ACI STRUCTURAL JOURNAL TECHNICAL PAPER

Title No. 112-S32

Automated Detection and Measurement of Cracks in Reinforced Concrete Components

by Jonathan P. Rivera, Goran Josipovic, Emma Lejeune, Bismarck N. Luna, and Andrew S. Whittaker

Damage and loss assessment of reinforced concrete elements are based in part on the length, width, and areal density of cracks. Crack information is traditionally collected using a crack width card and transferred to drawing sheets, which is both approximate and labor-intensive. An automated procedure involving digital image processing was developed and deployed to collect and process crack data. The procedure was developed and validated using data from the cyclic testing of nine reinforced concrete shear walls of varying aspect and reinforcement ratios.

Keywords: automated crack detection and measurement; damage assessment; imaging.

INTRODUCTION

The length, width, and areal density of cracks are used for damage assessment of reinforced concrete components in the laboratory and in the field. Damage to reinforced concrete components in buildings and infrastructure after earthquake shaking is often inferred using maximum residual crack widths. Strategies for post-earthquake repair are often based on this information. Cracking of reinforced concrete components during laboratory testing is routinely documented (along with other information) to enable a reconciliation of loss of strength and stiffness with observed damage. Information on the cracking of low aspect-ratio reinforced concrete shear walls[1] has been used to generate fragility functions and consequence functions that enable an estimate of the repair cost per unit area of wall.

Engineers have traditionally identified cracks visually in reinforced concrete components, measuring their widths using a crack-width card and then transferring that data to drawing sheets. Crack width is measured either at user-determined locations along the length of a crack, or a maximum crack width is reported. Cracks are assigned a width equal to one of the marks on the gauge (for example, 0.005 in., 0.016 in., or 0.06 in.; refer to Fig. 1). Widths of cracks between the marks on a gauge can only be estimated. The process is laborious and approximate because uncertainty is introduced in the measurement of the crack width, the use of few measurement locations, and the transfer of information to drawing sheets.

Imaging tools provide a means to improve the process of measuring and documenting cracks and their widths, and to substantially improve the quality and accuracy of the results. Noncontact identification and measurements of cracks enables data to be gathered from large-size laboratory tests at instances of peak loading and deformation in a safer and more reliable manner. Gathering information on the length, width, and areal density of cracks at instances of peak deformation and zero loading enables estimates to be made of structural damage to components, which are better correlated with peak transient crack widths and lengths than residual crack widths and lengths.

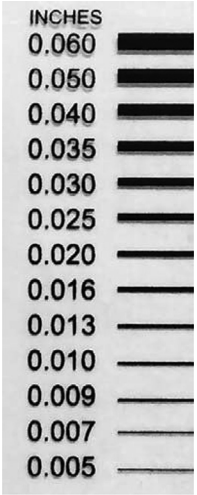


Fig. 1—Crack gauge.[11] (Note: 1 in. = 25.4 mm.)

An automated process for detecting cracks, and measuring their widths and lengths is presented in this paper. Data from the tests of nine large-size, low aspect-ratio reinforced concrete shear walls are used to develop and validate the algorithms and computer codes. The algorithms, source code, and documentation will be uploaded to NEEShub (www.neeshub.org) for use by researchers and forensic engineers.

RESEARCH SIGNIFICANCE

An automated, noncontact procedure is proposed for the estimation of crack width, length, and areal density. The proposed procedure, which could be used in the field and the laboratory, is safer, faster, and more accurate than the traditional method of measuring lengths and widths of cracks by hand using crack gauges and tapes and then transferring that information to drawing sheets.

LITERATURE REVIEW

Image-based crack-detection algorithms have been used primarily to detect cracks on pavement surfaces. Most crack-detection algorithms rely on edge detectors to locate

ACI Structural Journal, V. 112, No. 3, May-June 2015.
MS No. S-2014-084.R1, doi: 10.14359/51687424, received August 18, 2014, and reviewed under Institute publication policies. Pertinent discussion including author's closure, if any, will be published ten months from this journal's date if the discussion is received within four months of the paper's print publication.

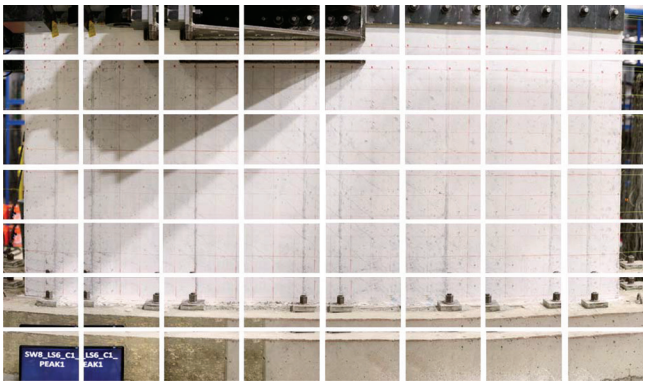


Fig. 2—Raw images of SW8, LS6.

cracks. Abdel-Qader et al.[2] investigated the utility of the Sobel and Canny edge detectors and the Fast Fourier and Fast Haar algorithms to identify cracks within a grayscale (intensity) image. The Sobel and Canny edge detectors locate edges based on the gradient of an image. The Fast Fourier and Fast Haar algorithms detect edges based on brightness modulation within an image. Each algorithm was applied to fifty 640 x 480 pixel images to determine if the algorithm could correctly identify the presence of a crack. They concluded the Fast Haar algorithm had the highest rate of correct identifications and the Fast Fourier algorithm had the lowest.

The algorithms investigated by Abdel-Qader et al.[2] also detected surface defects such as divots, blemishes, and stains, which are present on any concrete component. Fujita et al.[3] addressed the presence of surface defects using two preprocessing techniques: 1) image subtraction; and 2) line emphasis filtering. Image subtraction was similar to matrix subtraction because the pixel value (a number that describes the pixel brightness) of one image was subtracted from the pixel value of another image. Image subtraction removed minor irregularities (for example, shading and some lighting irregularities) and more clearly defined the cracks. A line emphasis filter was then used to better define edges and separate surface defects from the cracks. These two preprocessing steps were applied to 50 images of concrete surfaces with irregular lighting and blemishes. Fujita et al.[3] concluded that these two preprocessing steps were effective.

Although edge detection and noise removal were necessary steps in identifying cracks on concrete surfaces using image processing, they did not provide sufficient information for damage assessment. Miyamoto et al.[4] developed a method that would provide information for use in damage assessment by measuring crack widths. The method determined crack widths by using the difference in pixel brightness between cracks and the surrounding surfaces. This method involved the detection of cracks, approximating the cracks as a straight line, applying calibration factors, and then detecting the difference in pixel brightness. They evaluated this method using multiple images and manually measured crack widths. Miyamoto et al.[4] showed that this method compared reasonably well to the measurements taken manually.

Yamaguchi et al.[5] processed images using a percolation model for crack detection. The process was applied to fifty 480 x 480 pixel images and to a single image of 3040 x 2008 pixels. Yamaguchi et al.[5] concluded that this method could satisfactorily detect a crack in an image.

Choudhary and Dey[6] used fuzzy logic and neural network models to detect cracks. RGB images were converted to grayscale images and a Sobel edge detection algorithm was used to locate the edges of the cracks and the surface defects. Morphological operations were then used to fill the space between the detected edges and to remove some of the surface defects. Two parameters were then extracted from the cracks or surface defects (area and major-minor axes length ratio) using subroutines available in MATLAB.[7] These parameters were then entered into the neural network model or fuzzy logic model where the object was classified as a crack or not a crack. The neural network model provided better results than the fuzzy logic model.

For a noncontact damage assessment algorithm to be useful in the field or for large-scale testing, it must be able to: 1) identify cracks in concrete components; 2) accurately monitor crack width and propagation under changing amplitudes of loading; 3) enable measurement of crack width and length for the purpose of establishing damage and repair measures; 4) be capable of handling high-resolution images of an entire specimen; and 5) remove image irregularities due to shadows and lighting. None of the algorithms previously described can accomplish all of these goals.

EXPERIMENTAL SETUP OF DIGITAL HARDWARE

High-resolution images must be used for automated detection and measurement of cracks because cracks represent a very small fraction of the concrete surface. A digital camera and a robotic panohead were used to capture high-resolution images of nine 10 ft (3.05 m) long reinforced concrete shear walls with heights up to 10 ft (3.05 m). The camera used had a resolution of 18 megapixels and a maximum image size of 5200 x 3462 pixels. This camera could not photograph an entire wall in sufficient detail and was therefore mounted on a robotic panohead to enable the assembly of a gridded set of high-resolution images at a given load step. This combination of the high-megapixel camera and robotic panohead allows cracks as narrow as 0.01 in. (0.25 mm) to be detected by I-Crack (likely sufficient for most applications) and corresponds to 200 pixels per inch of concrete surveyed. (The detection of even narrower cracks will require a greater number of pixels per inch.)

Twelve walls, labeled SW1 through SW12,[8] were tested using a pseudo-static, reversed cyclic loading protocol with increasing levels of displacement. Each increase in displacement corresponded to a load step (LS). Figure 2 is a set of raw images of SW8 at LS6. Each image in the set was aligned and blended using the GigaPan Stitch[9] software to create a single, high-resolution panorama. Figure 3 is the stitched panorama of Fig. 2.

All twelve walls were tested with multiple cycles of displacements less than and greater than the displacement associated with peak strength. Damage in the form of spalled concrete, crack initiation, propagation and width, and reinforcing bar buckling was monitored at each displacement increment. Cracks were identified by: 1) visual inspection

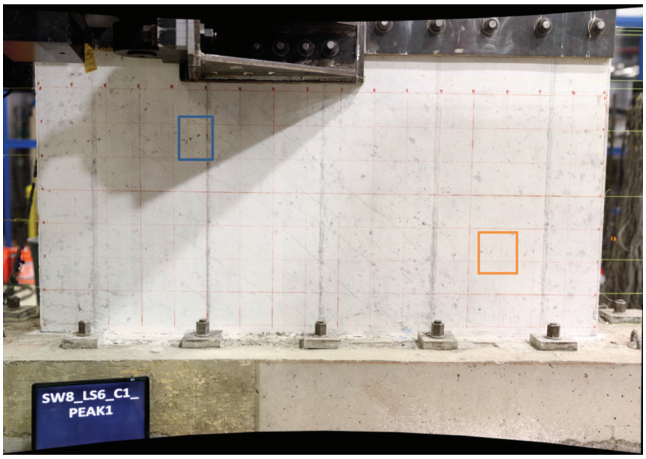


Fig. 3—Assembled high-resolution panorama of SW8, LS6.

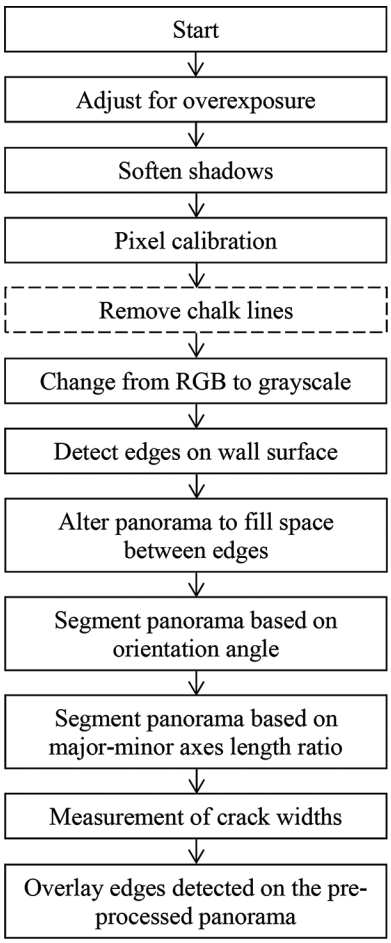
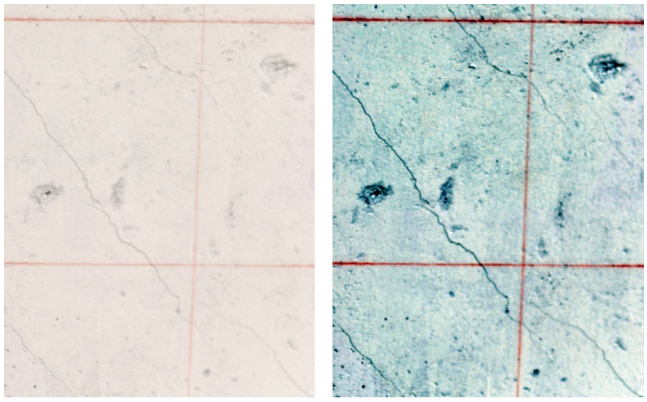


Fig. 4—Flowchart for I-Crack.

(with crack widths measured by hand); and 2) using an automated process based on digital imaging and processing, described in the following sections.

High-resolution panoramas and crack width data were collected for nine of the twelve walls at every peak transient displacement. Data from SW8 were used to develop the algorithm and data from eight other walls were used to validate the algorithm. Of the eight walls used to validate I-Crack, results are presented herein for SW2 and SW7; results for the remaining six walls will be available in the



(a) overexposed image (b) image after preprocessing

Fig. 5—Correction for overexposure, SW8, LS6.

dissertation of the lead author. Walls SW2 and SW8 were 65 in. (1.65 m) in height from the top of the foundation to the centerline of horizontal loading. The vertical and horizontal reinforcement ratios for SW2 and SW8 were 1.0% and 1.5%, respectively. The day-of-test concrete compressive strength was 7000 psi (48.3 MPa) for SW2 and 3500 psi (24.1 MPa) for SW8. The height of SW7 was 41 in. (1.04 m); the compressive strength of its concrete was 3800 psi (26.2 MPa) on the day of testing. The horizontal and vertical reinforcement ratios in SW7 were 0.33%.

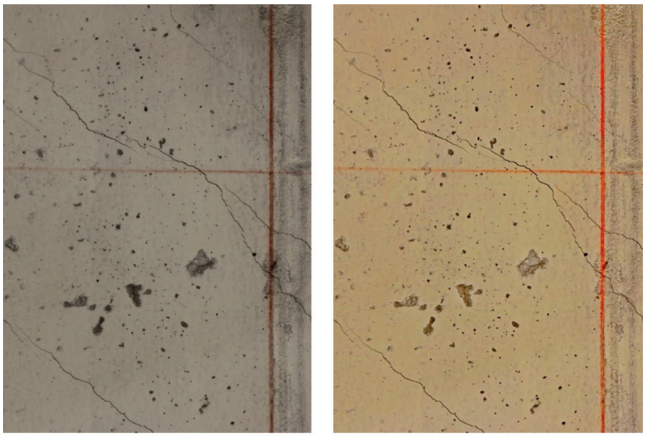
AUTOMATIC DETECTION AND MEASUREMENT OF CRACKS

I-Crack detects and measures the length and varying width of cracks in reinforced concrete components using high-resolution panoramas. I-Crack is constructed around algorithms available in MATLAB.[7] The building blocks of I-Crack are: 1) an edge detection algorithm; 2) morphological operations that fill cracks; 3) operations that distinguish and separate cracks from surface defects; and 4) measurement of crack widths. Each step is shown in Fig. 4 and described as follows to enable coding by others.

Image preprocessing and pixel length calibration

Image preprocessing precedes the implementation of I-Crack and is unique for each panorama analyzed. The need for preprocessing is highly dependent on the quality of the panorama, which is determined in part by the environment where it was captured. Ideally, the component being surveyed will be uniformly lit to avoid shadows and this should be possible for many field applications of I-Crack. The laboratory setting was less than ideal because the testing equipment cast shadows and the ambient lighting varied over the duration of a test.

Initial attempts to process the panoramas from SW8 identified two hurdles to I-Crack processing: 1) overexposure of images, and 2) shadows on the surface of the specimen. Overexposure produces areas within a panorama that are extremely light compared to the remainder of the image. A crack propagating through an overexposed part of a panorama is difficult to detect, as highlighted in the open orange box of Fig. 3 and enlarged in Fig. 5(a). Image-processing software such as FastStone Image Viewer[10]



(a) shadows present (b) image after pre-processing

Fig. 6—Correction for shadows, SW8, LS6.

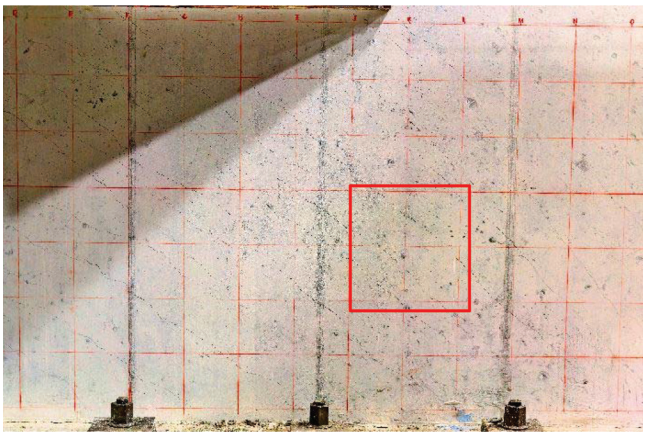


Fig. 7—Preprocessed panorama of SW8, LS6.

can be used to adjust a panorama for overexposure. The "Auto-Adjust Colors" option the in FastStone Image Viewer is used herein to reduce overexposure in the panoramas, with results shown in Fig. 5(b).

After a panorama is processed for overexposure, shadows present on the wall should be softened to improve crack detection. An example is enclosed in the open blue box in Fig. 3 and enlarged in Fig. 6(a). Shadow softening is preferred to shadow removal because the image manipulation required to completely remove a shadow will overexpose those areas where no shadows are present. The "Adjust Lighting" option in the FastStone Image Viewer is used to soften the shadows, as seen in panel Fig. 6(b). (A component must be evenly lit if no shadows are permissible.)

A completely preprocessed panorama of SW8 is shown in Fig. 7. The area enclosed in the open red box of Fig. 7 was used to develop the algorithm that is discussed in the following sections.

A calibration factor is calculated from the preprocessed panorama. The factor is required because I-Crack initially predicts crack properties based on the number of pixels within a feature (crack or surface defect). To obtain a useable measurement (in. or in.^{2}), a measurement in pixels is multiplied by the corresponding calibration factor. The factor can be determined several ways and is unique for each specimen

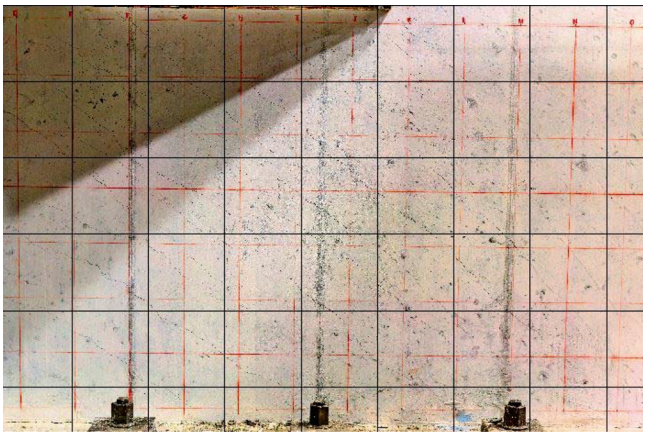


Fig. 8—SW8, LS6 panorama divided into blocks.

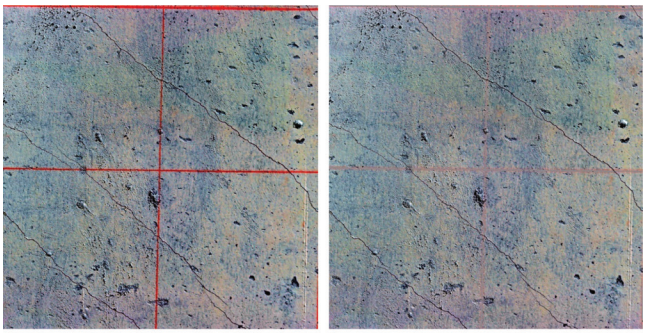
imaged. Within each panorama an object must be present that has a known length or area. Using image segmentation, that object is changed to white and the rest of the panorama is changed to black. MATLAB interprets a black-and-white image as a matrix full of zeroes and ones, which are black and white, respectively. The length can then be determined by summing along the row or column of the matrix; the area is the sum of the elements in the matrix. The calibration factor is determined by dividing the object's length or area by the number of pixels.

The specimens used to develop I-Crack were rectangular walls with a known geometry. The wall is used as the object in the panorama to determine the calibration factor. The panorama is loaded into MATLAB and changed to a black-and-white image using the function *im2bw*. The wall is changed to white and the background to black, thus locating the wall in the panorama. The number of pixels along the length of the wall is determined by summing along one of the rows. The calibration factor is determined by dividing the length of the wall (in inches or centimeters) by the number of pixels. For field applications, a fine scale ruler could be attached to the specimen being surveyed for the purpose of establishing a calibration factor.

Block processing

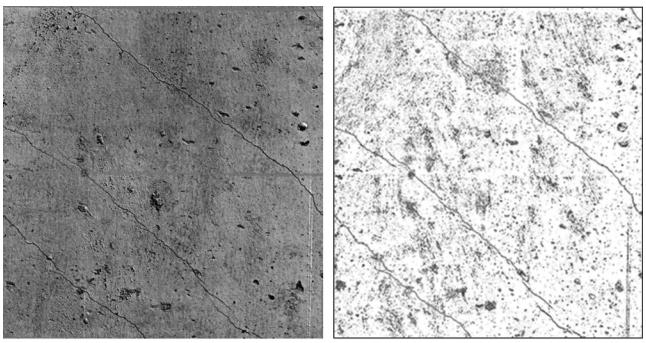
I-Crack is employed once the panorama has been preprocessed and a calibration factor has been determined. The panorama is loaded into MATLAB using a RGB color scheme, which is stored as a set of three matrixes. Each matrix is used to define the presence of red, blue, and green in each pixel. The dimensions of each matrix are equal to the number of horizontal and vertical pixels in the panorama (for example, a panorama with an image size of 24,000 x 6000 pixels will require three matrixes with 24,000 rows and 6000 columns). High-resolution panoramas require significant computational resources for processing.

MATLAB provides block processing (*blockproc*) to execute scripts and decrease processing time for larger images. Block processing divides the panorama into smaller sub-images (blocks), which are individually processed by the script. The sub-image sizes used in I-Crack are 500 x 500 pixels or 2000 x 2000 pixels (Fig. 8), resulting in 1130 or 70 sub-images for SW8, respectively. These sub-image



(a) chalk lines present (b) chalk lines replaced by averaged pixel

Fig. 9—Removal of chalk lines on SW8, LS6.



(a) grayscale input image (b) Prewitt edge detection algorithm

Fig. 10—Edge detection algorithm applied to SW8, LS6.

sizes were selected based on outputs obtained during the development of I-Crack and can be chosen by the user.

Gridline removal

Both faces of every wall were marked with a grid of red chalk lines (Fig. 7 and Fig. 9(a)) to facilitate the location and documentation of cracks. Although the chalk lines are useful for visual inspection of the walls, they are problematic for I-Crack analysis because a pixel gradient is present at each chalk line, causing I-Crack to detect and record many unnecessary edges.

Removal of these unnecessary edges is accomplished with a script written to replace the chalk lines with a pixel similar to the surrounding surface: an *averaged* pixel. The script uses a thresholding technique based on the contribution of red to each pixel. If the contribution of red in a pixel reaches the threshold, that pixel is replaced with an *averaged* pixel. By replacing the pixel with an *averaged* pixel, the gradient between the surface and the chalk line is reduced. The edge detection algorithm should not detect the edges of the chalk line after these pixels are replaced (refer to Fig. 9(b)). Gridline removal is not needed if gridlines are not marked on a specimen.

Edge detection

I-Crack uses *rgb2gray* to change the panorama from a RGB color scheme to grayscale to enable edge detection. All of the edge detection algorithms available in MATLAB (Roberts, Sobel, Prewitt, Canny, and Laplacian of Gaussian) require a grayscale image as input.

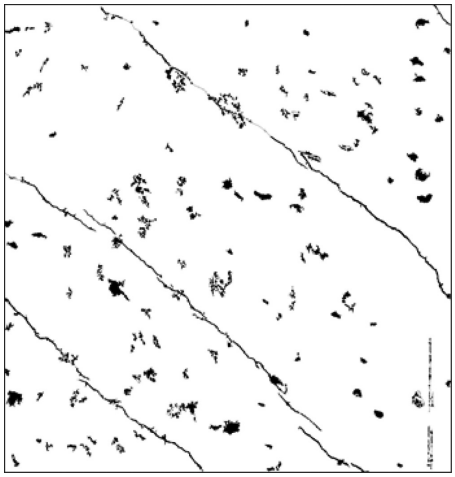


Fig. 11—Features filled after applying morphological operations, SW8, LS6.

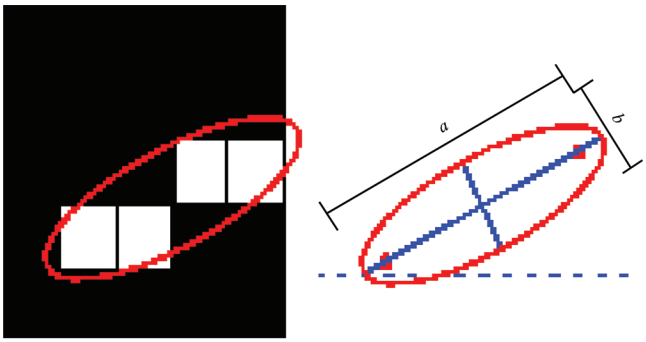
The Roberts, Sobel, Prewitt, Canny, and Laplacian of Gaussian edge detectors were each applied to panel Fig. 10(a) to determine the algorithm that best identified the edges of cracks and surface defects. The Roberts, Sobel, and Prewitt algorithms detected the edges of the features in Fig. 10(a), but the Roberts and Sobel algorithms recorded additional edges associated with the shading present on the wall surface. The Prewitt algorithm detected the edges of the cracks and the surface defects but fewer edges associated with shading. Both the Canny and Laplacian of Gaussian algorithms detected spurious edges. Accordingly, the Prewitt edge-detection algorithm is implemented in I-Crack, although the code is written to accommodate any of the MATLAB options. The result of applying the Prewitt algorithm to Fig. 10(a) is shown in Fig. 10(b).

Morphological operations

After edge detection, the panorama is manipulated to fill the space between the detected edges for image segmentation and measurement. MATLAB provides several morphological operations to accomplish this task: 1) *bridge:* changes black pixels to white if there are two neighboring white pixels; 2) *fill*: changes isolated black pixels to white; 3) *close:* changes black pixels to white pixels based on a matrix of ones and zeroes; 4) *bwareaopen:* removes connected objects in a black-and-white image with pixels less than a set threshold value; 5) *imfill:* fills in a region of the image based on the connectivity of the pixels within that region; and 6) *imclose:* dilates and then erodes the image. Figure 11 shows the cracks and surface defects of Fig. 10(b) after all of these morphological operations have been applied.

Image segmentation

Cracks and surfaces defects are separated after the morphological operations. The process of separating cracks and surface defects is accomplished through image segmentation: the separation of objects based on their properties (for example, color, area, and perimeter). I-Crack segments the image using scripts based on two criteria: 1) orientation angle, and 2) the major-minor axes length ratio. Both criteria



a. generic image b. major and minor axes lengths labeled

Fig. 12—Parameter extraction.

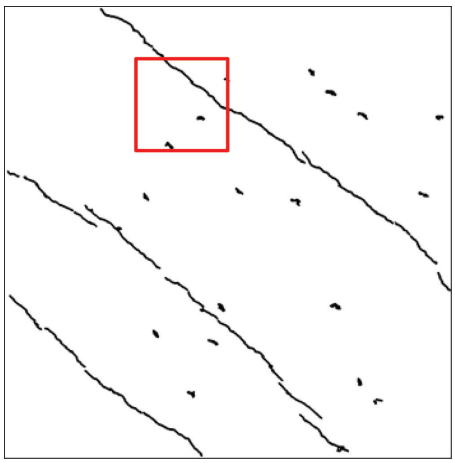


Fig. 13—Cracks after image segmentation, SW8, LS6.

are determined using the *regionprops* routine in MATLAB, which measures the properties of image features such as area, centroid, orientation angle, and perimeter.

The first script used to segment the panorama is based on the orientation angle of each crack or surface defect. Orientation angle is selected as a criterion because many cracks are inclined, whereas surface defects are often horizontal or vertical. MATLAB determines the orientation angle of the crack or surface defect based on the "…angle between the x-axis and the major axis of an ellipse that has the same second moment as the region." Figure 12 illustrates this process. Figure 12(a) shows four white pixels enclosed by an ellipse; both the four white pixels and the ellipse have the same second moment. The second moment for a two-dimensional dataset is the covariance matrix. The covariance matrix fits a multivariate normal distribution to the region and takes the shape of an ellipse. The eigenvalues of the covariance matrix are the lengths of the major and minor axes of that ellipse. Figure 12(b) shows the ellipse; lines *a* and *b* are the major and minor axes of the ellipse, respectively. The orientation angle is defined as the angle between the horizontal and line *a*.

The second script used to segment the panorama is based on the ratio of the major axis length to the minor axis length. This ratio is selected as a criterion for image segmentation based on observations made during the development of the script. Cracks have a much longer major axis (length) than minor axis (width), resulting in a ratio much greater than 1.

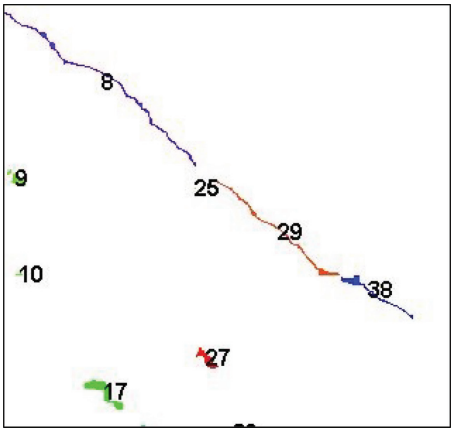


Fig. 14—Crack width labels of boxed region in Fig. 13, SW8, LS6.

Surface defects have a ratio much closer to 1 because many are circular or ellipsoidal in shape.

Measurement of crack width and length

The properties of each crack are determined after the panorama is segmented (Fig. 13). Crack widths are not measured directly in I-Crack. Rather, the MATLAB subroutine *regionprops* is used, which measures area by counting the number of pixels within a crack or surface defect. Because only the area can be measured using *regionprops,* two area measurements are required: 1) the area of the crack, and 2) the area of the crack when it has been altered to have a unit width (that is, length).

The area of a crack is calculated by calling *regionprops* and measuring area. The area of each crack is then stored in an array for subsequent use. The entire panorama is then altered using the morphological operator *skel*, which "…removes pixels on the boundaries of objects but does not allow objects to break apart." The product is a panorama of cracks that have unit width and are located on the centerline of the detected areas. *Regionprops* is then called to measure the length of the centerline of the areas detected in the altered panorama. The width of a crack is determined by dividing its area by its centerline length, and therefore is an averaged value over the centerline length. Each crack is assigned a label to allow the user to extract width and area. As an example, Fig. 14 identifies the cracks in the open red square of Fig. 13 that are detected and labeled by I-Crack. The square has a side dimension of 3 in. (76 mm). The cracks are color coded, based on their width, to enable further processing. The coding scheme could be based on the method of repair associated with the width of crack. Blemishes (for example, numbers 9, 10, 17, and 27 in Fig. 14) would not be included by the user in the calculation of crack lengths. All of the cracks in Fig. 15 are in one color (purple) because the authors chose two ranges for post-processing crack width w— $0.02 \leq w \leq 0.125$ in., and $w > 0.125$ in.—but no crack had a width greater than 0.125 in.

The length of a crack is measured in I-Crack using the *imdistline* tool in the MATLAB Image Processing Toolbox, which provides the user with a "…draggable, resizable line" that is superimposed on an image to measure the distance

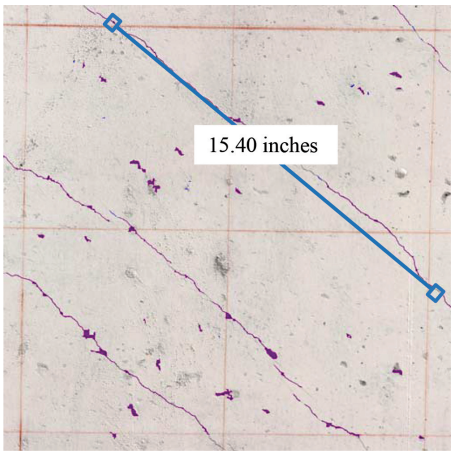


Fig. 15—Measuring crack length and color coding cracks, SW8, LS6.

between two endpoints. Figure 15 shows a line generated with *imdistline* superimposed on the wall surface. The endpoints (open blue boxes in Fig. 15) are selected by the user and could correspond to: 1) initiation of a crack; or 2) a crack of a given width (for example, 0.02 in. [0.51 mm]) or greater.

VALIDATION OF I-CRACK

I-Crack was evaluated using crack patterns and crack widths collected manually from the tests of nine walls, starting with SW2 and SW7. Results of the validation exercise for eight walls (SW1, SW2, SW3, SW5, SW7, SW9, and SW12) will be presented in the lead author's dissertation and are summarized as follows for two of the eight walls: SW2 and SW7.

I-Crack was first evaluated to ensure that cracks were properly being detected by the algorithm. Figure 16 shows SW2 at Load Step LS10 after pre-processing. Figure 17 shows the cracks detected by I-Crack. A composite of Fig. 16 and Fig. 17 is presented in Fig. 18. The areas enclosed in the open red and blue boxes are enlarged in Fig. 19, which shows that I-Crack can detect cracks accurately.

Crack widths predicted by I-Crack are compared to data collected manually at the locations denoted by the letters in Fig. 18 for SW2. Results at peak positive displacements in Load Steps LS10 and LS11 are presented in Table 1. The peak displacement in LS10 was 0.82 in. (21 mm) (story drift angle of 1.26%) and is the displacement associated with peak shearing strength. The peak displacement in LS11 was 1.45 in. (37 mm) (story drift angle of 2.23%) and greater than the displacement associated with peak shearing strength. The manually measured and I-Crack results are comparable at both displacements, noting that the variability in manually measuring and locating cracks is unknown and the intervals of the crack gauge are not uniform and range from 0.005 to 0.0625 in. (0.13 to 1.6 mm). For LS10, the greatest difference in crack width is 0.047 in. (1.2 mm) at Crack J, which is less than the interval of the crack gauge for this width (0.0625 in. [1.6 mm]). For LS11, the greatest difference in crack width is 0.043 in. (1.1 mm), which also occurs on Crack J.

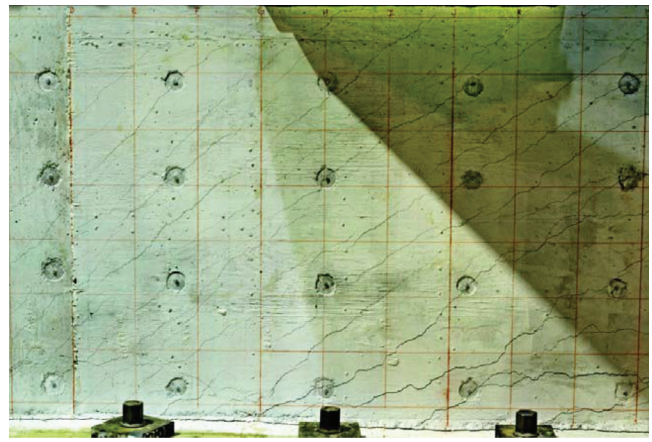


Fig. 16—Preprocessed image of SW2, LS10.

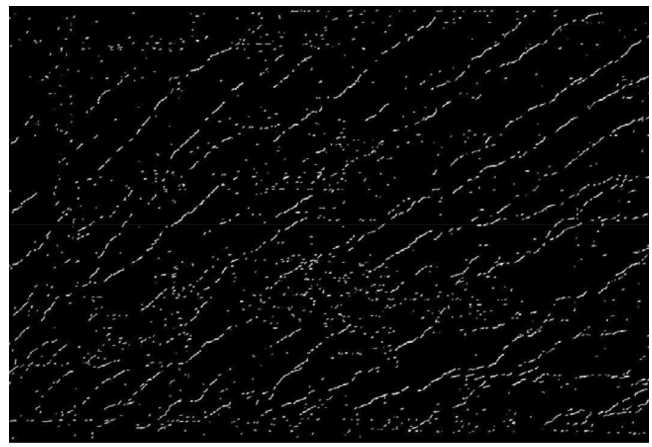


Fig. 17—Predicted crack locations for SW2, LS10.

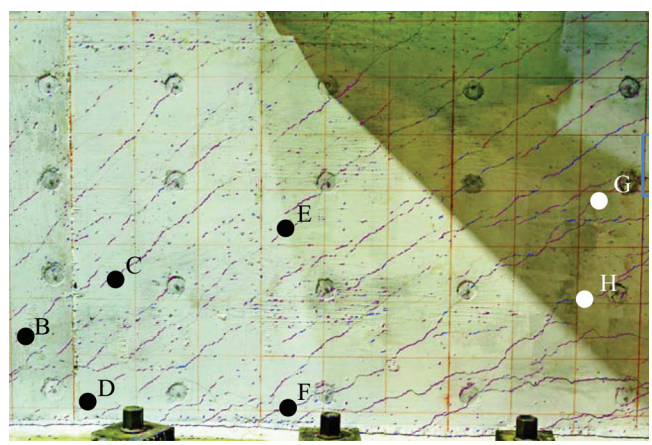
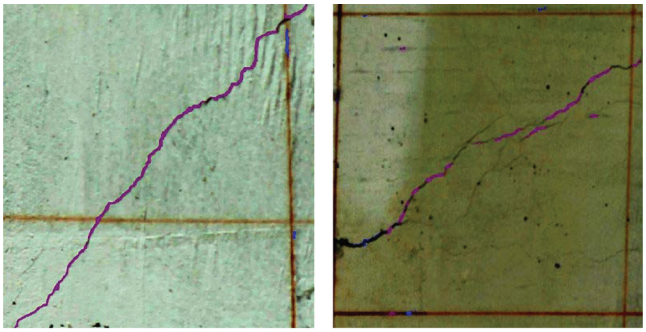


Fig. 18—Composite image of SW2, LS10.

Figure 20 identifies the locations, denoted by letters, where crack data were collected manually for SW7 and compared with the crack widths predicted by I-Crack. Table 2 presents results for peak positive displacements in Load Steps LS3 and LS10. The peak displacement in LS3 was 0.05 in. (13 mm) (story drift angle of 0.12%) and was smaller than the displacement associated with peak shearing strength in SW7. The peak displacement in LS10 was 0.34 in. (8.6 mm) (story drift angle of 0.85%) and was greater than the displacement associated with peak shearing strength. The results are comparable at both peak displacements considering the



a. red box b. blue box

Fig. 19—Close-up of boxed regions in Fig. 18, SW2, LS10.

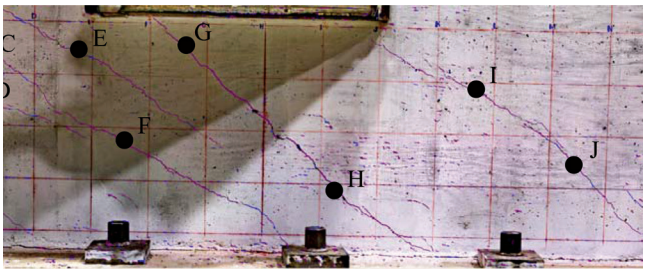


Fig. 20—Composite image of SW7, LS10.

Table 1—Crack width data for SW2

	Measurement, in.			
	LS10		LS11	
Crack ID in Fig. 18	Manual	I-Crack	Manual	I-Crack
A	0.020	0.027	0.040	0.040
B	0.016	0.024	0.016	0.017
C	0.016	0.033	0.020	0.024
D	0.025	0.031	0.025	0.029
E	0.020	0.038	0.020	0.030
F	0.050	0.039	0.050	0.051
G	0.016	0.019	0.016	0.027
H	0.020	0.023	0.020	0.032
I	0.020	0.033	0.020	0.026
J	0.125	0.078	0.125	0.082

Note: 1 in. = 25.4 mm.

uncertainty associated with measuring crack widths using a gauge with discrete widths. The largest difference between manual and I-Crack measurements occurs on Crack D in LS10; 0.019 in. (0.48 mm), which is insignificant.

LENGTH AND AREAL DENSITY OF CRACKS

Damage to reinforced concrete components can be measured in terms of peak crack width (a traditional measure reported by researchers), length of crack with a width greater than a user-specified limit (often tied to a method of repair; refer to Gulec et al.[1]), and areal density of cracks (for example, length of crack with a width greater than a threshold divided by the area of the component being surveyed). I-Crack provides this information, as illustrated in the following example.

Table 2—Crack width data for SW7

	Measurement, in.			
	LS3		LS10	
Crack ID in Fig. 20	Manual	I-Crack	Manual	I-Crack
A	NC*	NC	0.010	0.028
B	0.009	0.014	0.050	0.047
C	NC	NC	0.016	0.027
D	NC	NC	0.010	0.029
E	NC	NC	0.025	0.033
F	NC	NC	0.040	0.040
G	0.025	0.031	0.040	0.040
H	0.020	0.024	0.063	0.052
I	NC	NC	0.025	0.029
J	NC	NC	0.040	0.044

*NC is no crack present.

Note: 1 in. = 25.4 mm.

Table 3—Crack length and areal density data for SW2 and SW7

Wall	Load step	Crack width, w	Crack length, in.	Areal density, in/ft^2
SW2	LS10	0.02 in. $\leq w <$ 0.125 in.	1170	21.6
		$w \geq$ 0.125 in.	7	0.13
	LS11	0.02 in. $\leq w <$ 0.125 in.	1464	27.0
		$w \geq$ 0.125 in.	7	0.13
SW7	LS3	0.02 in. $\leq w <$ 0.125 in.	57	1.67
		$w \geq$ 0.125 in.	0	0.00
	LS10	0.02 in. $\leq w <$ 0.125 in.	222	6.50
		$w \geq$ 0.125 in.	0	0.00

Notes: 1 in. = 25.4 mm; 1 in./ft^2 = 273.4 mm/m^2.

The user-specified limits for the example are crack widths of 0.02 and 0.125 in. (0.51 and 3.2 mm). Results are presented in Table 3 for crack length and areal density of cracks for two ranges of crack width: $w > 0.125$ in. and $0.02 \leq w \leq 0.125$ in. These data for SW2 and SW7 could be used to characterize damage and estimate repair costs, noting that field calculations will employ residual rather than peak transient data.

SUMMARY AND CONCLUSIONS

An automated, non-contact procedure (I-Crack) to measure and document crack widths in reinforced concrete components was developed, deployed, and validated. I-Crack can replace traditional methods of collecting such data with crack gauges, which are labor-intensive and approximate.

I-Crack was developed and validated using data from the cyclic tests of nine low aspect-ratio reinforced concrete shear walls in the NEES facility at the University at Buffalo. The processing techniques and algorithms were developed

using data from one wall and validated using data from the remaining eight walls.

I-Crack requires preprocessing of images to reduce overexposure and shadows in the high-resolution panoramas. The image resolution required to detect cracks as narrow as 0.01 in. is 200 pixels per inch of concrete surveyed. Once preprocessing is completed the panorama is loaded into I-Crack, where an edge-detection algorithm and morphological operations are used to detect cracks and surface defects. The Prewitt algorithm and morphological operations, as implemented in MATLAB, are recommended for use. I-Crack uses image segmentation to separate the cracks from surface defects. The two parameters used for image segmentation are the major-minor axis ratio and orientation angle. Crack widths are measured using the MATLAB *regionprops* subroutine after the panorama is segmented. The lengths of individual cracks are measured using the MATLAB *imdistline* subroutine.

The utility of I-Crack to detect and measure crack widths was evaluated using data collected a) using traditional crack gages, and b) transferred manually to drawing sheets. Crack locations and widths were recorded at every load step for the nine walls. I-Crack results were compared to manually recorded values at discrete locations on every wall at two load steps: one at a peak displacement less than that associated with peak strength and one at a peak displacement substantially greater than that associated with peak strength. This choice of load steps enabled a comparison of results for narrow and wide cracks. I-Crack recovered the widths of cracks in the eight shear walls used to validate the algorithm well and this provides confidence that the I-Crack algorithm is robust.

The MATLAB code to implement I-Crack and the required image pre-processing is available at NEEShub (www.neeshub.org).

AUTHOR BIOS

ACI member **Jonathan P. Rivera** *is a PhD Candidate at the University at Buffalo, State University of New York (SUNY), Buffalo, NY. He received his BS in 2011 and MS in 2013 from the University at Buffalo, SUNY. His research interests include reinforced concrete structures and earthquake engineering.*

Goran Josipovic *is the Information Technology Manager for the Network for Earthquake Engineering Simulation at the University at Buffalo, SUNY. He received his BS in computer engineering in 2003 from the University at Buffalo, SUNY. His research interests include digital image processing and large data archival and processing.*

Emma Lejeune *is a Graduate Student at Stanford University, Stanford, CA. She participated in the NEES Research Experience for Undergraduates in 2012 at the University at Buffalo, SUNY. She received her BS in civil engineering from Cornell University, Ithaca, NY, in 2013. Her research interests include solid mechanics, computational mechanics, and mechanics of materials.*

ACI member **Bismarck N. Luna** *is a PhD Candidate at the University at Buffalo, SUNY. He received his BS from the University of the Philippines, Los Baños, Philippines, and his MS from Purdue University, West Lafayette, IN. His research interests include reinforced concrete structures and earthquake engineering.*

Andrew S. Whittaker, *FACI, is a Professor at the University at Buffalo, SUNY. He received his BS from the University of Melbourne, Victoria, Australia, and his MS and PhD from the University of California, Berkeley, Berkeley, CA, in 1977, 1985, and 1988, respectively. He is a member of ACI Committee 349, Code Requirements for Nuclear Safety-Related Concrete Structures. His research interests include earthquake and blast engineering and performance-based seismic design and assessment.*

REFERENCES

1. Gulec, C. K.; Whittaker, A. S.; and Hooper, J. D., "Fragility Functions for Low Aspect Ratio Reinforced Concrete Walls," *Engineering Structures*, V. 32, No. 9, Nov. 2010, pp. 2894-2901. doi: 10.1016/j.engstruct.2010.05.008

2. Abdel-Qader, I.; Abudayyeh, O.; and Kelly, M., "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *Journal of Computing in Civil Engineering*, ASCE, V. 17, No. 4, 2003, pp. 255-263. doi: 10.1061/(ASCE)0887-3801(2003)17:4(255)

3. Fujita, Y.; Mitani, Y.; and Hamamoto, Y., "A Method for Crack Detection on a Concrete Structure," *Proceedings of the 18th International Conference on Pattern Recognition*, Aug. 2006, pp. 901-904.

4. Miyamoto, A.; Konno, M. A.; and Brühwiler, E., "Automatic Crack Recognition System for Concrete Structures Using Image Processing Approach," *Asian Journal of Information Technology*, V. 6, No. 5, 2007, pp. 553-561.

5. Yamaguchi, T.; Nakamura, S.; Saegusa, R.; and Hashimoto, S., "Image-Based Crack Detection for Real Concrete Surfaces," *IEEJ Transactions on Electrical and Electronic Engineering*, V. 3, No. 1, 2008, pp. 128-135. doi: 10.1002/tee.20244

6. Choudhary, G. K., and Dey, S., "Crack Detection in Concrete Surfaces using Image Processing, Fuzzy Logic, and Neural Networks," *IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, Oct. 2012, pp. 404-411.

7. Mathworks, MATLAB version 8.1.0.604 (R2013a), 2013, Natick, MA.

8. Luna, B. N.; Rivera, J. P.; Rocks, J. F.; Goksu, C.; and Whittaker, A. S., "Seismic Performance of Low Aspect Ratio Reinforced Concrete Shear Walls," *Proceedings of the 22nd International Conference on Structural Mechanics in Reactor Technology*, San Francisco, CA, Aug. 2013.

9. GigaPan Systems, "GigaPan Stitch.EFX," GigaPan Systems, Portland, OR, 2013, http://www.gigapan.com/cms/shop/software/gigapan-stitch-efx. (last accessed April 30, 2015)

10. FastStone Soft, FastStone Image Viewer for Windows version 4.7, 2013, http://www.faststone.org/FSViewerDetail.htm. (last accessed April 30, 2015)

11. CTLGroup, "Crack Comparator Card," CTLGroup, Skokie, IL.

NOTES: