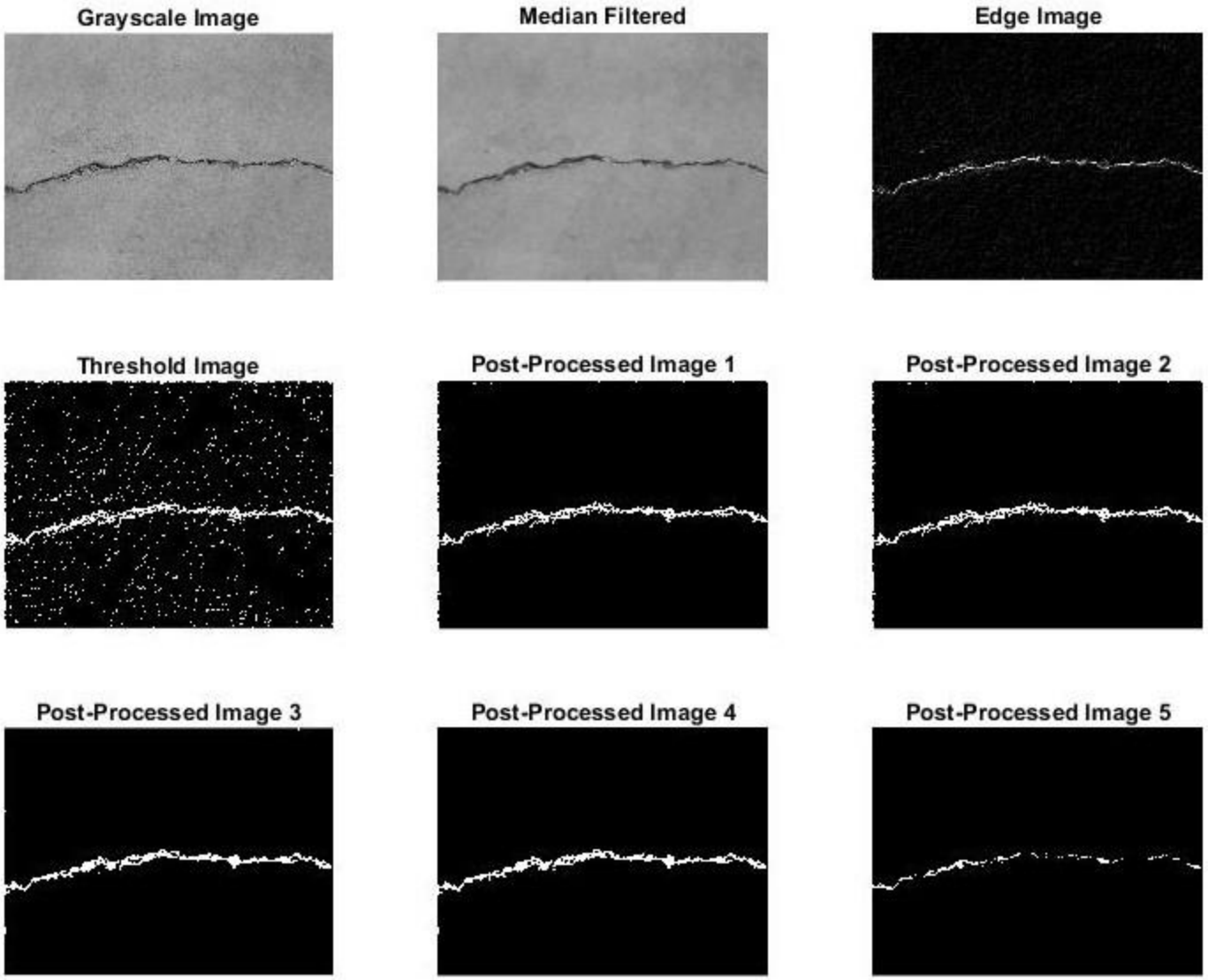


Figure A. 25. Crack detection results using the proposed method on database image #25

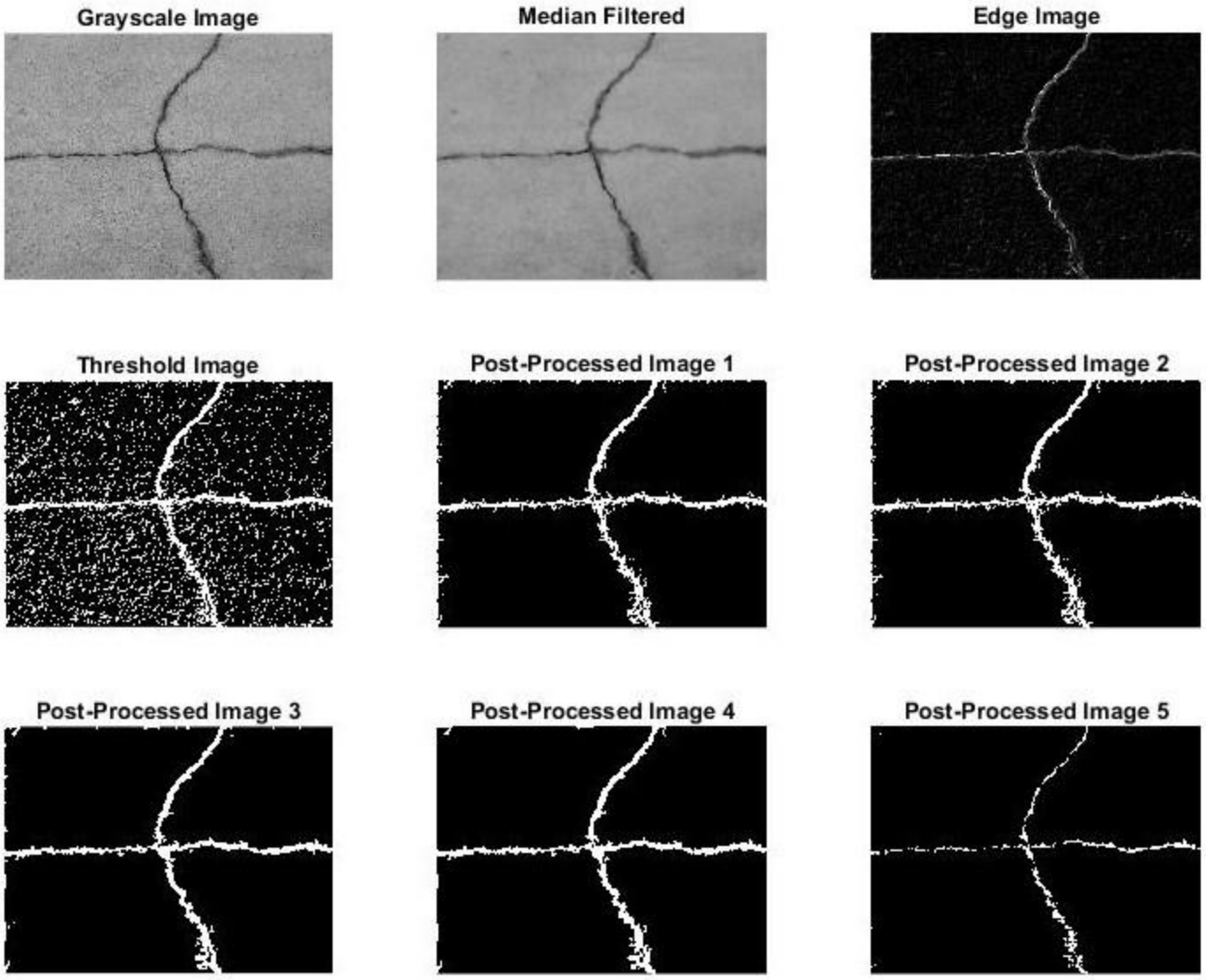


Figure A. 26. Crack detection results using the proposed method on database image #26

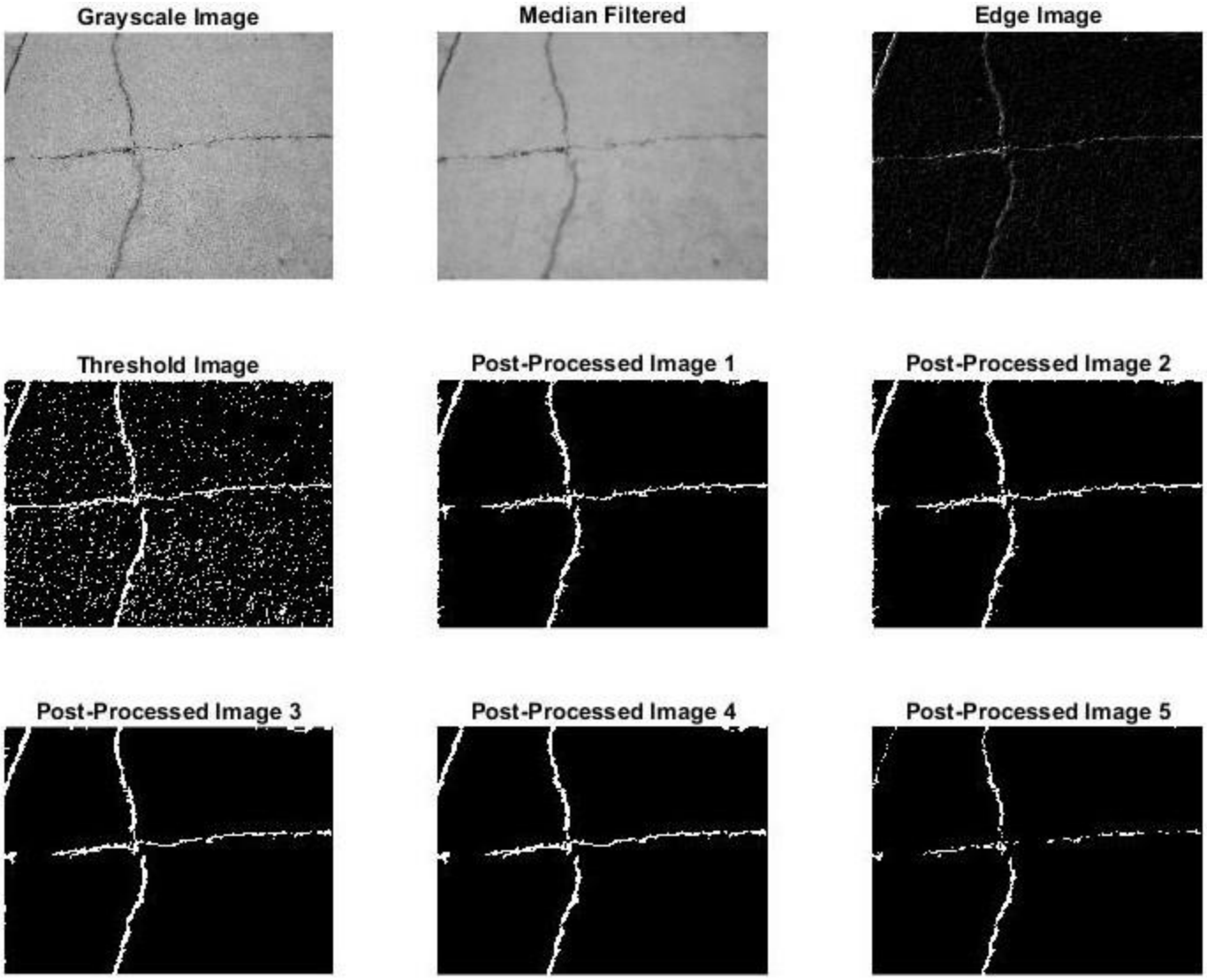


Figure A. 27. Crack detection results using the proposed method on database image #27

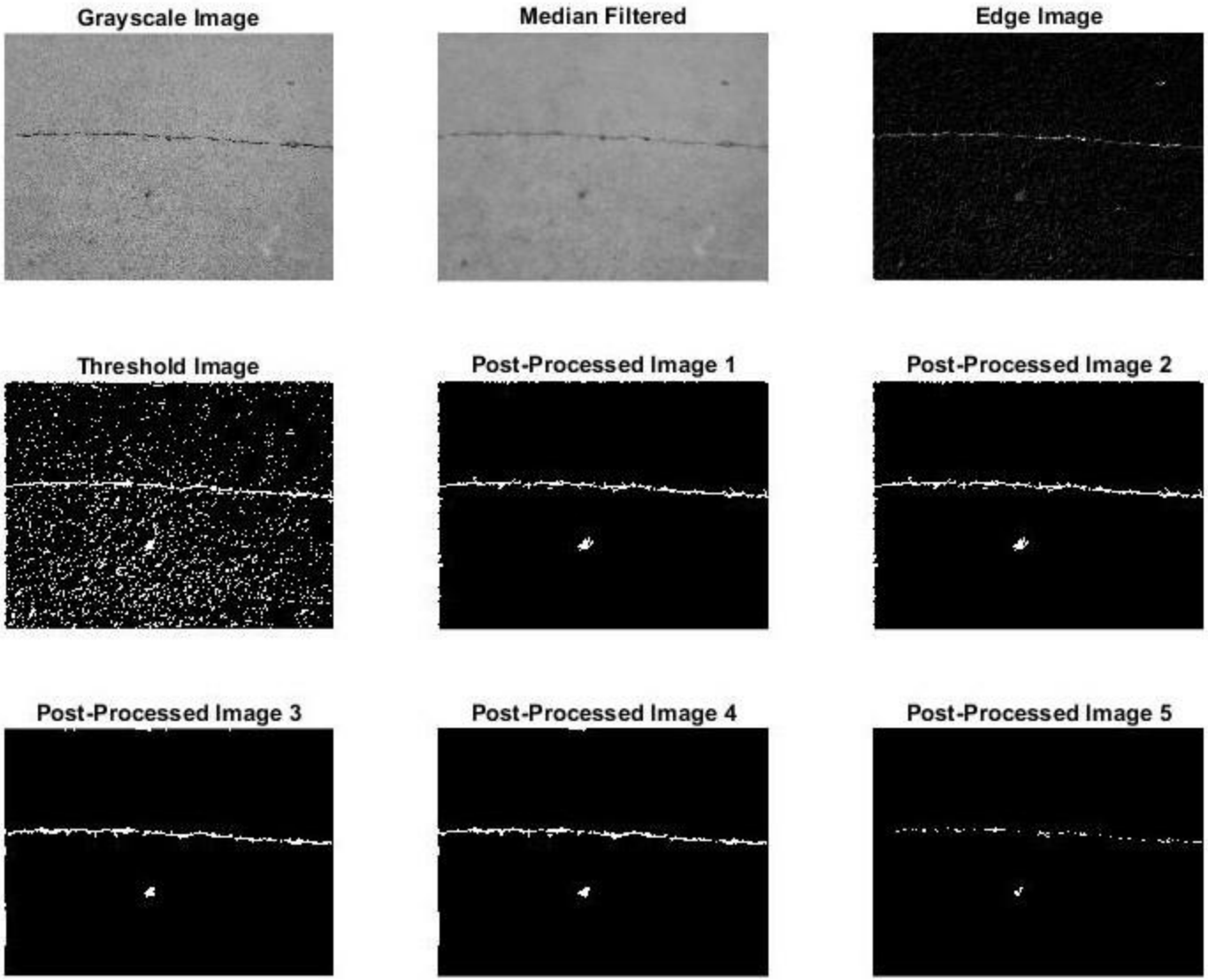


Figure A. 28. Crack detection results using the proposed method on database image #28

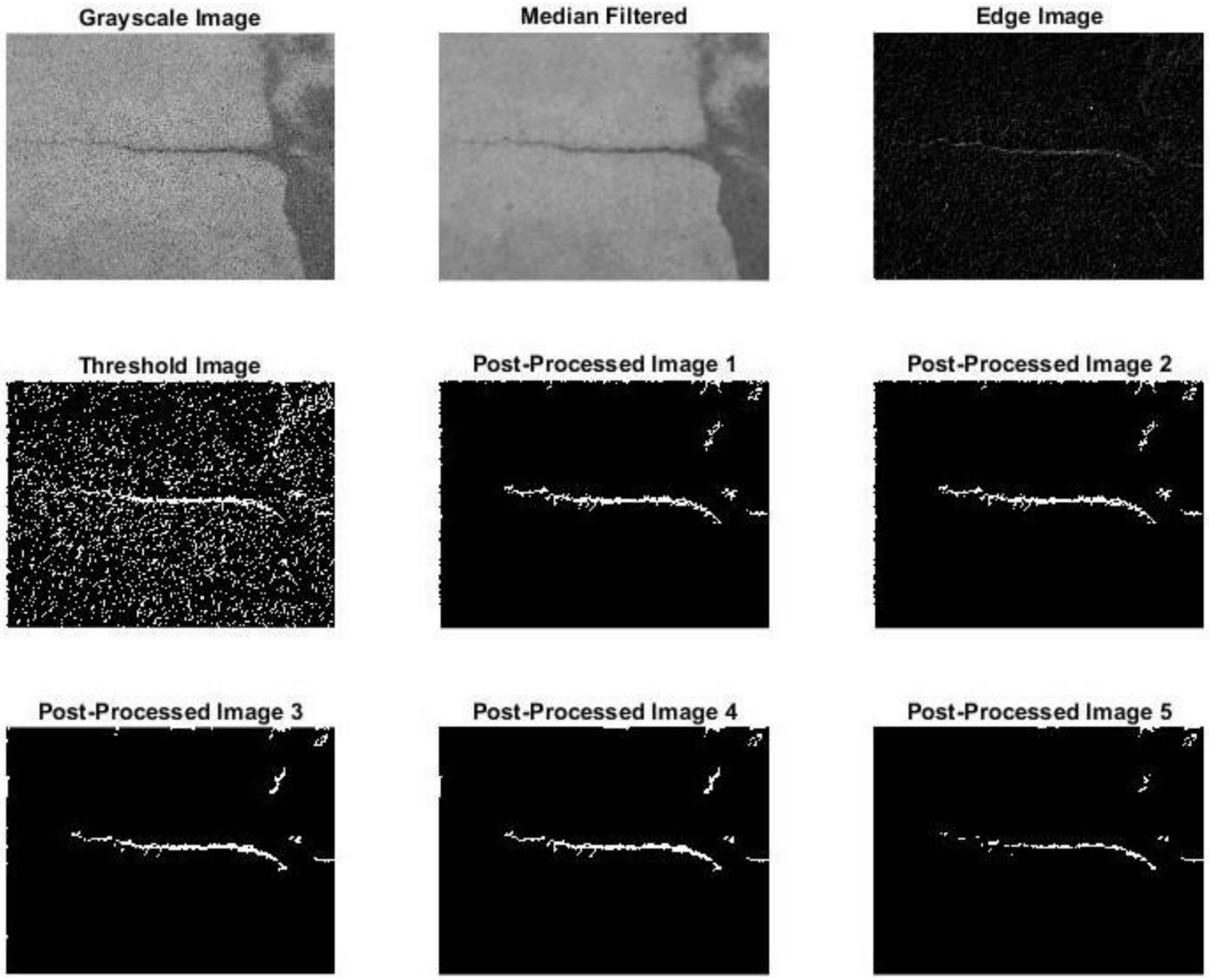


Figure A. 29. Crack detection results using the proposed method on database image #29

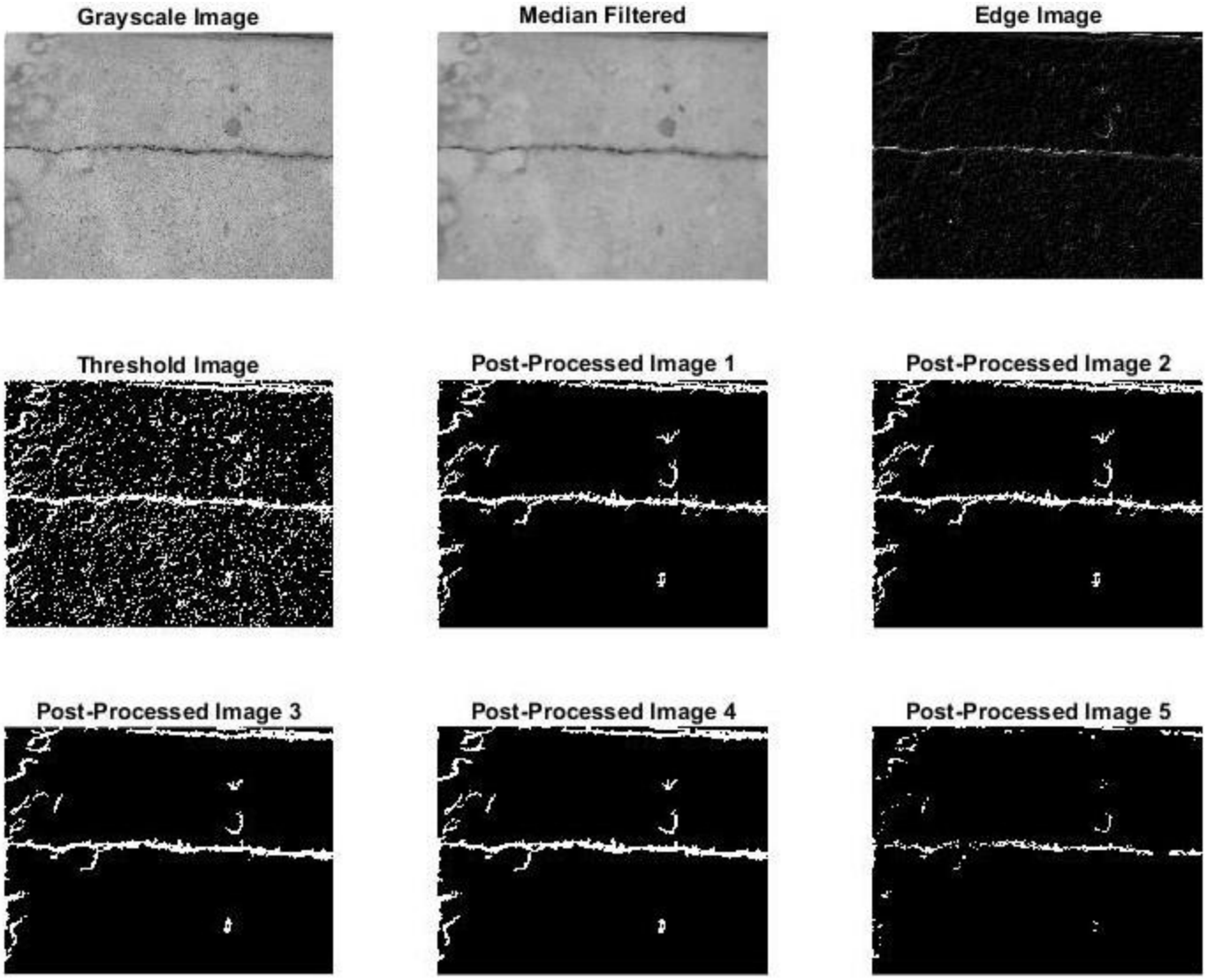


Figure A. 30. Crack detection results using the proposed method on database image #30

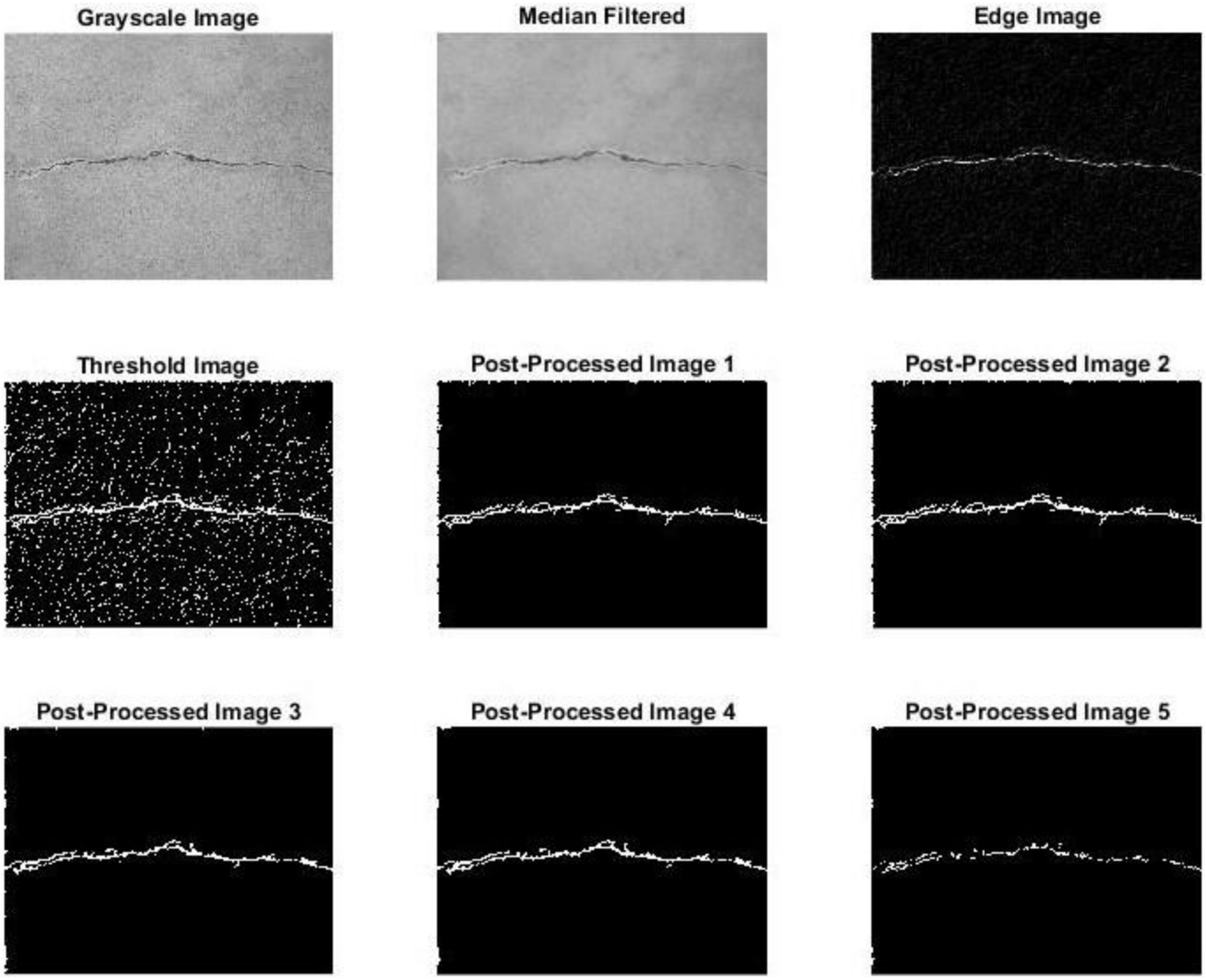


Figure A. 31. Crack detection results using the proposed method on database image #31

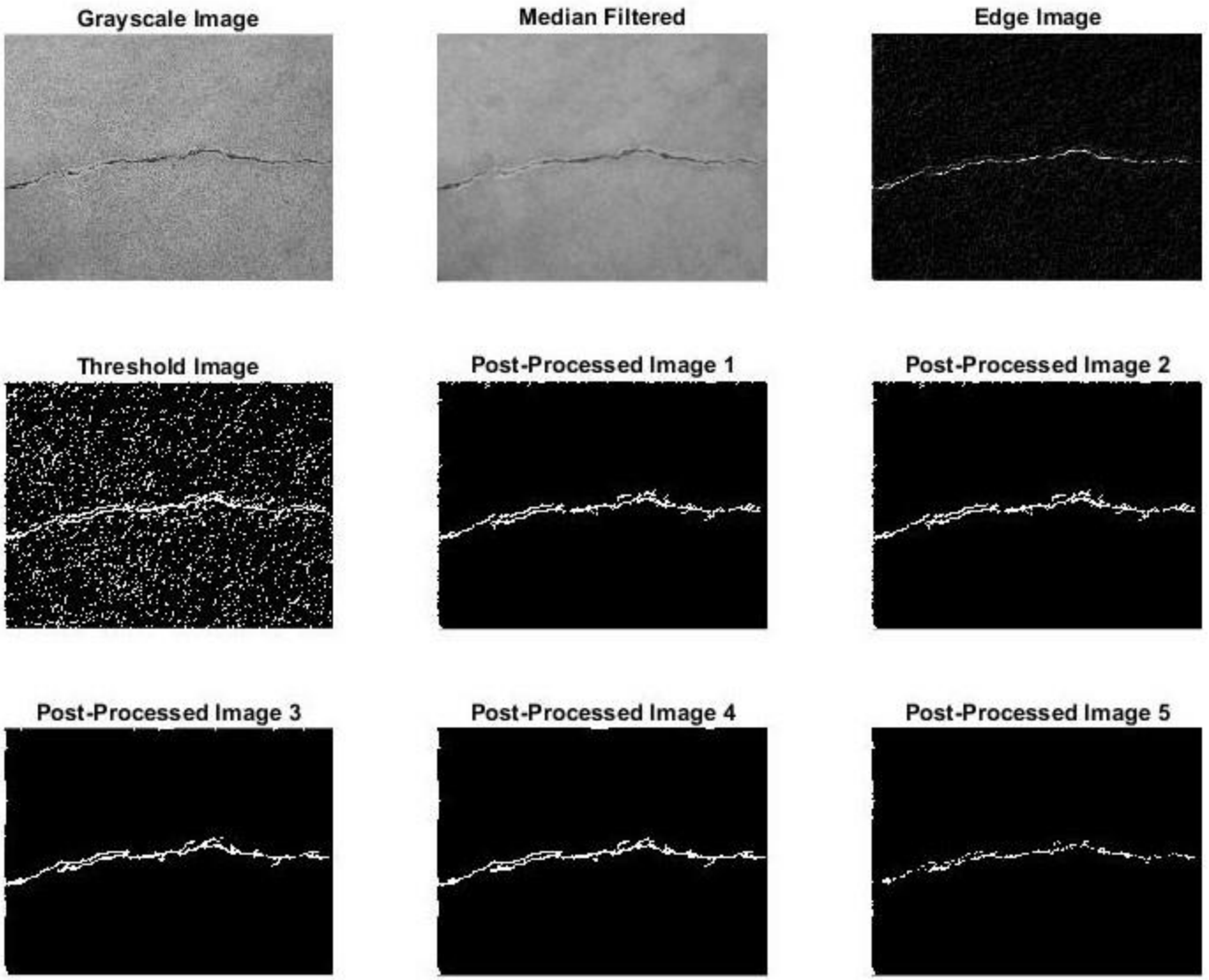


Figure A. 32. Crack detection results using the proposed method on database image #32

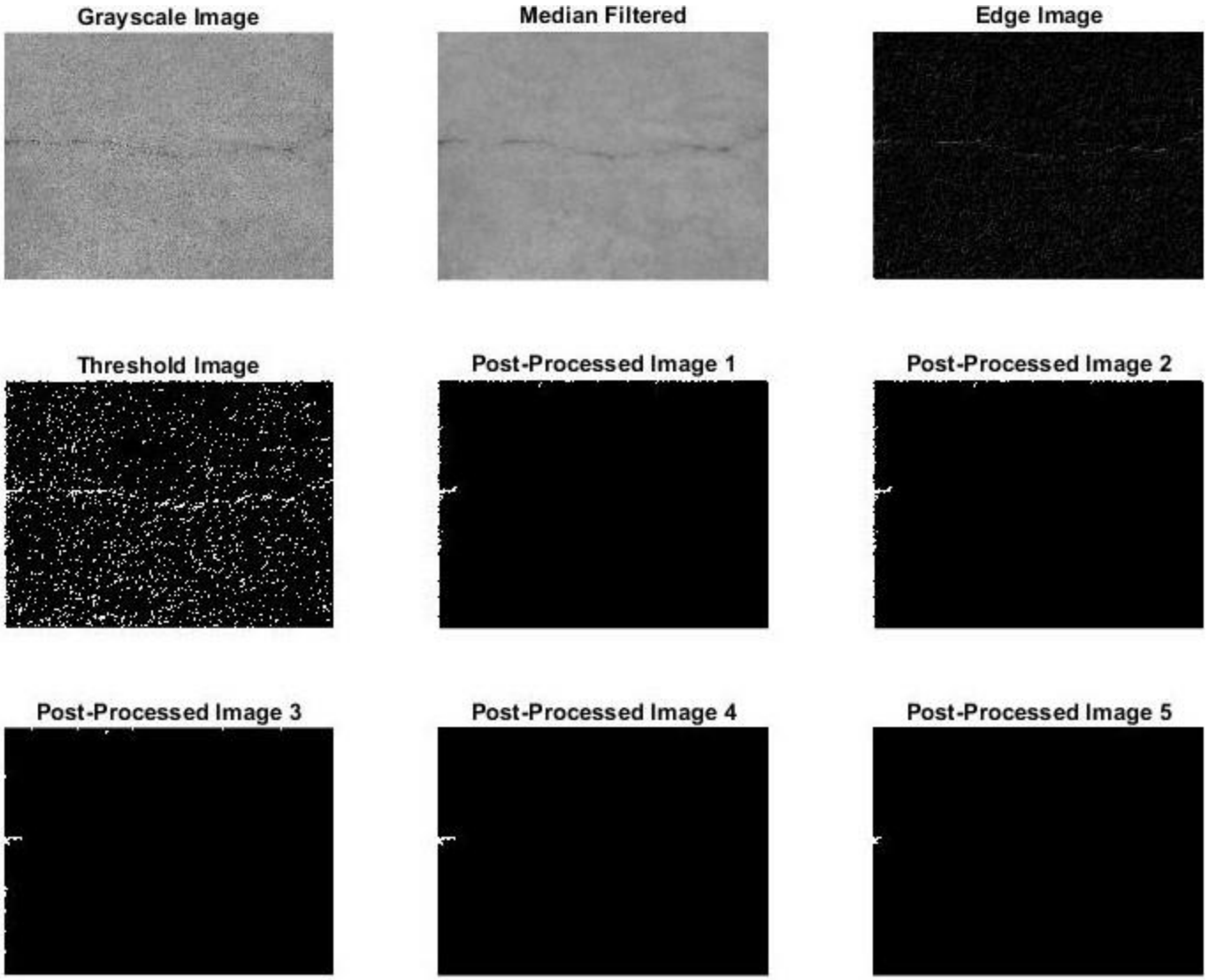


Figure A. 33. Crack detection results using the proposed method on database image #33

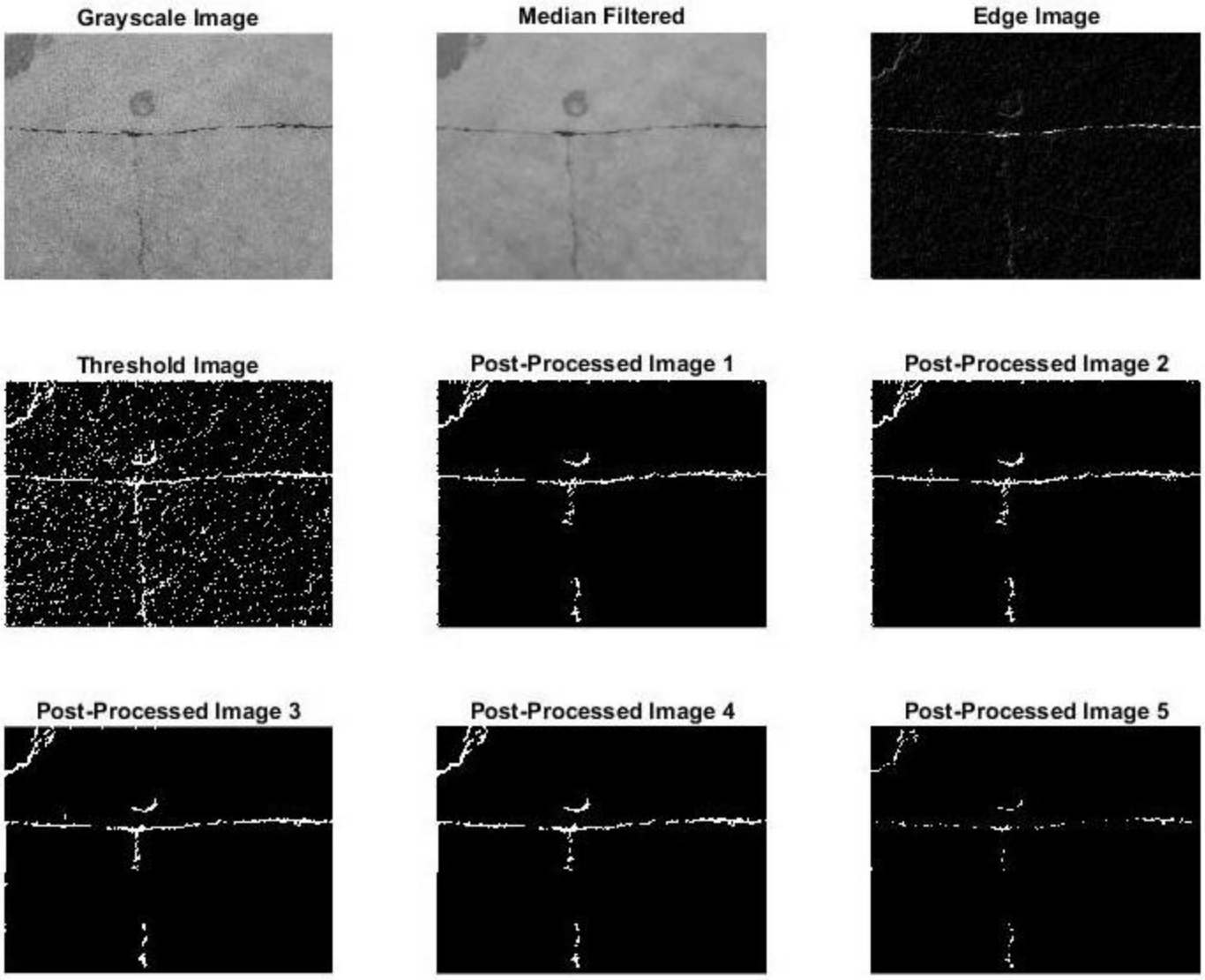


Figure A. 34. Crack detection results using the proposed method on database image #34

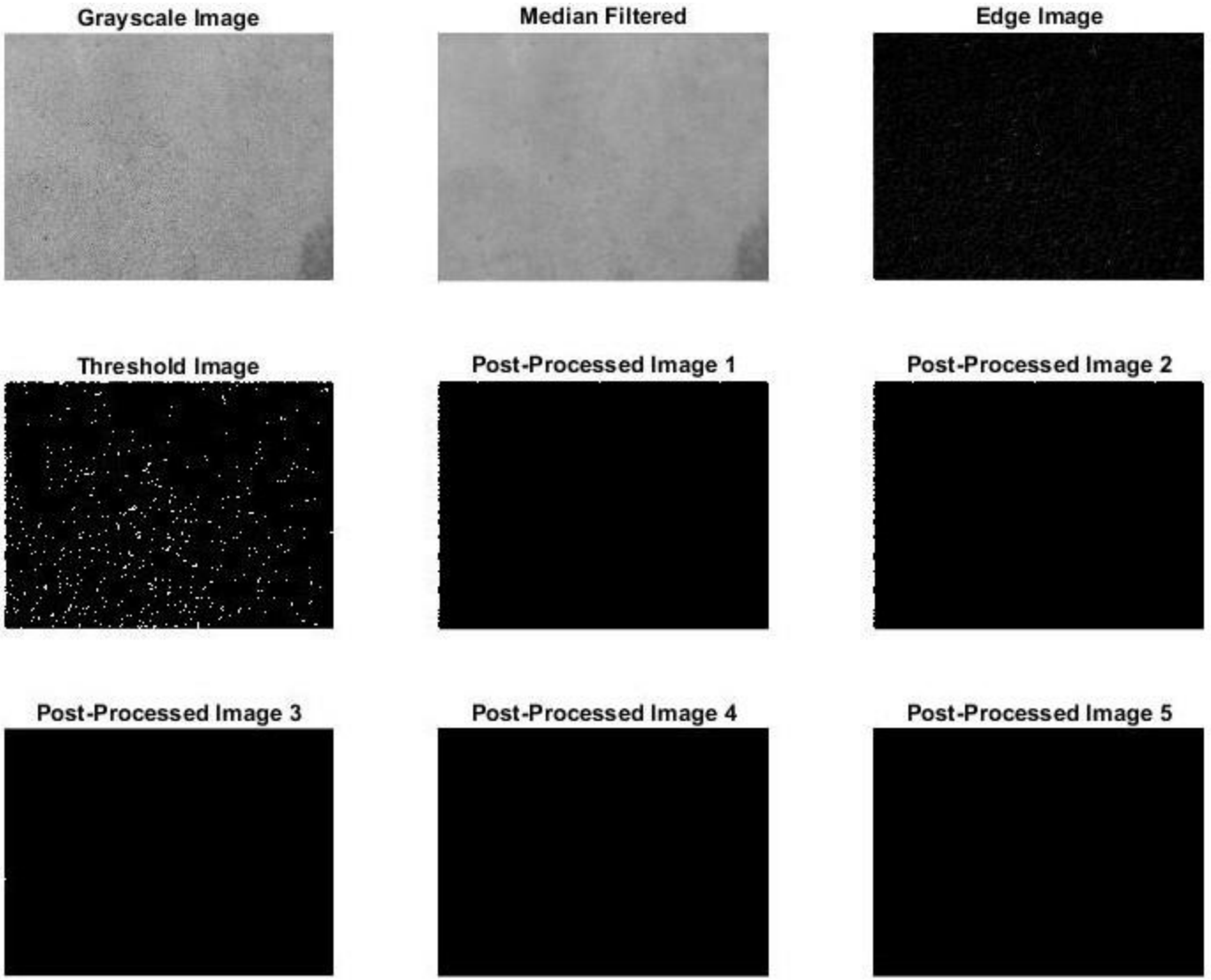


Figure A. 35. Crack detection results using the proposed method on database image #35

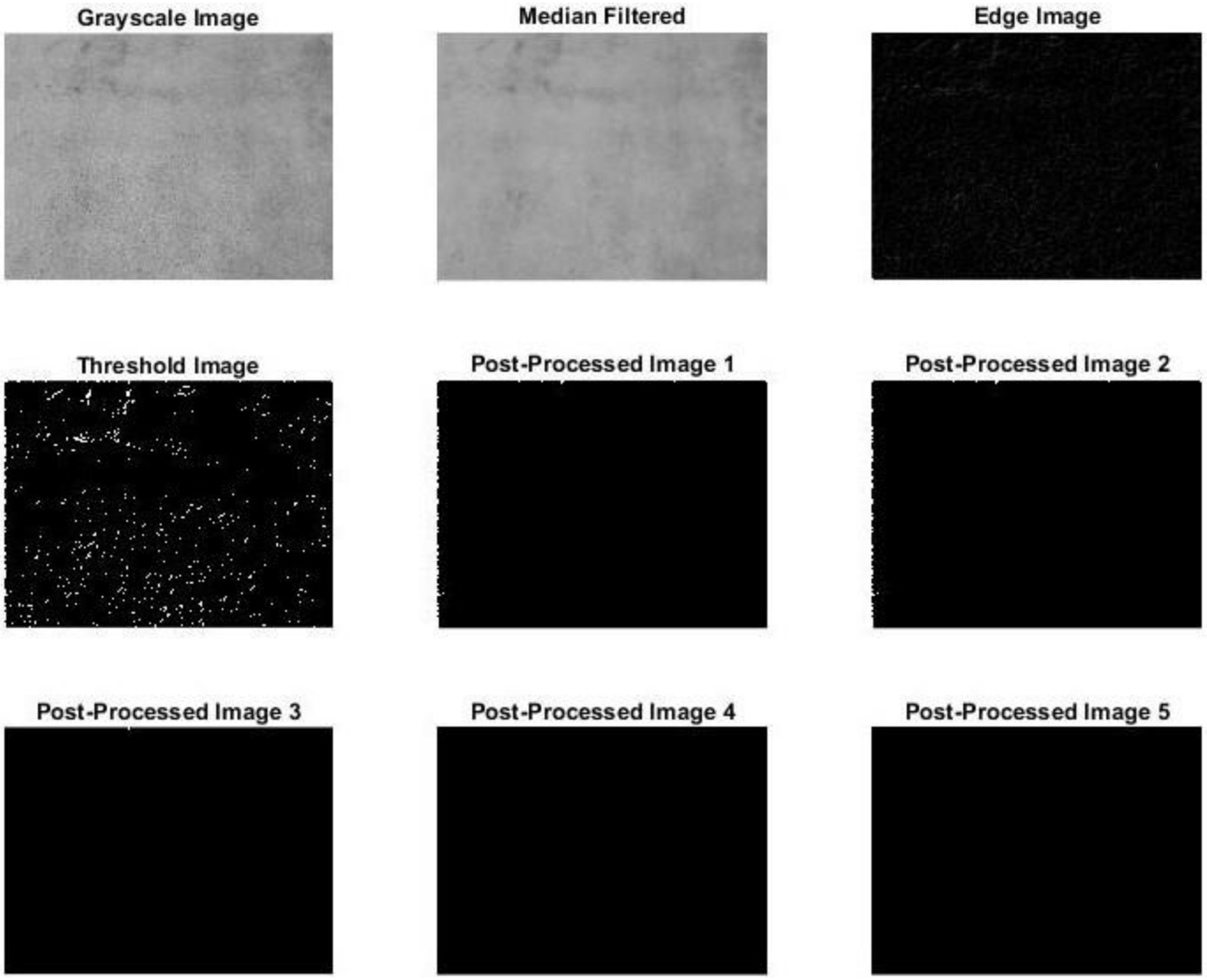


Figure A. 36. Crack detection results using the proposed method on database image #36

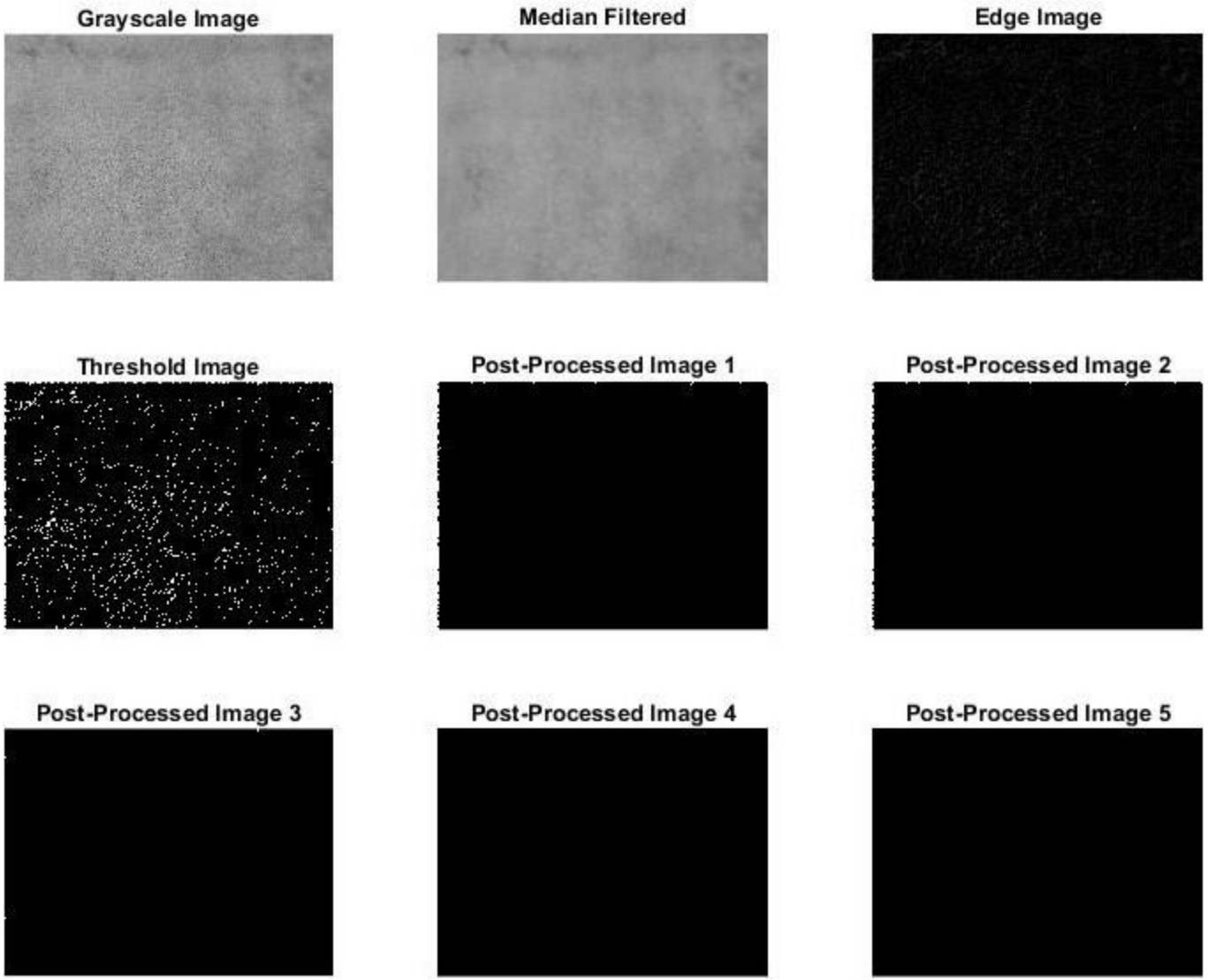


Figure A. 37. Crack detection results using the proposed method on database image #37

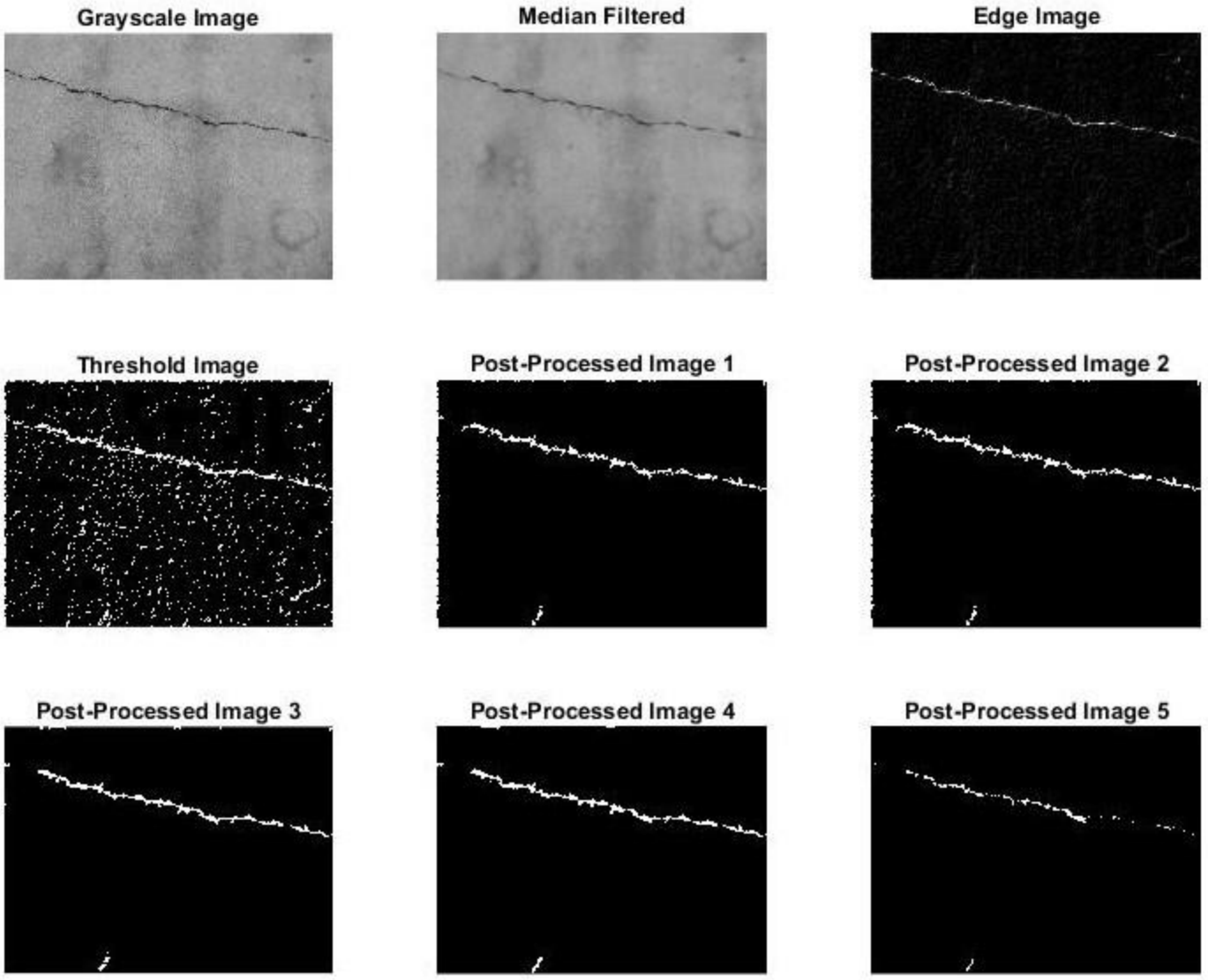


Figure A. 38. Crack detection results using the proposed method on database image #38

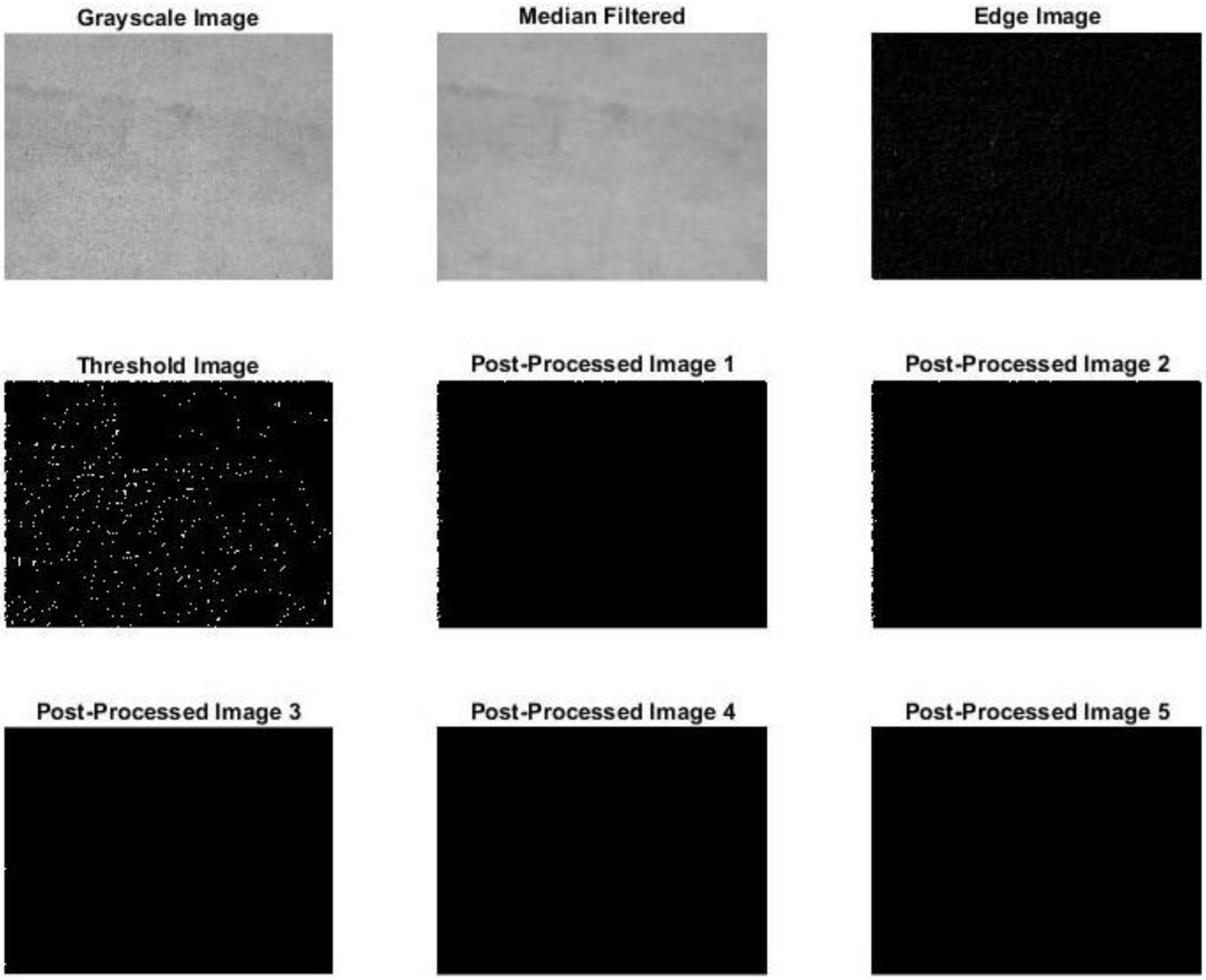


Figure A. 39. Crack detection results using the proposed method on database image #39

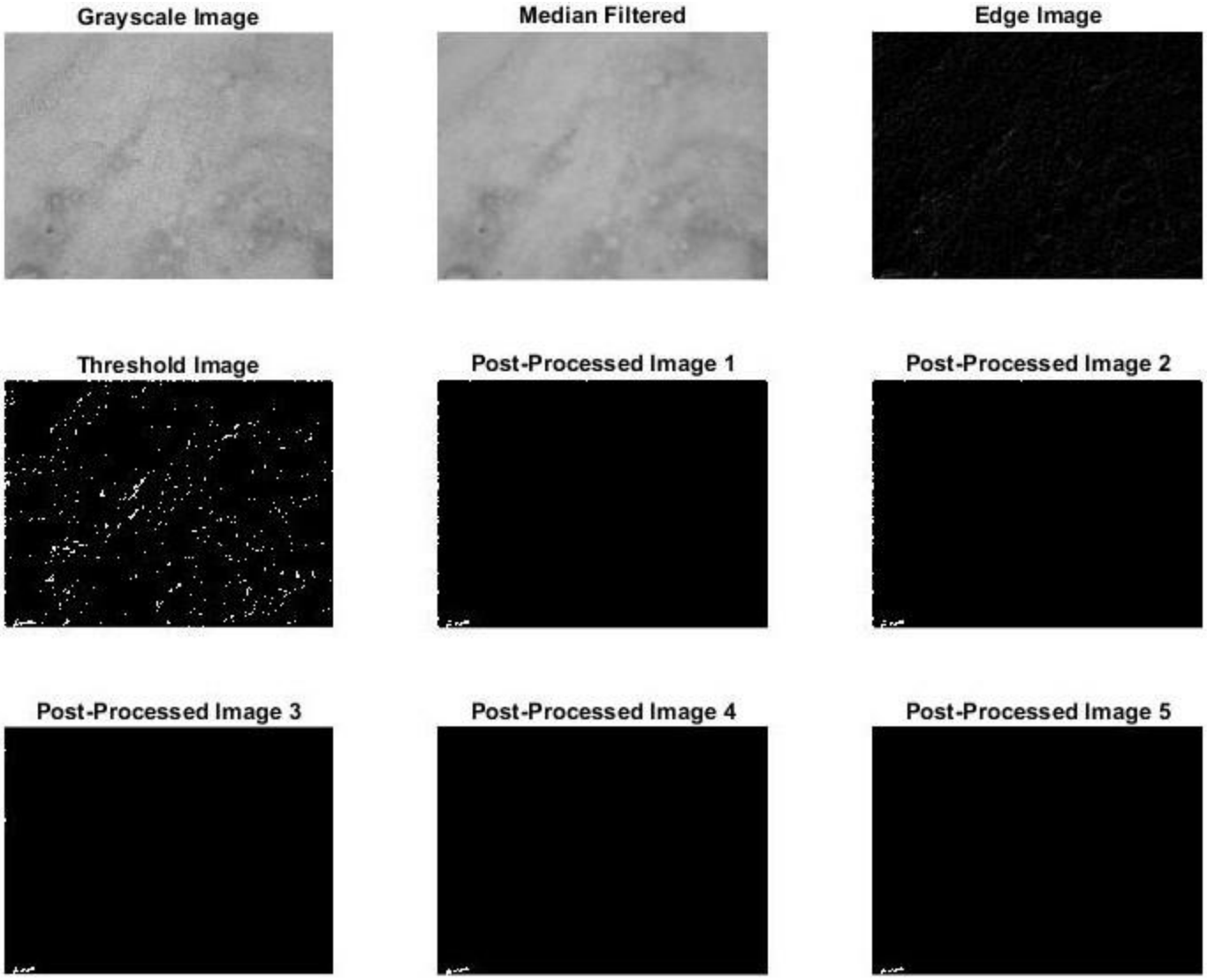


Figure A. 40. Crack detection results using the proposed method on database image #40

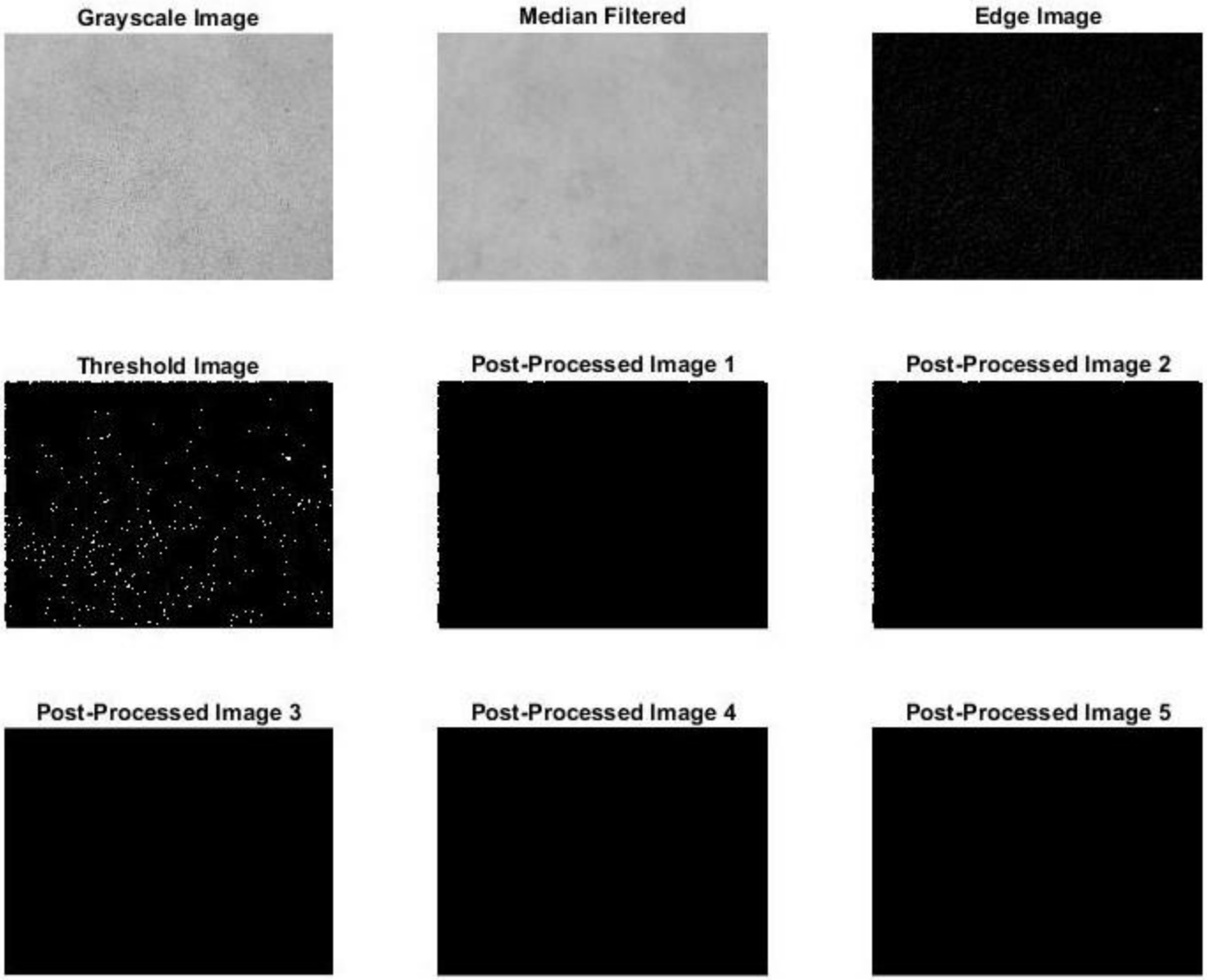


Figure A. 41. Crack detection results using the proposed method on database image #41

8. APPENDIX B: RESULTS OF COMPARISON OF TWO METHODS

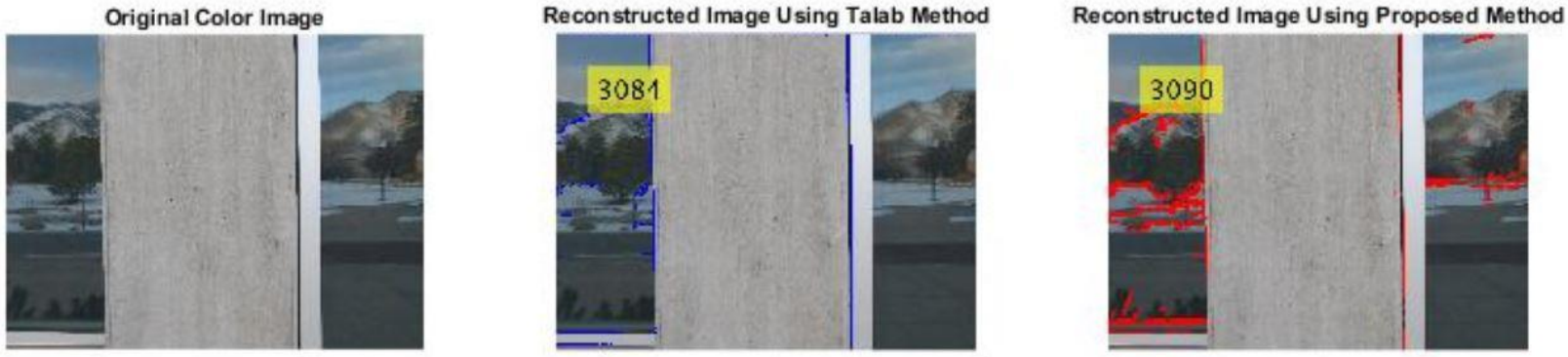


Figure B. 1. Reconstructed color image with cracks and statistics using both methods on database image #1

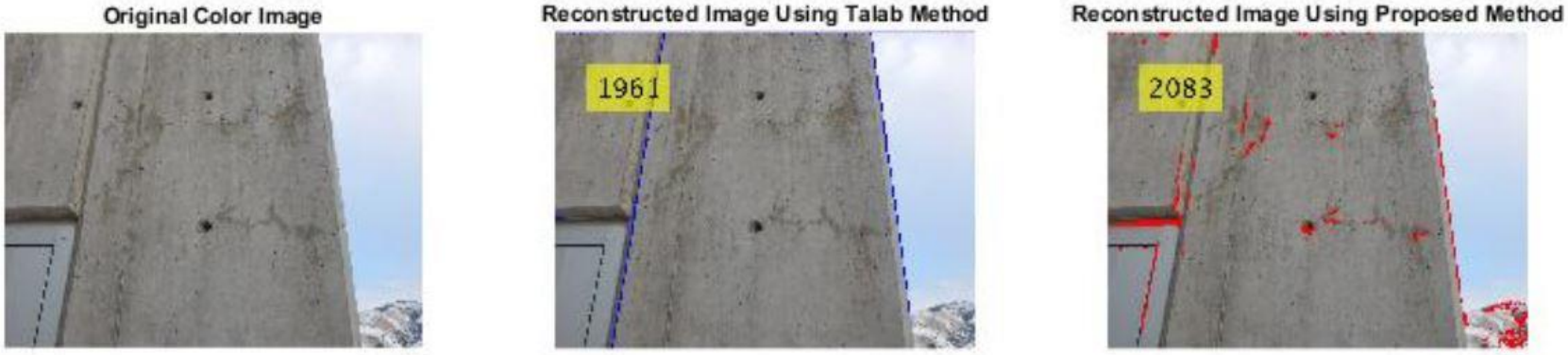


Figure B. 2. Reconstructed color image with cracks and statistics using both methods on database image #2

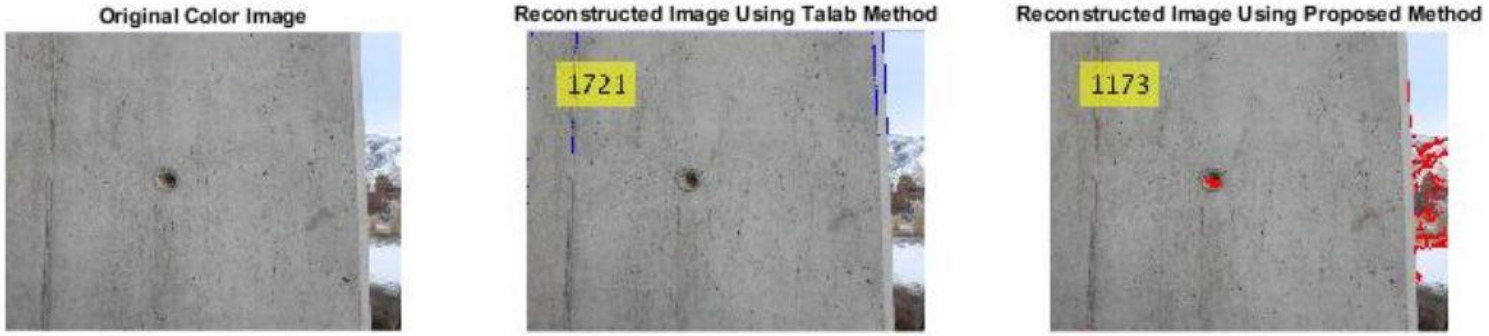


Figure B. 3. Reconstructed color image with cracks and statistics using both methods on database image #3

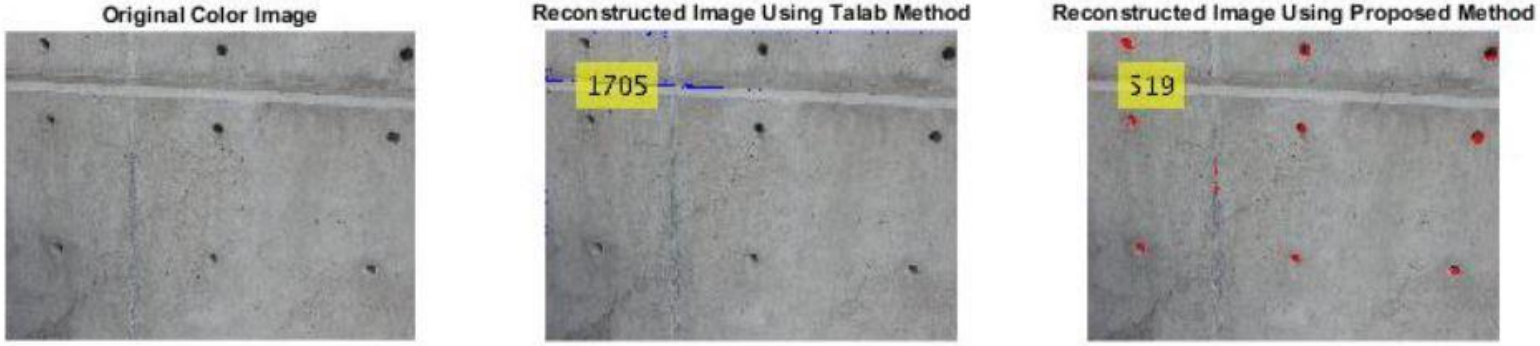


Figure B. 4. Reconstructed color image with cracks and statistics using both methods on database image #4

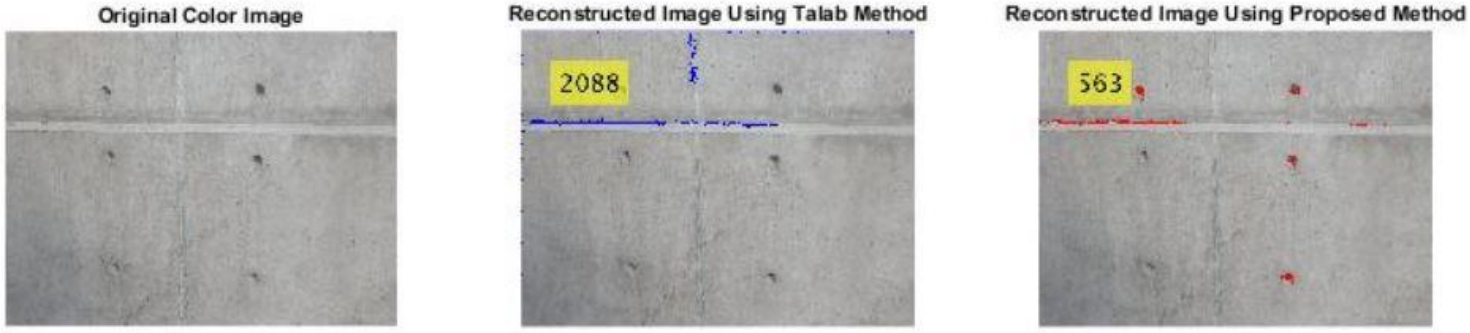


Figure B. 5. Reconstructed color image with cracks and statistics using both methods on database image #5

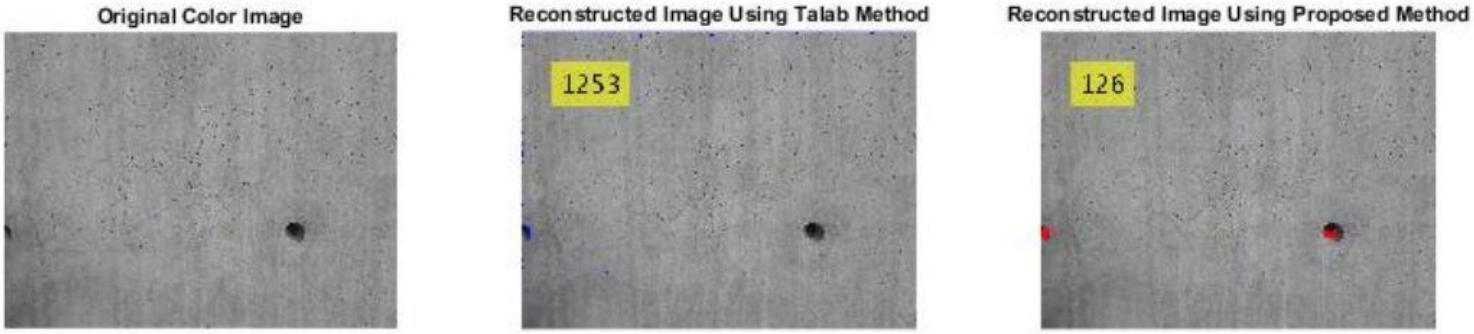


Figure B. 6. Reconstructed color image with cracks and statistics using both methods on database image #6

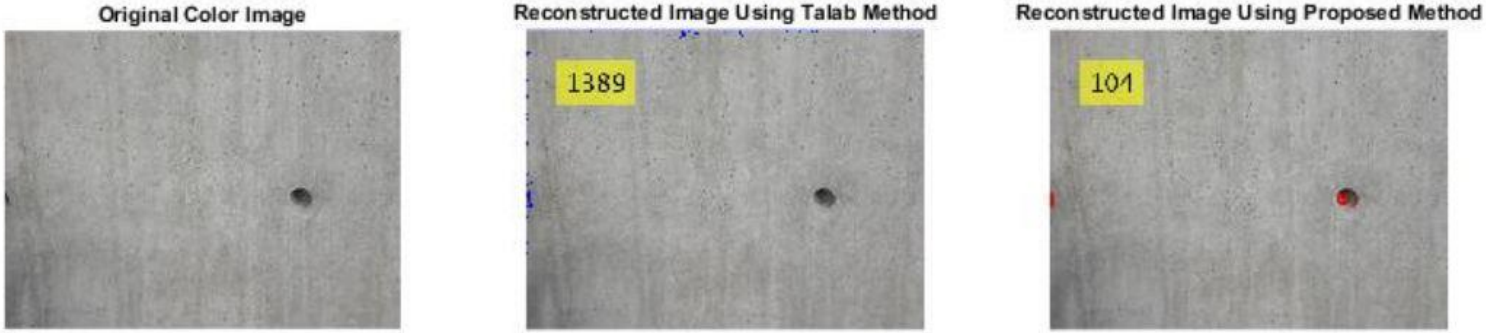


Figure B. 7. Reconstructed color image with cracks and statistics using both methods on database image #7

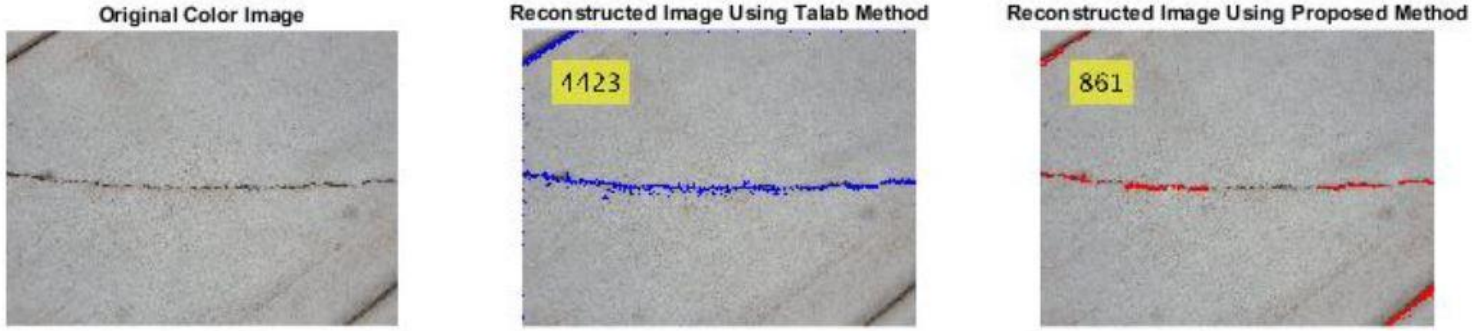


Figure B. 8. Reconstructed color image with cracks and statistics using both methods on database image #8

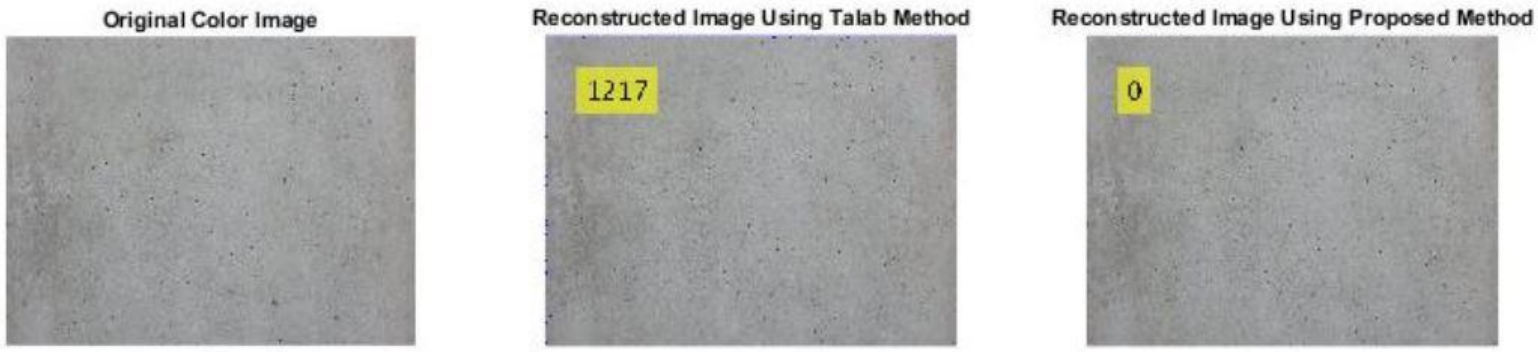


Figure B. 9. Reconstructed color image with cracks and statistics using both methods on database image #9

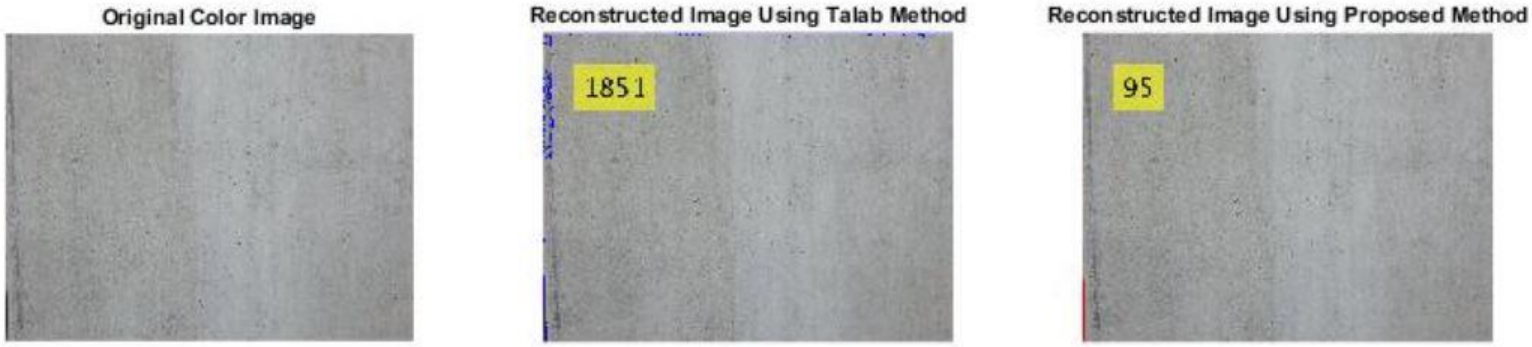


Figure B. 10. Reconstructed color image with cracks and statistics using both methods on database image #10

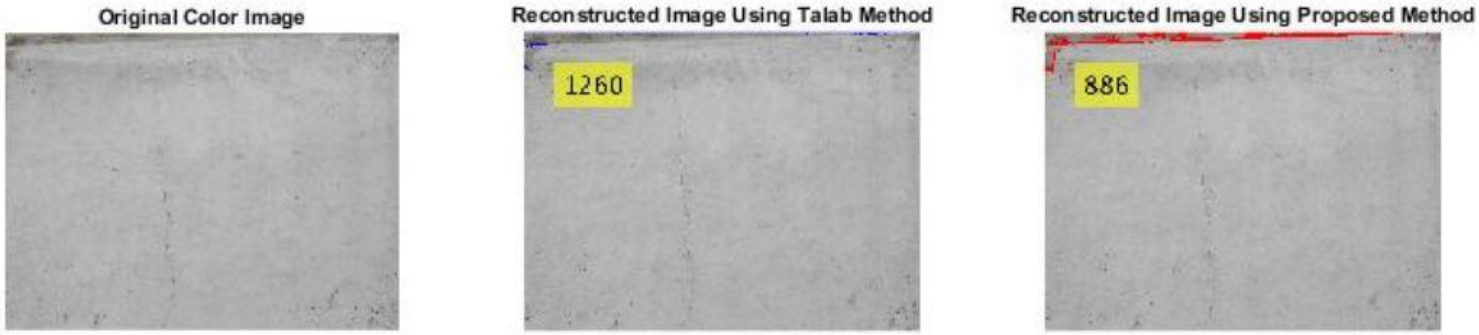


Figure B. 11. Reconstructed color image with cracks and statistics using both methods on database image #11

Figure B. 12. Reconstructed color image with cracks and statistics using both methods on database image #12

Figure B. 13. Reconstructed color image with cracks and statistics using both methods on database image #13

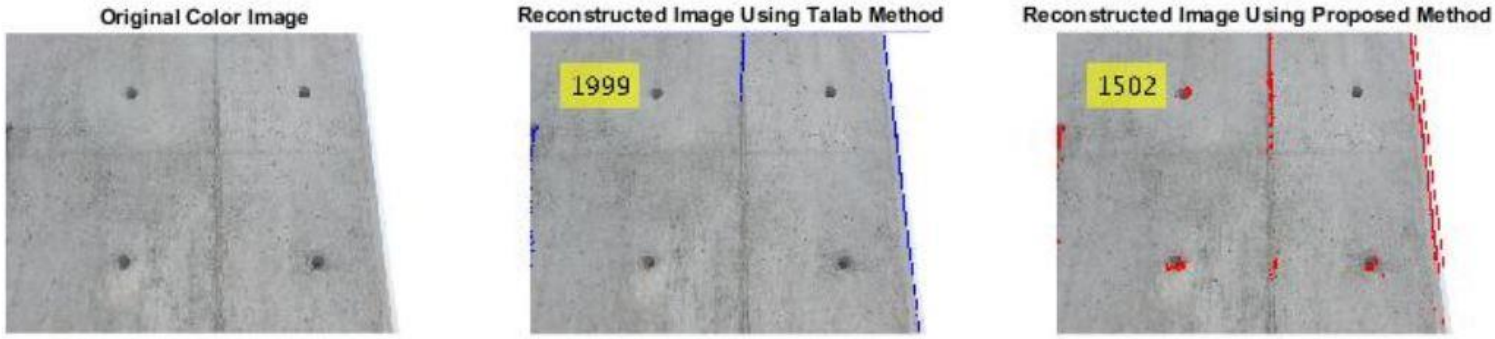


Figure B. 14. Reconstructed color image with cracks and statistics using both methods on database image #14

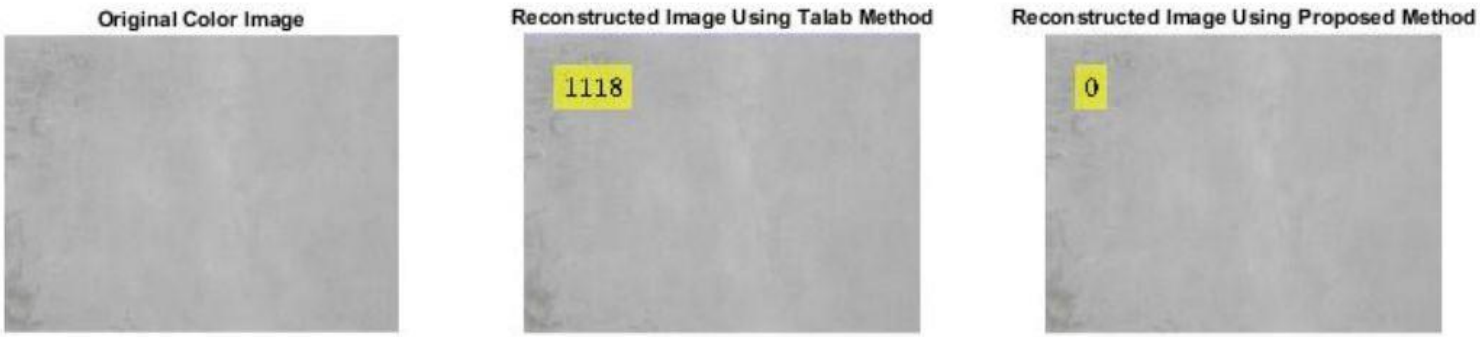


Figure B. 15. Reconstructed color image with cracks and statistics using both methods on database image #15

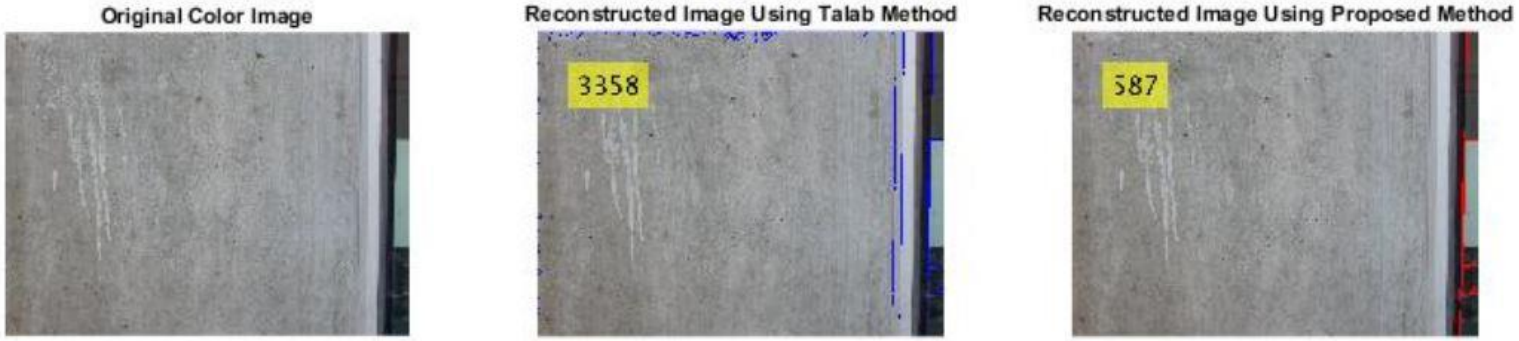


Figure B. 16. Reconstructed color image with cracks and statistics using both methods on database image #16

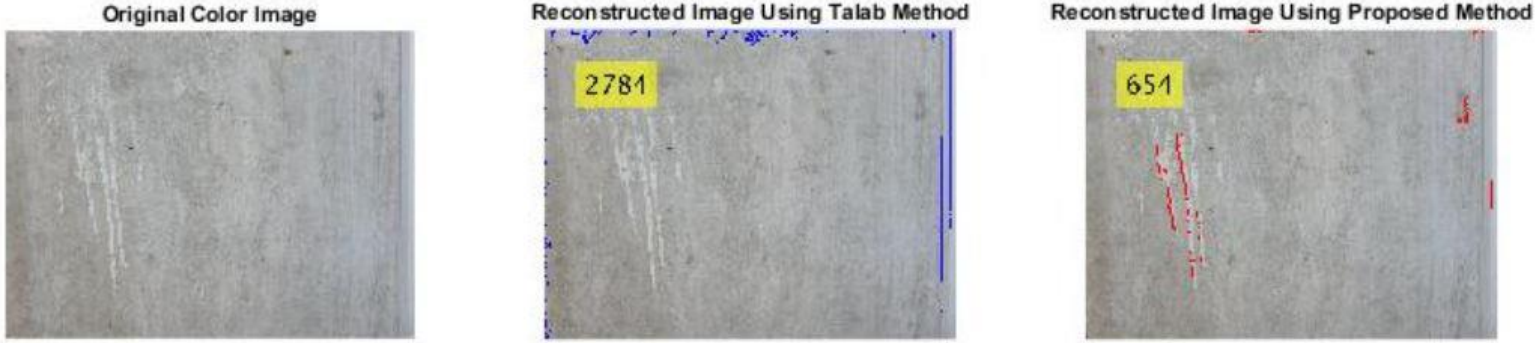


Figure B. 17. Reconstructed color image with cracks and statistics using both methods on database image #17

Figure B. 18. Reconstructed color image with cracks and statistics using both methods on database image #18

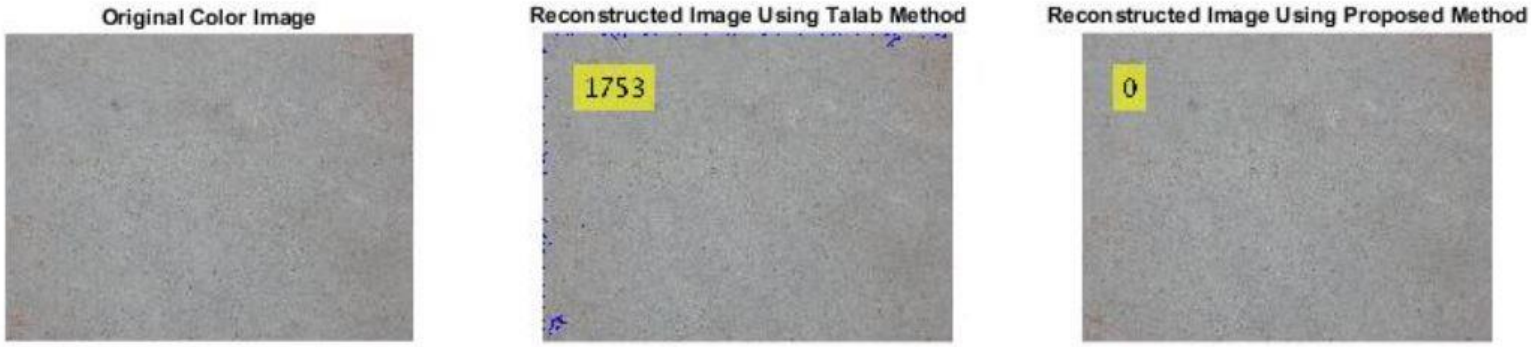


Figure B. 19. Reconstructed color image with cracks and statistics using both methods on database image #19

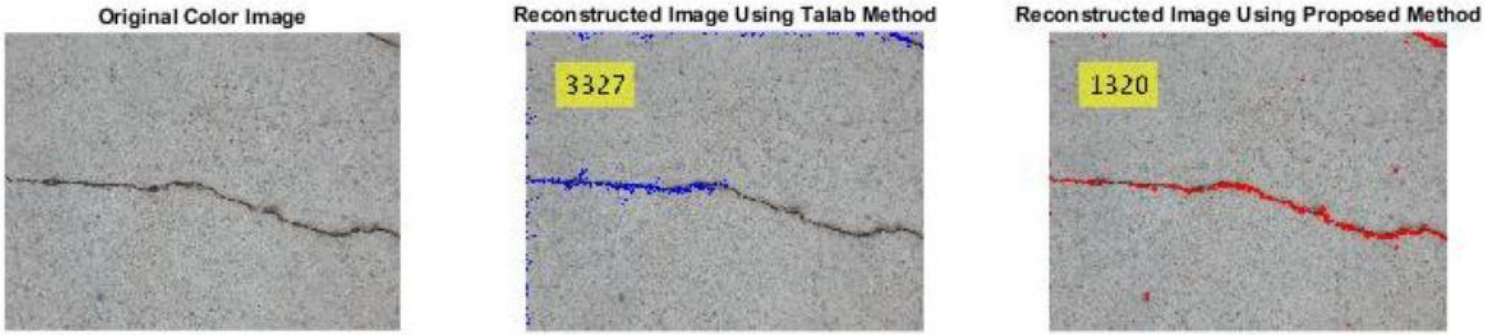


Figure B. 20. Reconstructed color image with cracks and statistics using both methods on database image #20

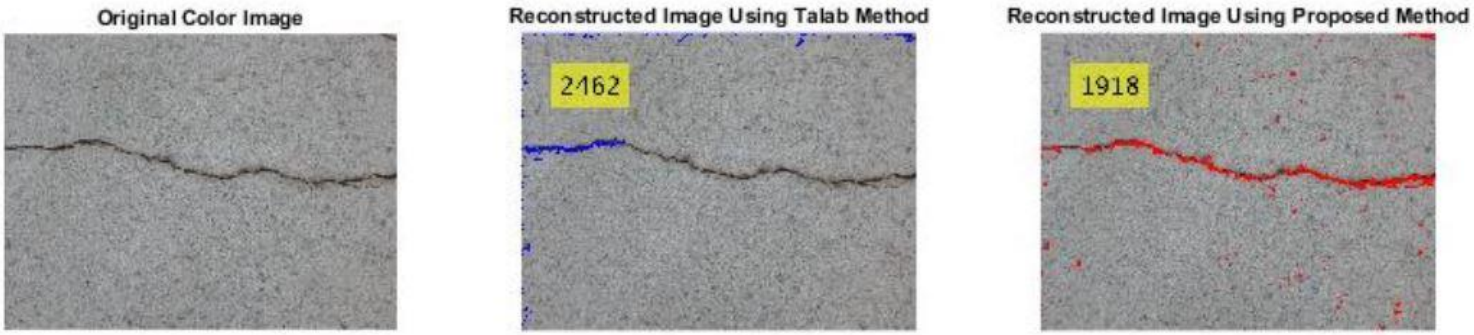


Figure B. 21. Reconstructed color image with cracks and statistics using both methods on database image #21

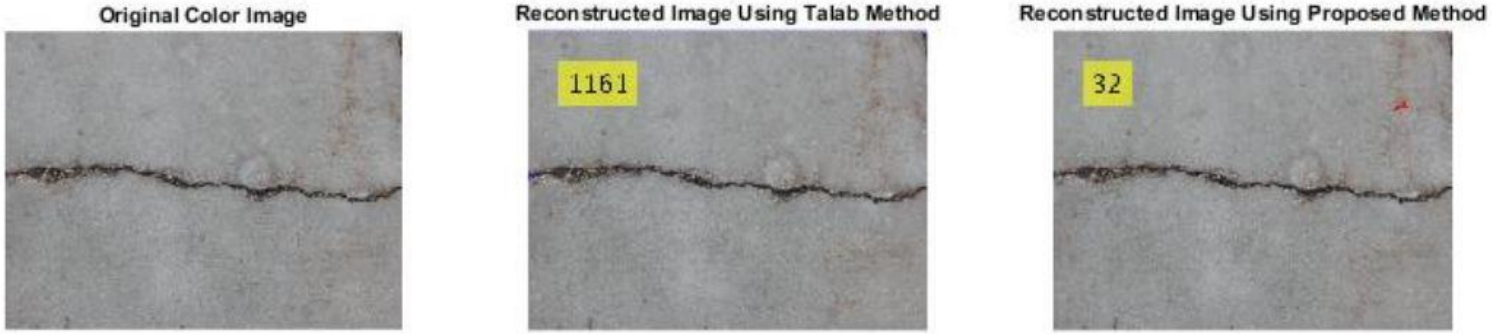


Figure B. 22. Reconstructed color image with cracks and statistics using both methods on database image #22

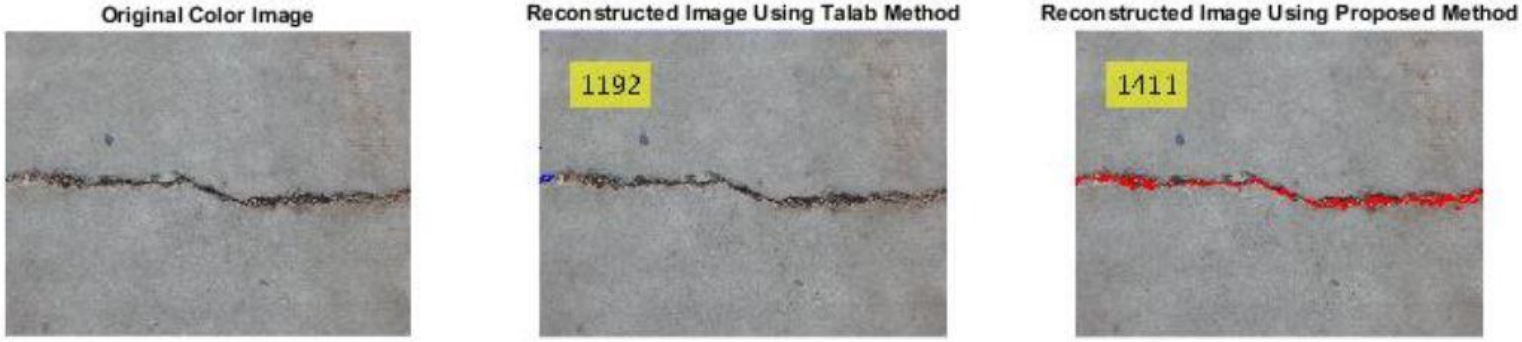


Figure B. 23. Reconstructed color image with cracks and statistics using both methods on database image #23

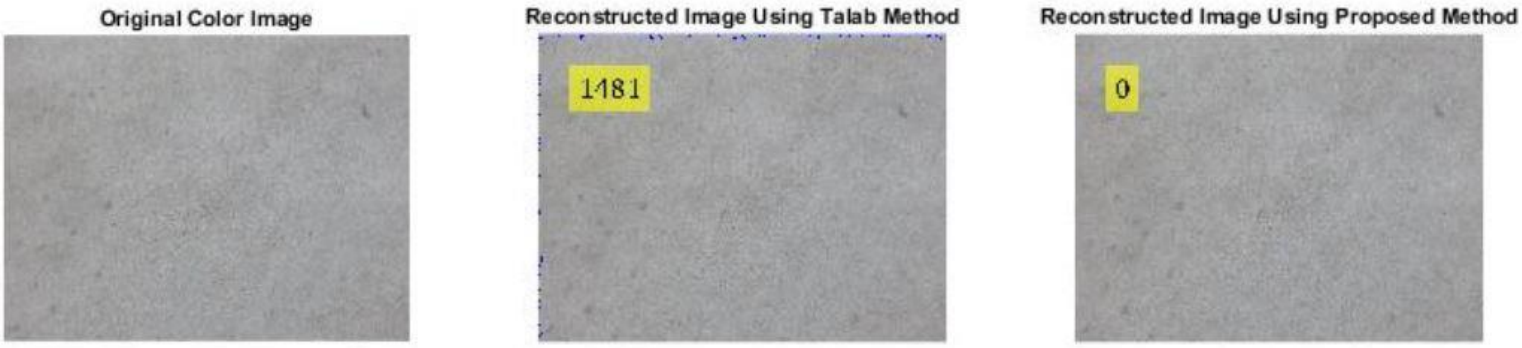


Figure B. 24. Reconstructed color image with cracks and statistics using both methods on database image #24

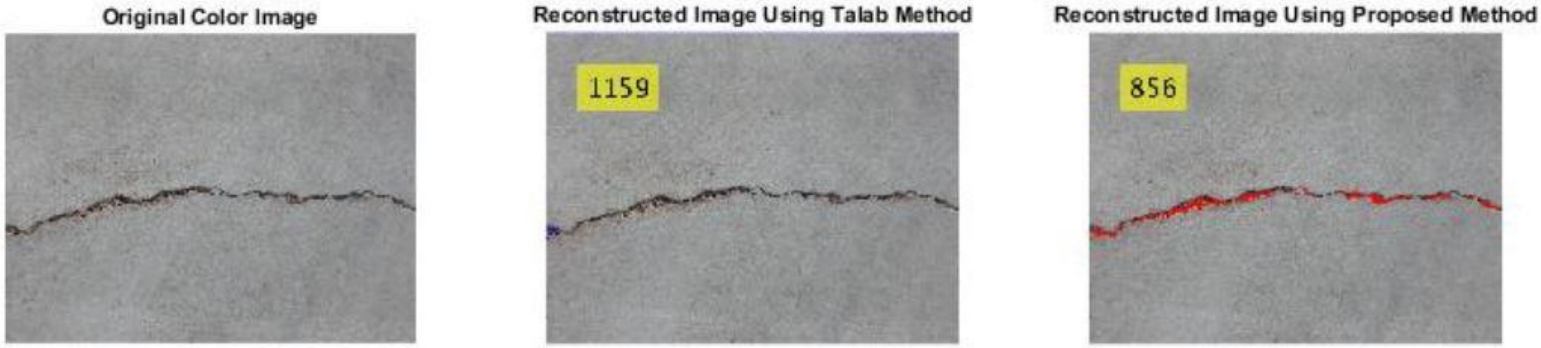


Figure B. 25. Reconstructed color image with cracks and statistics using both methods on database image #25

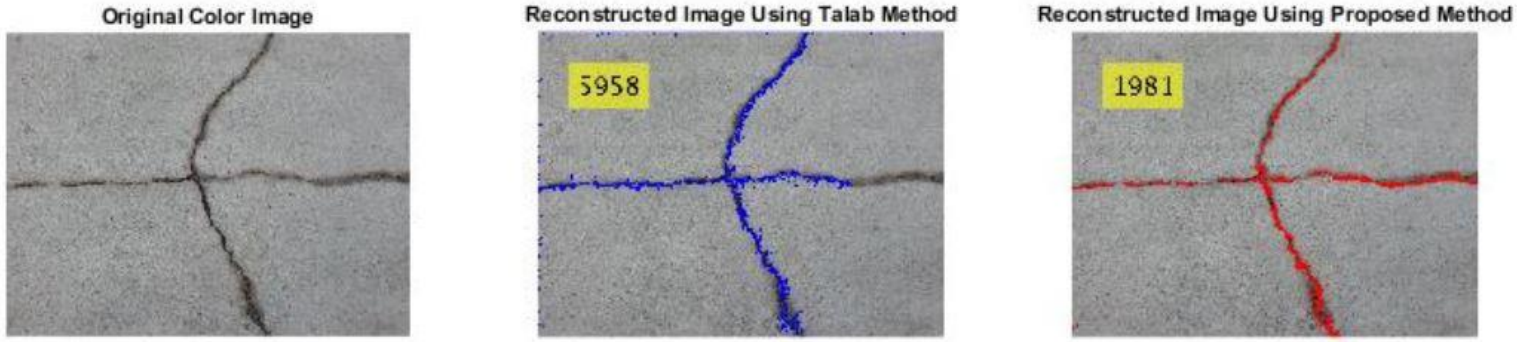


Figure B. 26. Reconstructed color image with cracks and statistics using both methods on database image #26

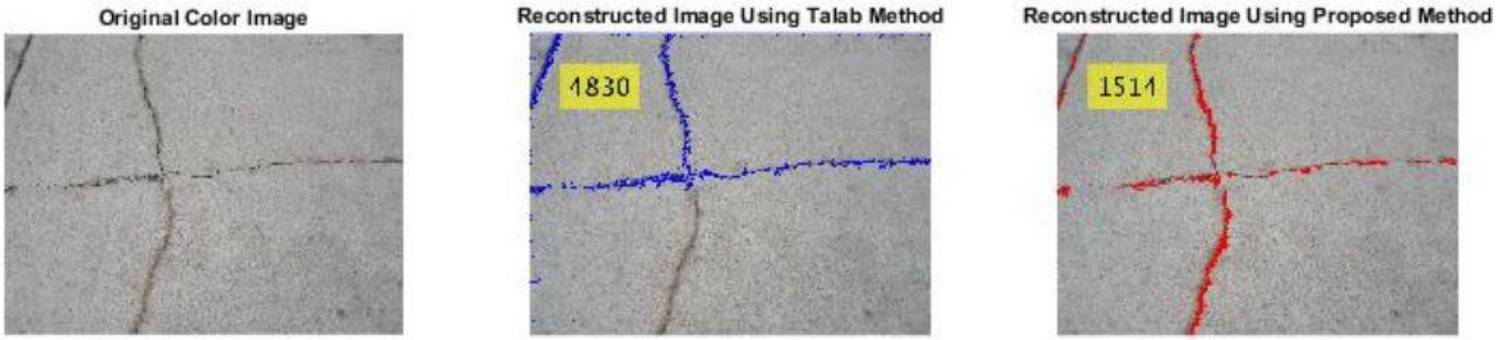


Figure B. 27. Reconstructed color image with cracks and statistics using both methods on database image #27

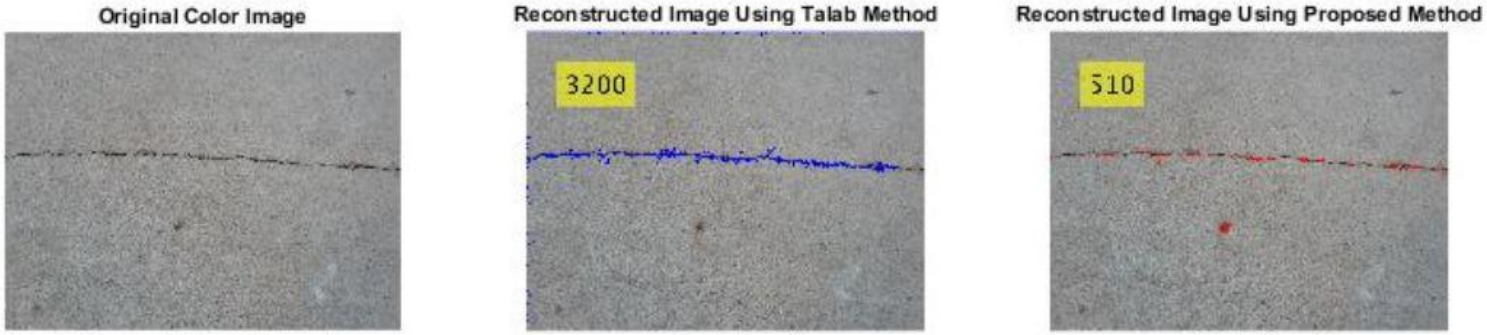


Figure B. 28. Reconstructed color image with cracks and statistics using both methods on database image #28

Figure B. 29. Reconstructed color image with cracks and statistics using both methods on database image #29

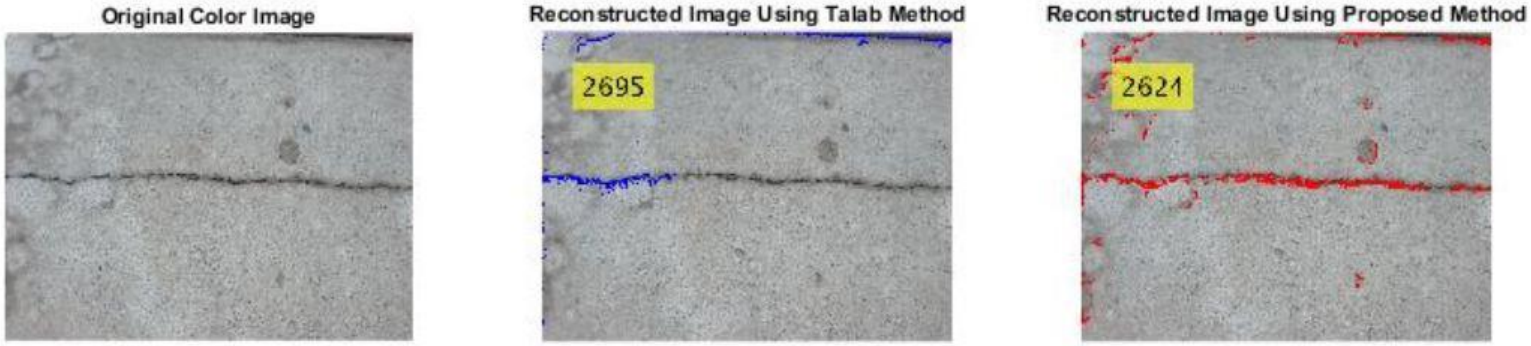


Figure B. 30. Reconstructed color image with cracks and statistics using both methods on database image #30

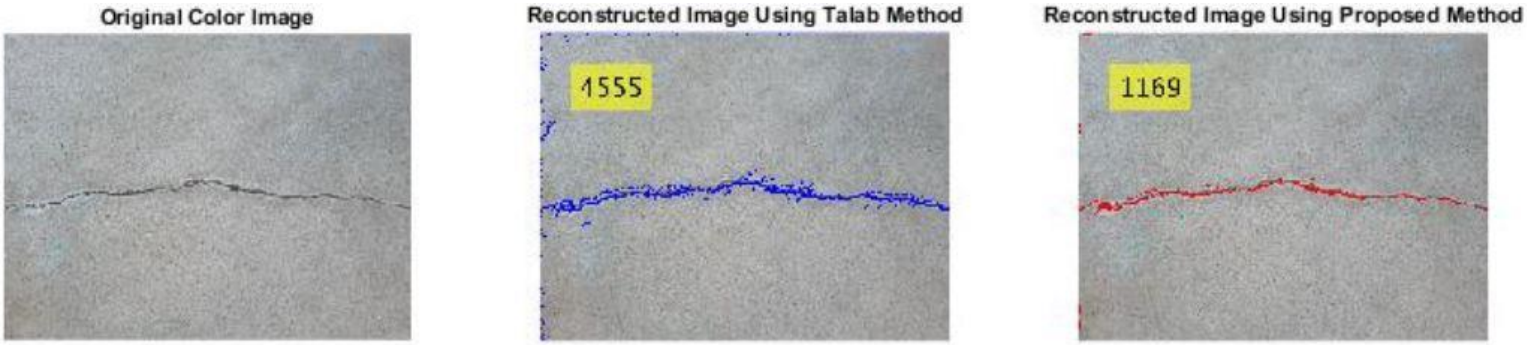


Figure B. 31. Reconstructed color image with cracks and statistics using both methods on database image #31

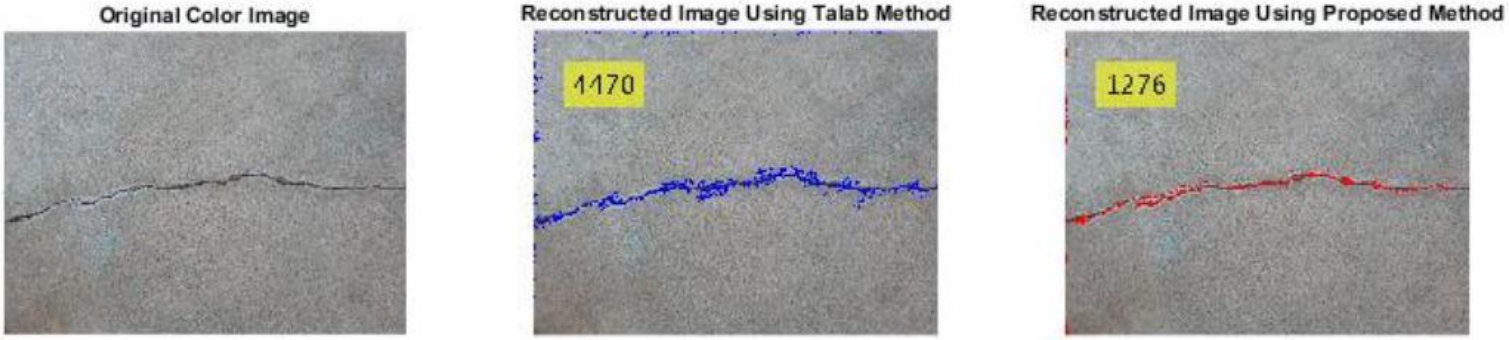


Figure B. 32. Reconstructed color image with cracks and statistics using both methods on database image #32

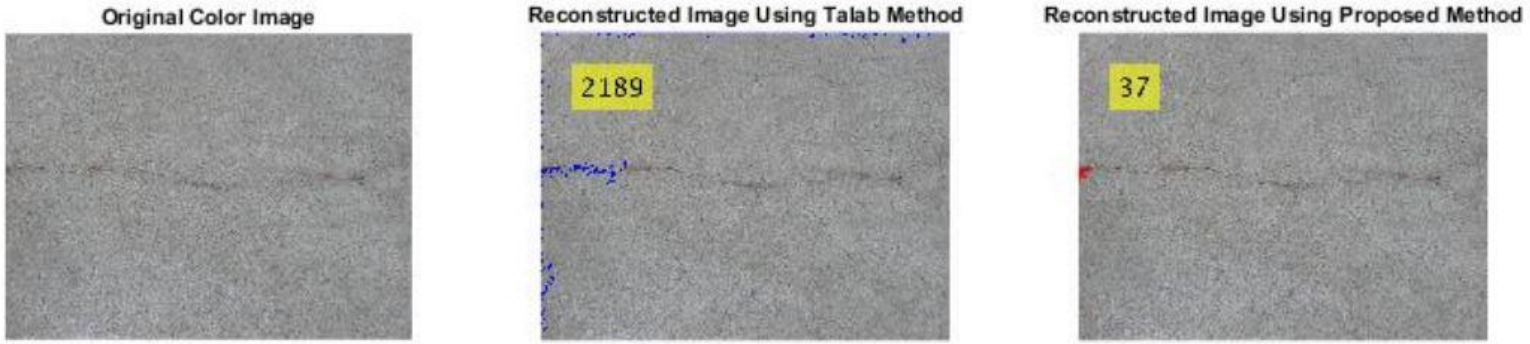


Figure B. 33. Reconstructed color image with cracks and statistics using both methods on database image #33

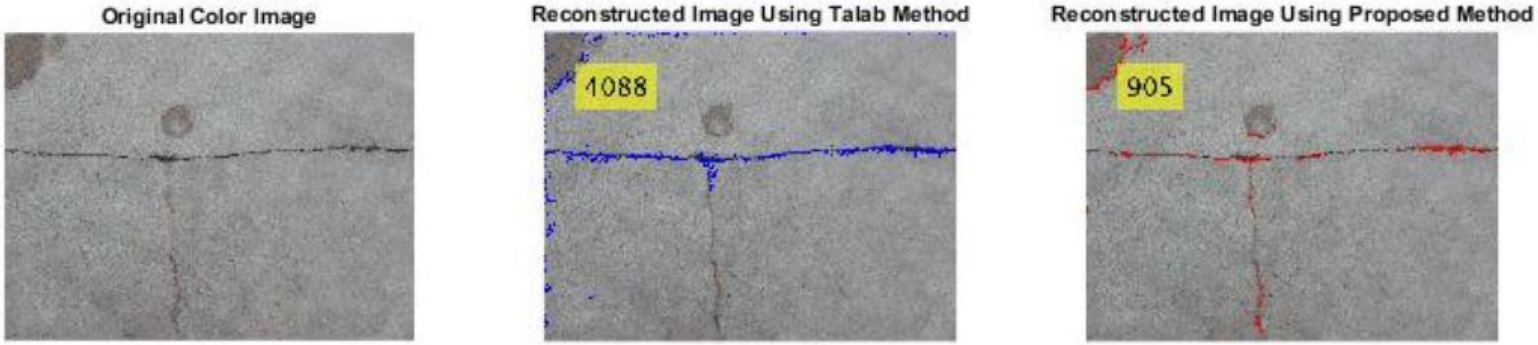


Figure B. 34. Reconstructed color image with cracks and statistics using both methods on database image #35

Figure B. 35. Reconstructed color image with cracks and statistics using both methods on database image #35

Figure B. 36. Reconstructed color image with cracks and statistics using both methods on database image #36

Figure B. 37. Reconstructed color image with cracks and statistics using both methods on database image #37

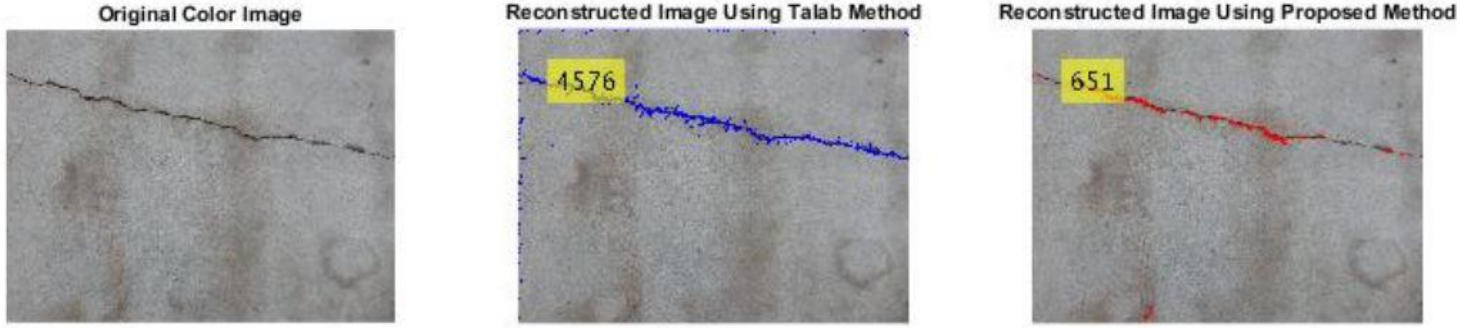


Figure B. 38. Reconstructed color image with cracks and statistics using both methods on database image #38

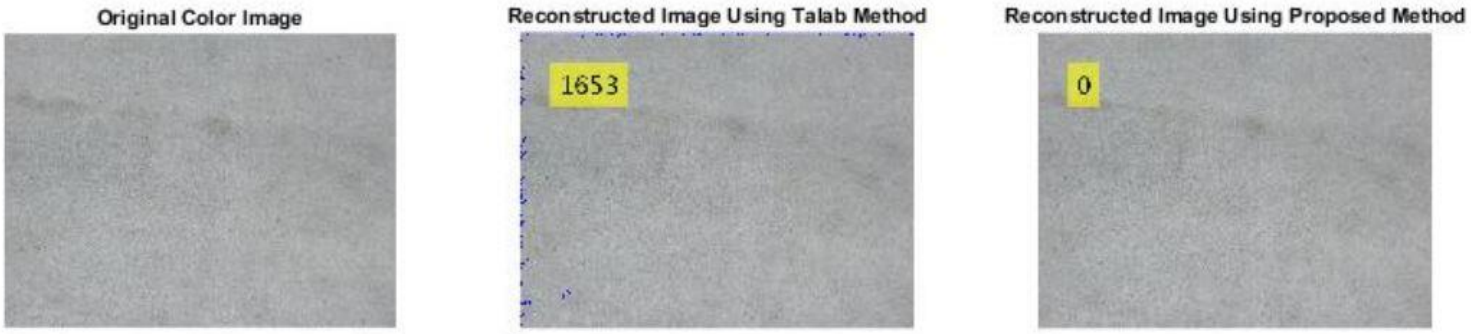


Figure B. 39. Reconstructed color image with cracks and statistics using both methods on database image #39

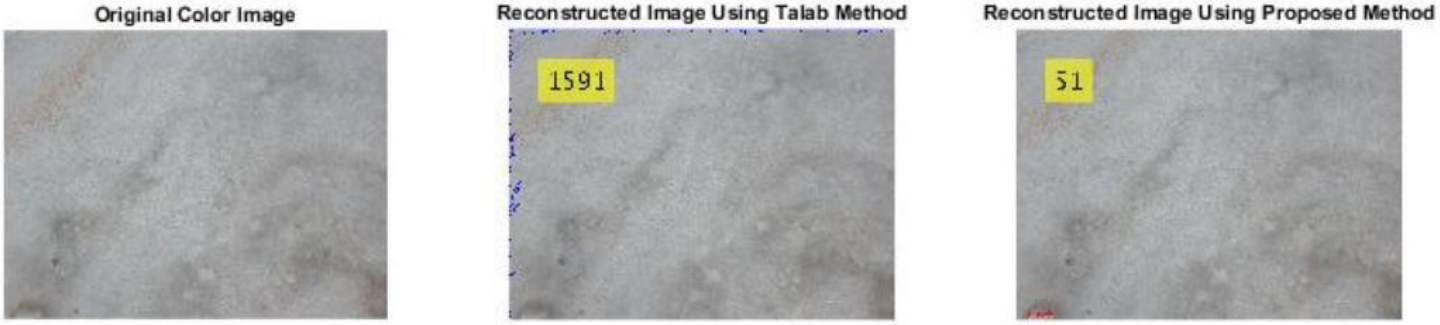


Figure B. 40. Reconstructed color image with cracks and statistics using both methods on database image #40

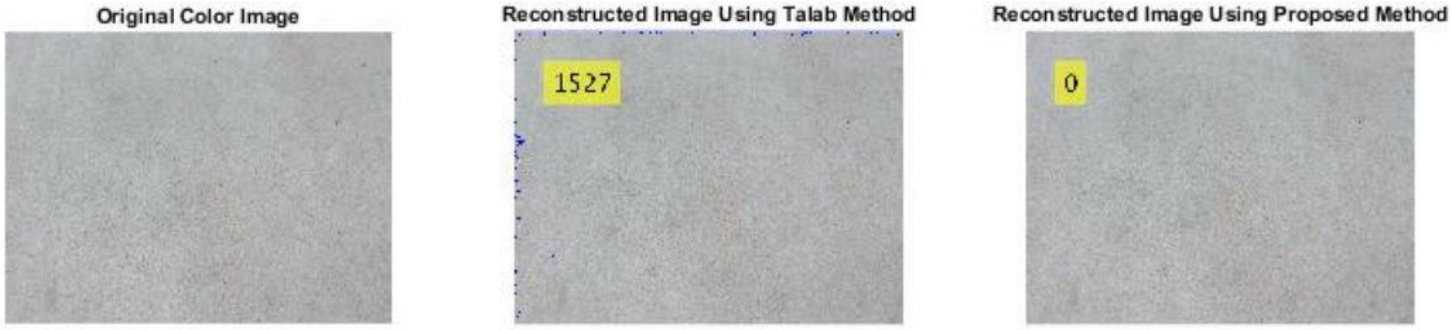


Figure B. 41. Reconstructed color image with cracks and statistics using both methods on database image #41

9. APPENDIX C: PROPOSED METHOD README FILE

Steps to run the code:

Copy all the images into the hard drive of the computer in Drive C in a folder named 'CrackImages'. The code only recognizes the picture with 'jpg' extension and count them as input images, however it is very convenient to change the input image extension to other formats just by changing 2 line in the code. No function used in the code except Matlab built in functions so there is no need to copy the additional function in Matlab directory except for the main code. After copying the pictures and Matlab code in appropriate location, the program could be initiated by hitting 'Run' in Editor Window in Matlab.

The code resizes all the pictures into 480 by 640 pixels to reduce the computational time, but this feature also could be easily changed if necessarily and this would cause no interruption during the code run. The code contains two parts, in the first one the new approach I used to detect cracks is used, and in the second one only paper approach used. The differences will be discussed later.

Inputs: Any RGB image with any size having 'jpg' extension saved in the C hard drive of the computer in a folder named 'images'. As mentioned before program could easily be modified for other formats. The Sobel edge detectors defined again in the program. Since the pictures were taken in daylight using flash, light did not change the pictures, thus the output pictures also did not change significantly. To reduce the computational effort and time, the redundant pictures were deleted from the current database.

Outputs: The output of the program for each picture involves 10 pictures which will be saved in each cell layer (number of cell layers are equal to the number of input pictures) with

different names for each cell. ‘grayIm’ is used to collect the gray-scale images for each image, ‘edge’ is the cell array for result of applying Sobel edge detector on the gray-scale image. Seven enhancements have been done on the ‘edge’ pictures using different methods and thresholding, which all resulted in binary images at each step, so seven cell arrays used to save all the changes in the pictures using cell name ‘binaryIm’. ‘RecIm’ is defined to save reconstructed RGB images with the cracks imposed on in red color, and, finally, the cell ‘Crancks_L’ store the whole number of 1s in the seventh binary images, for each image. These images could be interpreted as the total pixels with cracks i.e. crack length. The graphical output consists of nine sub-images for each input image, including all the mentioned process. For the second part of the program, i.e. paper approach, the outputs include a cell array ‘paper_ap’ for all the intermediate results and also a vector called ‘paper_cracksL’ for number of pixels detected with cracks. The graphical output was also provided using eight sub-image for each image.

The basic operation performed: The whole idea is to find the cracks using thresholding. The paper used Otsu’s method to transfer the image to binary, and then applied the thresholding. The main functions used this code are: 1. ‘dir’: to determine the location of the pictures in hard drive. 2. Using cell properties to automatically add results of each image to a corresponding cell layer. 3. A 5 by 5 median filter, used to smooth the pictures since the concrete surface is uneven, which could help the program not to recognize them as cracks (not used in paper approach). 4. Sobel edge detectors used to find the edges in the pictures. 5. ‘graythresh’ function is used to define the thresholding value to convert the grayscale to binary. The value of 0.2 of the Otsu’s threshold was used in the first approach, while in paper, the whole value implemented. The reason was imperia, meaning this value seems to more applicable to the database used as input. 6. Using ‘regionprops’ and ‘bwlabel’ to find the area of each connected component and eliminate the areas

less than a cutoff value. The matching cutoff for 0.2 threshold, considered to be 200, and for paper approach used 30. 7. Find the horizontal and vertical connected components and remove them using ‘regionprops’ function (not used in paper approach). 8. ‘bwmorh’ function used to apply morphological operations to make the resultant image more consistent (not used in paper approach). 9. ‘bwareaopen’ function implemented to eliminate the are regions containing 50 pixels or less to eliminate reminded noises or small areas in resultant image so far. 10. Changing the color space of the original picture to HSV space using function called ‘rgb2hsv’. Since it is very likely to have non-cracked lines in the original picture, such as color mark on members, edges of member and so on, the lines identified as cracks so far should pass another filter. It has been observed that S values in HSV domain at the locations of the cracks have a relatively larger magnitude. The S space domain value of identified pixels up to this level will be applied to define another thresholding operation. This allows the program to some extent separates the cracked pixels from false positive pixels. Using the Red, Green and Blue component to impose the final cracks into the RBG image (not used in the paper approach). 12. Compute the ultimate number of pixels with cracks, and attach that number to the reconstructed image using ‘insertText’ function (not used in paper approach). 13. Showing the original, intermediate and ultimate result for each image.

Measures to evaluate effectiveness of the method: The visual inspection has been used to see the effectiveness of the methods and, since cracks are not very small, it is a reasonable method. The final results of both approaches can be compared together. As the result shown in general, the first approach gives less false positive, but it requires more computational effort to solve the problem.

External package: Matlab 2015a used to write the code with image processing toolbox and no external packages used to write or run the test.

Special Data: No special data has been used for this problem. The database consists of simple RGB pictures.

10. APPENDIX D: MATLAB CODE

```
% % Sattar Dorafshan
clc
% Clean the work space
clear
% Clear all assigned values to variables
close all
% Close all the images or etc
file=dir('C:\CrackImages\*.jpg');
% Specify the address e.g drive and folder name of the database
% and specify to recoginze all the files with jpg format as database
% and save them in the vector named 'file'
NF=length(file);
% NF or number of files in the folder
orImage=cell(NF,1);
% Creating a cell array for original RGB images in database
grayIm=cell(NF,1);
% Creating a cell array for gray-scale gray images for each RGB image after
% applying 5 by 5 median filter
edge=cell(NF,1);
% Creating a cell array edge image of each gray-scale image after applying
% soble edge detector
binaryIm=cell(NF,7);
% Creating a cell array for each binary image during each stage of the code,
% '1' binary image constructed from applying Otsu's thresholding method
% '2' binary image constructed from applying area cutoff value of 200
% '3' binary image constructed from applying orientation cutoff value to
% ramove vertical and horizontal lines
% '4' binary image constructed from applying morphological operation of
% majority
% '5' binary image constructed from applying morphological operation of
% bwareaopen to connect objects
% '6' binary image constructed from applying another threshold on S
% component of the HSV of original image to reduce the non-concrete
% background
% '7' binary image to show the skleton of the cracks
```

```
RecIm=cell(NF,1);
% Cell array to save the reconstructed image
Cracks_L=cell(NF,1);
% Cell array to save length of cracks detected in final step
Hx=[-1 -2 -1;0 0 0;1 2 1];
% Sobel filter to detect Horizontal lines
Hy=Hx';
% Sobel filter to detect Vertical lines
paper_ap=cell(NF,7);
% Define cell array to save resultant images using paper approach
paper_cracksL=zeros(NF,1);
% Vector to save length of cracks detected in final step
position=[50,50];
% Define the position of the label
for i=1:NF
% Loop start, each i for each original images
orImage{i}=imread(fullfile('C:\CrackImages\',file(i).name));
orImage{i}=imresize(orImage{i},[480,640]);
% Read all the pictures in the specified folder in line 10 and resizekk
```



```
% it to aceptable ratio
grayIm{i}=medfilt2(orImage{i},[5,5]);
% Convert every image into gtay scale and also apply the median filter
% to smooth the pictures since the concrete surface is not smooth
edge{i}=abs(imfilter(grayIm{i},Hx))+abs(imfilter(grayIm{i},Hy));
% Finding the edges of the picture using absolute value of sobel edge
% detector in both vertical and horizontal direction
binaryIm{i,1}=im2bw(edge{i},0.2*graythresh(edge{i}));
% Using Otsu's method on edges of each image with a thresholding of 0.2
% of Otsu's thresholding to obtain a binary image consitent with cracks
L1=bwlabel(binaryIm{i,1});
% Labeling the bianry image usuing connected component with 8
% neighbourhood
stat1=regionprops(binaryIm{i,1},'Area');
% Finding the area of each connected component in the label martix
% using ragion properties which save the results in structure arrays
```

```
area=cat(1,stat1.Area);
% Transform the structure array to a matrix
v1=find(area>200);
% Find the area larger than 200 for each picture
binaryIm{i,2}=ismember(L1,v1);
% Keeping only connected objects with area bigger than 200
L2=bwlabel(binaryIm{i,2});
% Obtain another label matrix for binary images from prevvious step
stat2=regionprops(binaryIm{i,2},'Orientation');
% Finding the orientaton of each region and save the results in
% structural array
orientation=cat(1,stat2.Orientation);
% Convert the structural array to a vactor
v2=find(abs(orientation)<89 & abs(orientation)>1);
% Find ragions with the orientation angle magnitude smaller than 89
% degrees and bigger than 1 degree to find vertical and horizontal
% regions
binaryIm{i,3}=ismember(L2,v2);
% Eliminate the regions with vertical and horizontal orientations
binaryIm{i,4}=bwmorph(binaryIm{i,3},'majority',inf);
% Use morphological operation to fill the holes in the binary images
binaryIm{i,5}=bwareaopen(binaryIm{i,4},50);
% Use morphological operation to eliminate small area in the resultant
% binary image with area bigger than 50
[h,s,v]=rgb2hsv(orImage{i});
% Finding the component of the original RGB image in Hoe color space
tempS=s(binaryIm{i,5});
if isempty(tempS)==0
% Check to see if this thresholding is neccessary
% Find the values in S component of the original image corresponding to
% non zero values in binary image from previous step.
level=(min(tempS)+std(tempS));
% Calculating a threshold based on the minimum and standard deviation
% for the S values computed in the previous step
tempIm1=s.*binaryIm{i,5};
% obtaining an matrix using the binary image with actual s values
% values instead of 1s
```

```
binaryIm{i,6}=im2bw(tempIm1,level);
else binaryIm{i,6}=binaryIm{i,5};
end
% Applying the new thresholding value on the previous matrix to get the
```

```
% final detection of cracks and save it as a binary image
tempIm2=orImage{i};
% Using an temporary matirx to reconstrcuct the image
R=tempIm2(:,:,1);
NR=R;
% Extraction of Red component of the orginal image
G=tempIm2(:,:,2);
NG=G;
% Extraction of Green component of the orginal image
B=tempIm2(:,:,3);
NB=B;
% Extraction of Blue component of the orginal image
NR(binaryIm{i,6})=255;
% Find the cracks location in Red component and make the maximum
NG(binaryIm{i,6})=0;
% Find the cracks location in Green component and make the minimum
NB(binaryIm{i,6})=0;
% Find the cracks location in Blue component and make the minimum
tempIm2(:,:,1)=NR;
% Reconstruct of the Red component of the original picture with new
% values of R
tempIm2(:,:,2)=NG;
% Reconstruct of the Green component of the original picture with new
% values of G
tempIm2(:,:,3)=NB;
% Reconstruct of the Blue component of the original picture with new
% values of B
RecIm{i}=tempIm2;
% Save the recunstructed picture in the cell array, the constructed
% picture shows the detected cracks in 'binaryIm{i,6}' in red on the
% RGB picture
```

```
binaryIm{i,7}=bwmorph( binaryIm{i,6},'skel',inf);
% Use morphological operation to transform the cracks into lines
Cracks_L{i}=size(find(binaryIm{i,7}),1);
value=Cracks_L{i};
RecIm{i}=insertText(RecIm{i},position,value,'AnchorPoint','Lefttop','FontSize',40);
% Find the total crack length in each picture by counting total lines
% and imposing this number to RGB picture
figure
subplot(3,3,1)
imshow(orImage{i})
title('Original RGB Image')
subplot(3,3,2)
imshow( grayIm{i})
title('Grayscale image+applying median filter')
subplot(3,3,3)
imshow(edge{i})
title('Image after applying Sobel')
subplot(3,3,4)
imshow( binaryIm{i,1})
title('Bw using 0.2 Otsu TH value')
subplot(3,3,5)
imshow(binaryIm{i,3})
title('Bw with 200 cutoff area and elimination of V and H lines')
subplot(3,3,6)
imshow( binaryIm{i,5})
title('Bw after morphological opration and 50 cutoff area')
```

```
subplot(3,3,7)
imshow( binaryIm{i,6})
title('Bw eliminated uncracks lines')
subplot(3,3,8)
imshow( binaryIm{i,7})
title('Cracks skeleton')
subplot(3,3,9)
imshow( RecIm{i})
title('Reconstructed RGB image with cracks and pixels with cracks')
```

```
% Showing the results of the code in figures
end
% End of the loop
pause
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% paper approach%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:NF
% Loop start, each i is for one original image
paper_ap{i,1}=rgb2gray(orImage{i});
% grayscale image
paper_ap{i,2}=abs(imfilter(paper_ap{i,1},Hx))+abs(imfilter(paper_ap{i,1},Hy));
% edge detection usuing sobel method
paper_ap{i,3}=im2bw(paper_ap{i,2},graythresh(paper_ap{i,2}));
% Using Otsu's method to change the gray-scale image into binary
paper_stat=regionprops(paper_ap{i,3},'Area');
% Finding the area of each connected component
paper_area=cat(1,paper_stat.Area);
% Saving the area inforamtion in a matrix(vector)
paper_L=bwlabel(paper_ap{i,3});
% Label the connected component
paper_v1=find(area>30);
% Find component with area more than 30
paper_ap{i,4}=ismember(paper_L,paper_v1);
% Remove the components with area less than or equal to 30 from edges
paper_ap{i,5}=bwmorph(paper_ap{i,4},'skel');
% Linearization the cracks
paper_temp=orImage{i};
% Temp iamge to create cracked imposed image
paper_cracksL(i)=size(find(paper_ap{i,5}),1);
% Find the total numebr of crack pixels
paper_R=paper_temp(:,:,1);
paper_G=paper_temp(:,:,2);
paper_B=paper_temp(:,:,3);
% Finding the 3 color component of the temp image
paper_R(paper_ap{i,4})=0;
paper_G(paper_ap{i,4})=0;
paper_B(paper_ap{i,4})=255;
```

```
% Making cracked pixels in the original picture blue
paper_temp(:,:,1)=paper_R;
paper_temp(:,:,2)=paper_G;
paper_temp(:,:,3)=paper_B;
% Reconstruct the new RGB image
paper_ap{i,6}=paper_temp;
% The the temp new RGB reconstructed image as the new output
paper_value=paper_cracksL(i);
% Define the number attached to Final result
paper_ap{i,7}=insertText(paper_ap{i,6},position,paper_value,'AnchorPoint','Lefttop','
FontSize',
40);
```



```
% Define the new picture with number of cracked pixels attached to it
figure
subplot(3,3,1)
imshow(orImage{i})
title('Original RGB Image')
subplot(3,3,2)
imshow( paper_ap{i,1})
title('Grayscale image')
subplot(3,3,3)
imshow(paper_ap{i,2})
title('Image after applying Sobel')
subplot(3,3,4)
imshow(paper_ap{i,3})
title('Bw using Otsu TH value')
subplot(3,3,5)
imshow(paper_ap{i,4})
title('Bw with 30 cutoff area')
subplot(3,3,6)
imshow(paper_ap{i,5})
title('Crack skeleton')
subplot(3,3,7)
imshow( paper_ap{i,6})
title('Reconstructued image')
```

```
subplot(3,3,8)
imshow( paper_ap{i,7})
title('Reconstructed RGB image')
% Showing the result
end
% End of the loop
```