Automated Crack Detection and Growth Monitoring: An Integrated SOM-KAZE Approach for Structural Health Monitoring

Xinxin Sun* (ORCID: 0000-0002-9963-2109) and Peter Chang

Department of Civil and Environmental Engineering, University of Maryland, College Park, MD 20742, USA

*Corresponding Author

Email: xinxin68@umd.edu

Abstract

This paper presents a novel approach for automated crack detection and growth monitoring in concrete structures, integrating Self-Organizing Maps (SOM) for pixel-level clustering with a KAZE feature descriptor for robust feature matching. The proposed method first employs SOM to cluster pixels in an initial crack image and derive a representative weight vector. Critically, this weight vector can be directly applied to subsequent images, thereby enabling the detection of crack growth without the need to retrain or rely on multiple reference images. Following SOM classification, noise is minimized through labeling, dilation, and superimposition, ensuring that thin or small crack portions are retained. Finally, the KAZE algorithm is employed to match features in different images, facilitating accurate crack localization and growth tracking across varying scales, rotations, and viewpoints. Experimental results demonstrate that the proposed method accurately detects crack evolution with minimal user intervention, offering a scalable and efficient solution for structural health monitoring.

Keywords: Structural Health Monitoring (SHM), Self-Organizing Map (SOM), KAZE Features, Crack Growth Detection, Damage Evolution

Statements and Declarations:

Competing Interests: The authors declare no competing interests.

1. Introduction

Crack formation represents an early and critical indicator of potential structural deterioration in civil infrastructure, including roads, bridges, tunnels, and buildings [1, 2]. When left undetected or unmanaged, these localized defects can grow and interact with other damage mechanisms, ultimately undermining load-bearing capacity and structural integrity [3, 4]. Ensuring timely detection and continuous monitoring of cracks is therefore fundamental for maintaining safety, extending service life, and mitigating the economic consequences associated with major repair interventions or catastrophic failures [5–7].

Traditionally, crack detection has relied heavily on manual inspections, where experienced specialists visually assess structural surfaces and use simple tools to measure crack dimensions [8–10]. Although this approach has been long established, it is inherently time-consuming, subjective, and increasingly impractical given the complexity and scale of modern infrastructure [1, 11]. To address these limitations, research has progressively shifted toward automated methods, leveraging advanced sensing technologies and image processing techniques [12–14]. Early image-based approaches, however, frequently struggled with challenges posed by noise, shadows, variable illumination, irregular crack morphology, and perspective distortions, making consistent and accurate crack detection a persistent obstacle [14–17].

Recent developments in machine learning (ML) and deep learning (DL) have significantly improved crack detection capabilities by automating feature extraction, reducing reliance on subjective human judgment, and enhancing scalability [18–21]. Nonetheless, many of these advanced models still focus on image-level or bounding-box analysis, limiting their ability to capture subtle crack characteristics and intricate growth patterns that are crucial for comprehensive structural assessments [22, 23]. In addition, model transferability and robustness across varying environmental conditions, structural materials, and geometric complexities remain key challenges [1, 24, 25].

Addressing these gaps requires methods capable of pixel-level analysis that can adapt to diverse conditions without extensive retraining, as well as strategies that effectively mitigate noise and perspective distortions. The integration of unsupervised learning techniques such as Self-Organizing Maps (SOM) has shown promise in clustering and classifying crack features with minimal manual parameter tuning, while advanced feature descriptors can capture fine-grained details at multiple scales and orientations [26–28]. Leveraging such techniques holds the potential to enhance crack growth detection, improve localization accuracy, and reduce computational overhead.

Building on these insights, this paper proposes a novel framework that integrates SOM-based pixel-level clustering with robust feature descriptors to detect and track crack evolution. By training SOM once and reusing its weight vector across successive images, the method eliminates the need for large training sets and repeated model adjustments. Morphological operations and labeling are employed to address noise and minor feature losses, while the use of multi-scale feature descriptors facilitates accurate crack identification even under variable lighting, orientation, and scale conditions. This approach seeks to offer a more reliable, efficient, and transferable solution to crack detection, ultimately contributing to more proactive and cost-effective structural health monitoring practices.

2. Methodology

The proposed methodology integrates a SOM with KAZE feature matching to enable robust crack detection and monitoring in concrete structures. As shown in Figure 1, the approach comprises four main stages. First, pixels in the initial crack image are classified using a SOM, which provides a representative weight vector that can be directly applied to subsequent images. This facilitates efficient crack growth detection over time. Second, a crack labeling process is employed to remove residual noise from the SOM-classified image. Third, a combination of dilation and superimposition techniques is introduced to restore thin or partially removed crack segments, ensuring a more complete and accurate crack representation. Finally, the KAZE feature matching algorithm is implemented to identify, localize, and track crack features, supported by a comparative evaluation of alternative feature detection algorithms available in MATLAB.

Figure 1: Method system architecture

2.1 SOM for Crack Detection

The proposed approach employs a SOM to classify pixels in an initial crack image, thereby generating a representative weight vector that can be directly applied to subsequent images for efficient crack growth detection. By performing pixel clustering on the initial image and storing the resulting SOM weight vector, future images can be classified without retraining, significantly reducing computational overhead and streamlining long-term SHM efforts. In this

manner, changes in the crack pattern can be rapidly identified and assessed, providing a cost-effective and time-efficient solution for ongoing infrastructure maintenance.

2.1.1 Introduction to SOM

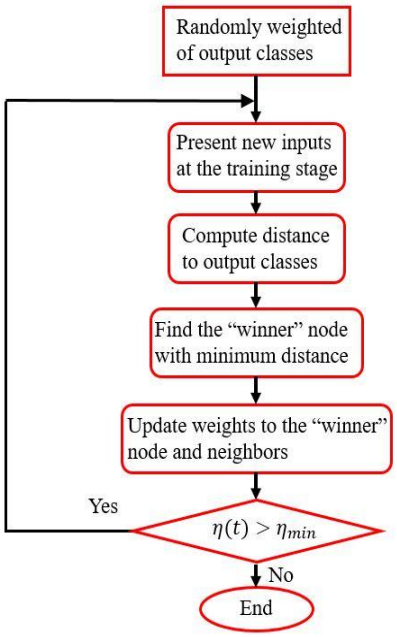


Figure 2: SOM algorithm

A SOM is an unsupervised learning method that maps high-dimensional input data onto a low-dimensional grid while preserving topological relationships [29]. The training process of a SOM involves two main stages: a competitive stage, where a "winner" node—or Best-Matching Unit (BMU)—is found based on minimal distance to the input vector, and a cooperative stage, where the BMU's weights and those of its neighbors are updated [30–32]. Figure 2 illustrates the SOM algorithm.

Let $p = \{p_1, p_2, \dots, p_n\}$ be an n-dimensional input vector, and $w_i = \{w_{i1}, w_{i2}, \dots, w_{in}\}$ represent the weight vector of node i. Upon receiving a new input p, the distance d_i between p and w_i is calculated as:

$$d_i = \sqrt{\sum_{j=1}^{n}(p_j(t) - w_{ij}(t))^2} \tag{1}$$

Where $p_j(t)$ is the input to property j at time t and $w_{ij}(t)$ is the weight of property j of output node i at time t [33]. The node c that minimizes d_i is identified as the BMU [34]. After identifying the BMU, weights are updated to increase their similarity with the input:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)h_{ij}^{c}(t)(p_j(t) - w_{ij}(t)) \tag{2}$$

Where $\eta(t)$ is the learning rate at time t and $h_{ij}^{c}(t)$ is the neighborhood kernel around the winner node at time t, both of which should be non-increasing functions, and t denotes the current time step. For pixel clustering tasks, a simple linear decay in the learning rate is sufficient [29]:

$$\eta(t) = \eta(0)\left(1 - \frac{t}{T}\right) \tag{3}$$

Where T is the training length. As t progresses, $\eta(t)$ diminishes until reaching the threshold η_{min}.

The neighborhood function influences how far the BMU's influence extends during training [35]:

$$h_{ij}(t) = \begin{cases} 1 & if\ i \in N_c(d) \\ 0 & if\ i \notin N_c(d) \end{cases} \tag{4}$$

Where $N_c(d)$ includes nodes lying within radius d of the BMU c [36].

$$N_c(d) = \{j \mid d_{cj} \leq d\} \tag{5}$$

A one-dimensional SOM map is employed, where each node's neighbors depend on its grid position. After this process, the input data are classified into several classes with similar characteristics. Crucially, the weight vector derived from the initial crack image remains a stable representation of the "crack" class. When subsequent images are processed, these same weight vectors can be used to rapidly identify crack pixels and highlight changes in crack morphology, thereby enabling efficient and reliable crack growth detection over time without the need for retraining.

2.1.2 Properties of pixels applied for SOM

To enhance the SOM's ability to classify pixels as crack or non-crack, multiple pixel-level features are incorporated:

Grayscale

Cracks generally appear as darker regions. Consequently, grayscale intensity is a key feature. Lower grayscale values indicate a higher likelihood of crack presence [37, 38].

Edge Detection

Identifying crack edges relies on the observation that pixels along a crack's boundary typically exhibit greater intensity differences compared to their surroundings [39] .

Pixel (i, j) and its eight neighbors forming a 3×3 matrix. To evaluate the edge property, multiple lines passing through (i, j) are examined, including four diagonal lines and eight obtuse folds. As an example, line A runs from (i, j) to $(i + 1, j + 1)$:

$$A_{ij}^n = |(p_{i,j-1} + p_{i+1,j-1} + p_{i+1,j}) - (p_{i,j+1} + p_{i-1,j} + p_{i-1,j+1})|$$

where $p_{i,j}$ denotes the grayscale value of the pixel at coordinates (i, j).

Similarly, consider line E, which passes through $(i - 1, j)$, (i, j), and $(i + 1, j)$:

$$E_{ij}^n = \left|2 \times \left(p_{i-1,j-1} + p_{i,j-1}\right) - (p_{i-1,j+1} + \mathrm{p}_{i,j+1} + p_{i+1,j+1} + p_{i+1,j})\right|.$$

By applying analogous computations to all four diagonal lines and eight obtuse folds, a total of 12 line-based intensity differences are obtained. The maximum absolute difference among these 12 values represents the normal edge detection property:

$$\epsilon_{ij}^n = \max\left\{A_{ij}^n, B_{ij}^n, C_{ij}^n, D_{ij}^n, E_{ij}^n, F_{ij}^n, G_{ij}^n, H_{ij}^n, I_{ij}^n, J_{ij}^n, K_{ij}^n, L_{ij}^n\right\} \tag{6}$$

Figure 3 illustrates the lines used for edge detection, with each red line denoting a specific direction for intensity evaluation.

(a) Red line A (b) Red line B (c) Red line C (d) Red line D

(e) Red line E (f) Red line F (g) Red line G (h) Red line H

(i) Red line I (j) Red line J (k) Red line K (l) Red line L

Figure 3: Edge detection algorithm

Contrast

Contrast measures the intensity difference distinguishing an object from its surroundings [40] . For a pixel (i, j), considers a 3×3 matrix centered at (i, j):

$$S_{ij} = p_{i-1,j-1} + p_{i-1,j} + p_{i-1,j+1} + p_{i,j-1} + p_{i,j} + p_{i,j+1} + p_{i+1,j-1} + p_{i+1,j} + p_{i+1,j+1}$$

Sixteen neighboring pixels around this 3×3 region are located at $(i-2, j-2)$, $(i-2, j-1)$, $(i-2, j)$, $(i-2, j+1)$, $(i-2, j+2)$, $(i-1, j-2)$, $(i-1, j+2)$, $(i, j-2)$, $(i, j+2)$, $(i+1, j-2)$, $(i+1, j+2)$, $(i+2, j-2)$, $(i+2, j-1)$, $(i+2, j)$, $(i+2, j+1)$, and $(i+2,$

j+2). Let $A_1, A_2, \ldots, A_{16}$ represent the grayscale values of these pixels. The contrast *C* is defined as the maximum absolute difference between S_{ij} and each A_k:

$$\mathrm{C} = \max(|S_{ij} - A_1|, |S_{ij} - A_2|, \ldots, |S_{ij} - A_{16}|) \quad (7)$$

This measure helps highlight subtle differences in intensity, aiding in the differentiation of cracks from their immediate surroundings.

Thin Crack Detection

Thin cracks, particularly those that are only one pixel wide, require specialized processing to distinguish them from noise. To achieve this, ratios of average grayscale values are computed along specific lines passing through the target pixel. For example, consider line A again:

$$A_{ij}^t = \left(\frac{(p_{i-1,j-1} + p_{i,j} + p_{i+1,j+1})/3}{\frac{p_{i,j-1} + p_{i+1,j-1} + p_{i+1,j} + p_{i-1,j} + p_{i-1,j+1} + p_{i.j+1}}{6}} \right) \times 100$$

Performing a similar calculation for all 12 lines and taking the maximum value identifies pixels likely to belong to thin cracks:

$$\epsilon_{ij}^t = \max\left\{A_{ij}^t, B_{ij}^t, C_{ij}^t, D_{ij}^t, E_{ij}^t, F_{ij}^t, G_{ij}^t, H_{ij}^t, I_{ij}^t, J_{ij}^t, K_{ij}^t, L_{ij}^t\right\} \quad (8)$$

This property ensures that even very narrow cracks are detected and that small, noise-induced features are more effectively filtered out.

Average grayscale (8 and 24 Neighbors)

Both the average grayscale of a pixel plus its 8 neighbors and the average grayscale with 24 neighbors are considered. These measures help differentiate dark, crack-related pixels from the background and improve noise resilience.

Relative Grayscale Values

Cracks form pronounced "valleys" in grayscale profiles along rows and columns. By comparing a pixel's grayscale to the median row and column values [41–43], pixels that deviate significantly (i.e., are much darker) can be identified as crack pixels.

$$\rho_{ij} = \frac{p_{ij}}{\mu(p_i)} \times 100$$

$$\gamma_{ij} = \frac{p_{ij}}{\mu(p_j)} \times 100 \quad (9)$$

Where p_{ij} is the pixel's grayscale value, $\mu(p_i)$ is the median grayscale of row i, and $\mu(p_j)$ is the median grayscale of column j.

Hue

Not all dark regions represent cracks—some may be stains or grout joints [44]. Hue, defined as the dominant wavelength of color, can help distinguish actual cracks from other dark features [27, 45, 46]:

$$\text{Hue} = \begin{cases} \frac{60(G-B)}{\max(R,G,B)-\min(R,G,B)} \; if\ R\ is\ max \\ \left[120 + \frac{60(B-R)}{\max(R,G,B)-\min(R,G,B)}\right] \; if\ G\ is\ max \\ \left[240 + \frac{60(R-G)}{\max(R,G,B)-\min(R,G,B)}\right] if\ B\ is\ max \end{cases} \tag{10}$$

By integrating these diverse properties into the SOM classification process and subsequently applying the initially derived weight vector to future images, the proposed method effectively identifies cracks and monitors their progression. This robust approach enables efficient long-term crack growth detection and offers a valuable tool for proactive SHM strategies.

2.2 Crack labeling

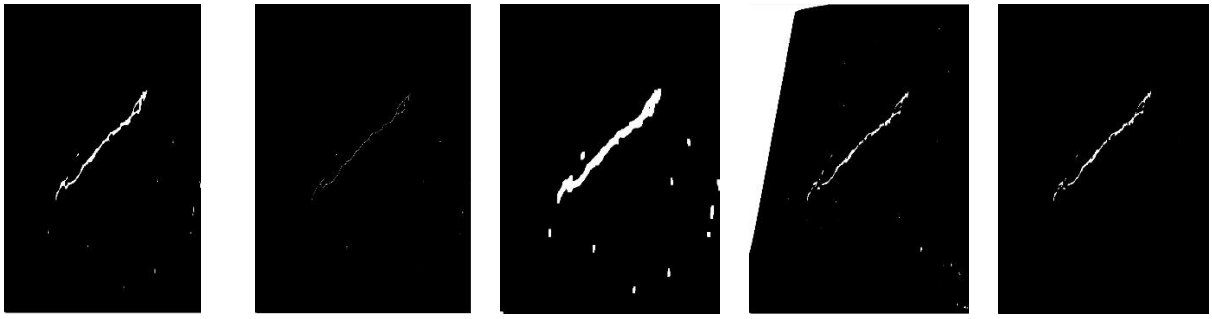
Following SOM classification, residual noise may still remain in the detected crack image. To address this, the MATLAB Image Processing Toolbox is employed to streamline the noise removal process. A key step involves using the built-in "bwlabel" function, which assigns labels to 8-connected objects in a binary image. Prior to labeling, the SOM-classified image is converted to a binary format, with the background set to black (0) and the crack pixels to white (1). The "bwlabel" function operates by run-length encoding the input image, assigning preliminary labels, recording label equivalences, and ultimately resolving these equivalences before relabeling the runs accordingly [47]. This procedure effectively separates distinct crack regions from extraneous noise, yielding a more coherent and interpretable representation of the detected cracks.

To further refine this representation, the concept of area moment of inertia is incorporated. In image analysis, the moment of inertia can be envisioned as the "rotational mass" of pixel intensities. By computing the area moment of inertia for each labeled object, it becomes possible to differentiate genuine crack elements from spurious noise. Setting an appropriate threshold on this measure ensures that only meaningful crack structures are retained, while irrelevant fragments are discarded. The outcome is a reconstructed crack image in which significant structural features are preserved, thereby improving the clarity and reliability of subsequent analyses.

2.3 Dilation and Superimposition

Although labeling and thresholding the moment of inertia substantially reduce noise and isolate crack features, small artifacts may still persist, and certain crack segments may have been inadvertently removed. To restore these missing segments and eliminate lingering noise, morphological operations are applied. The combined crack image obtained after labeling is first skeletonized to represent the cracks in their minimal form. Dilation is then performed to enhance the visibility and continuity of the crack structures.

Finally, the binary image of the original crack is superimposed onto the dilated skeletonized image using the logical "and" operation in MATLAB. This step reintroduces previously removed crack elements and eliminates any residual noise, resulting in a more accurate and continuous crack representation. Figure 4 provides an illustrative example of this process, demonstrating how dilation and superimposition refine the final crack image to ensure reliable, high-fidelity inputs for subsequent analysis or monitoring tasks.



(a) Reconstructed crack image after labeling | (b) Skeletonized crack | (c) Dilated crack | (d) Binary of original crack image | (e) Superimposition of dilated crack and binary crack

Figure 4: Example Process of Dilation and Superimposition

2.4 Feature Match

Identifying and extracting reliable feature points are key steps in enhancing image-based crack detection. Feature detection typically involves locating interest points—such as blobs, corners, or edges—that offer distinctive and repeatable patterns for subsequent processing tasks [48–51]. Ideal feature detectors should be scale-invariant, distinctive, resilient to noise and distortions, highly repeatable, well-localized, straightforward to implement, and computationally efficient. Achieving this balance is critical in demanding applications like crack detection, where variations in scale, rotation, illumination, and environmental conditions are common.

MATLAB's Image Processing and Computer Vision Toolboxes provide numerous built-in feature detection algorithms widely employed in object recognition, image matching, and structure-from-motion analysis. Among the well-known methods are Histogram of Oriented Gradients (HOG), Harris-Stephens (Harris), Minimum Eigenvalue (MinEigen or KLT), Scale-Invariant Feature Transform (SIFT), Maximally Stable Extremal Regions (MSER), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Oriented FAST and Rotated BRIEF (ORB), Binary Robust Invariant Scalable Keypoints (BRISK), and KAZE. Each of these methods possesses distinct advantages and limitations, influencing their suitability for crack detection.

Histogram of Oriented Gradients (HOG)

HOG [48, 49, 52, 53] leverages local intensity gradients or edge directions to characterize object shapes. While effective in object recognition, HOG's sensitivity to image rotation [54] and relatively slow computation for large-scale images reduces its utility in time-sensitive crack detection applications.

Harris-Stephens (Harris)

The Harris corner detector [50, 55–57] identifies points where intensity changes occur in multiple directions. Despite robustness to translation, rotation, and illumination changes, its corner-based nature and limited scale invariance undermine its efficacy for detecting non-corner-like crack patterns [58, 59].

Minimum Eigenvalue (MinEigen)

The KLT detector [58, 60, 61] focuses on selecting points with sufficient intensity variation. Although KLT effectively detects corners and is insensitive to certain image changes, it may fail when tracking features under noisy or dynamically changing conditions, making it less suitable for robust crack detection.

Scale-Invariant Feature Transform (SIFT)

SIFT [51, 62–64] is widely used and recognized for its rotational, scale, and affine invariance, enabling it to detect stable keypoints across a range of transformations. However, its high computational cost can be prohibitive for real-time or large-scale crack detection scenarios.

Maximally Stable Extremal Regions (MSER)

MSER [65–69] identifies stable regions across varying intensity thresholds. While offering multi-scale detection and affine invariance, MSER's high sensitivity to lighting changes limits its applicability in crack detection, where subtle variations in illumination can occur.

Speeded Up Robust Feature (SURF)

SURF [70–72] provides a faster, Hessian-based alternative to SIFT and is well-suited for real-time applications. Although often effective, SURF may still struggle with the nuanced patterns and conditions encountered in crack detection.

Features from Accelerated Segment Test (FAST)

FAST [73, 74] excels in speed, making it ideal for real-time applications, but it may perform poorly in noisy environments. For crack detection, where fine features and texture variations are prevalent, FAST's minimal pixel analysis can limit its effectiveness.

Oriented FAST and Rotated BRIEF (ORB)

ORB [75–79] combines the FAST detector with the BRIEF descriptor, optimizing speed and efficiency. ORB offers improved scale invariance and good performance in the presence of noise and blurring. This makes it a promising candidate for crack detection, though certain scenarios may still challenge its consistency.

Binary Robust Invariant Scalable Keypoints (BRISK)

BRISK [63, 64, 80] is scale- and rotation-invariant, emphasizing computational efficiency and real-time applications. While BRISK may be less robust under severe image degradation or large in-plane rotations, it can still be a viable choice depending on the specific constraints of the crack detection task.

KAZE

KAZE [63, 81, 82] employs a nonlinear diffusion-based scale space to preserve edges and details. By adapting image blurring locally, KAZE achieves reduced noise while maintaining sharp boundaries, providing rotation, scale, and restricted affine invariance. Its capacity to handle camera distortion and lack of explicit scale or rotation information makes KAZE particularly well-suited for crack detection.

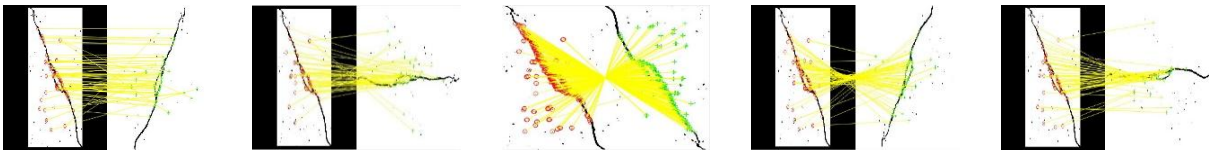
Comparison and Selection of KAZE for Crack Detection

In the context of concrete crack detection, algorithms like ORB, SIFT, SURF, BRISK, and KAZE demonstrate enhanced scale and rotation invariance compared to HOG, Harris, MinEigen (KLT), FAST, and MSER. Among these top candidates, KAZE consistently outperforms the others, especially under challenging conditions such as varying scales, rotations, and distortions (Figures 5–7). While ORB, SIFT, SURF, and BRISK show promise, KAZE's superior

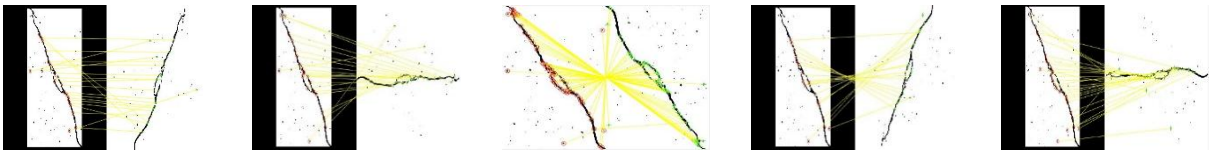
performance in accurately detecting and matching crack features, even under severe distortions, makes it the most suitable algorithm for this application

60° rotation 120° rotation 180° rotation 240° rotation 300° rotation

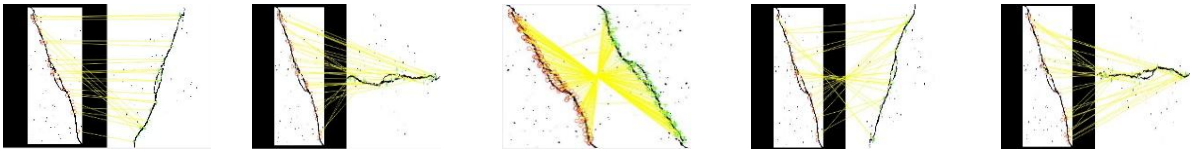
Concrete crack images with different rotation angles



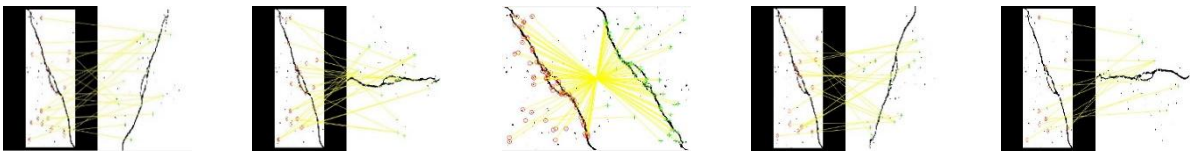
Feature matching with different rotation angles using ORB algorithm



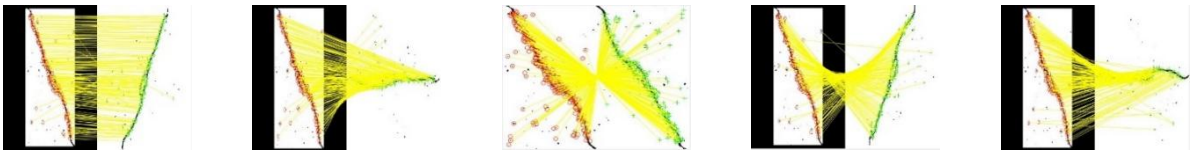
Feature matching with different rotation angles using SIFT algorithm



Feature matching with different rotation angles using SURF algorithm



Feature matching with different rotation angles using BRISK algorithm

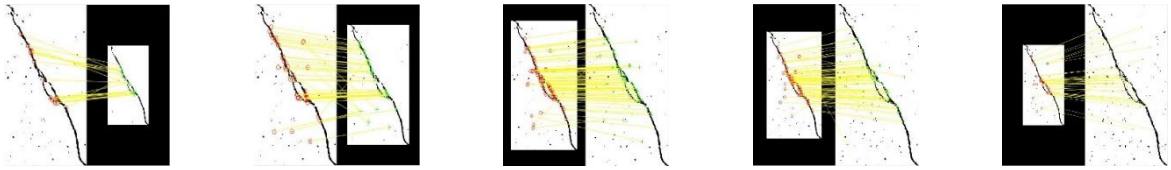


Feature matching with different rotation angles using KAZE algorithm

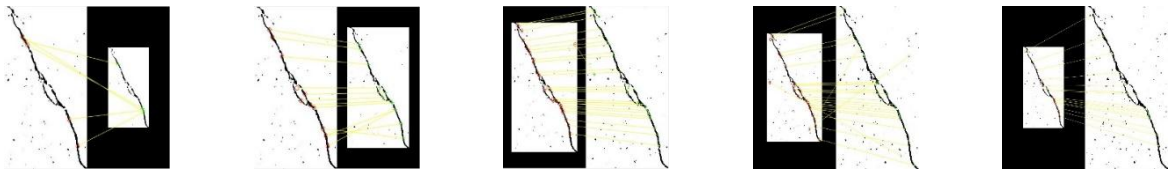
Figure 5: Crack feature matching with varied rotation using ORB, SIFT, SURF, BRISK, and KAZE

Scale of 0.5 Scale of 0.75 Scale of 1.25 Scale of 1.5 Scale of 2

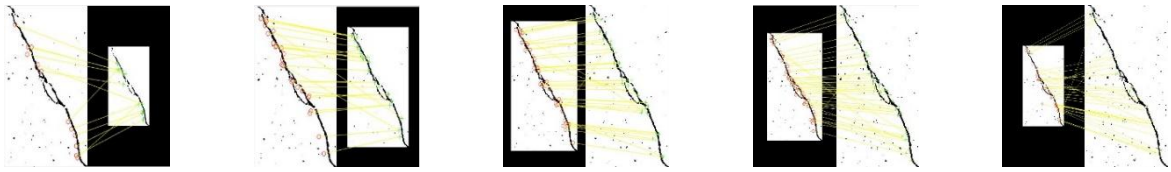
Concrete crack images with different scales



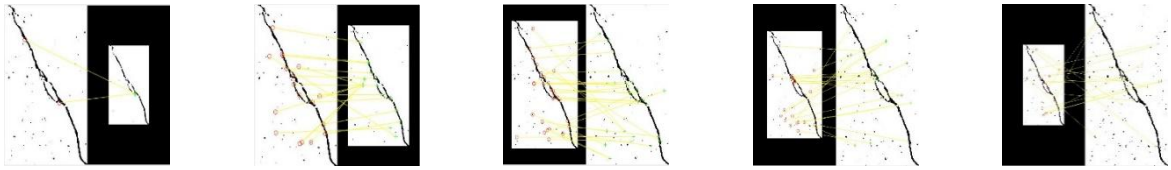
Feature matching with different scales using ORB algorithm



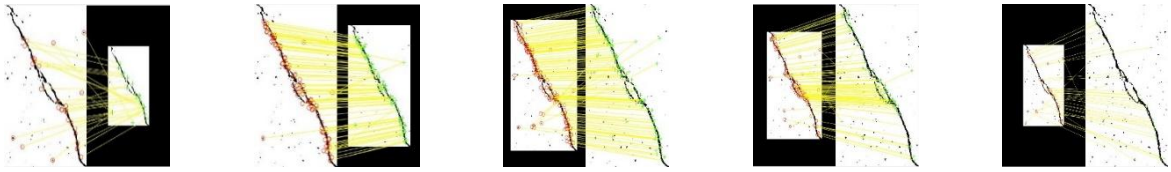
Feature matching with different scales using SIFT algorithm



Feature matching with different scales using SURF algorithm

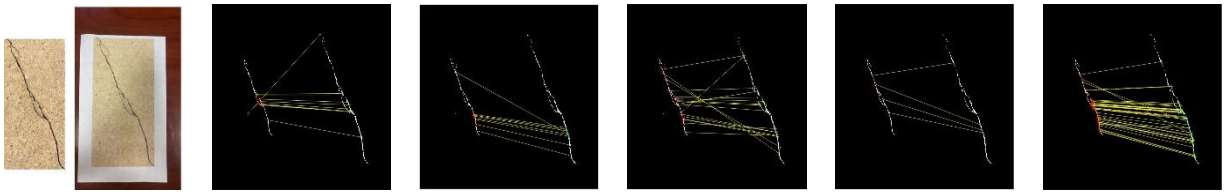


Feature matching with different scales using BRISK algorithm



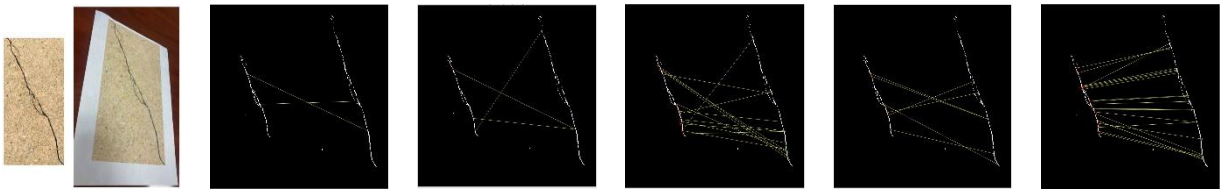
Feature matching with different scales using KAZE algorithm

Figure 6: Crack feature matching with varied scale using ORB, SIFT, SURF, BRISK, and KAZE



(a) Images (b) ORB (c) SIFT (d) SURF (e) BRISK (f) KAZE

Feature matching for crack images with mild distortion by different algorithms



(a) Images (b) ORB (c) SIFT (d) SURF (e) BRISK (f) KAZE

Feature matching for crack images with severe distortion by different algorithms

Figure 7: Crack feature matching with distortion using ORB, SIFT, SURF, BRISK, and KAZE

Given that real-world scenarios often lack reliable scale or rotation references, KAZE's adaptability and robustness provide a distinct advantage. Although this study focuses on rotation- and scale-related disturbances, future work will consider other factors, such as varying perspectives and lighting conditions, to further refine the crack detection methodology. Based on the current analysis, KAZE is selected as the feature matching algorithm for the proposed crack detection framework.

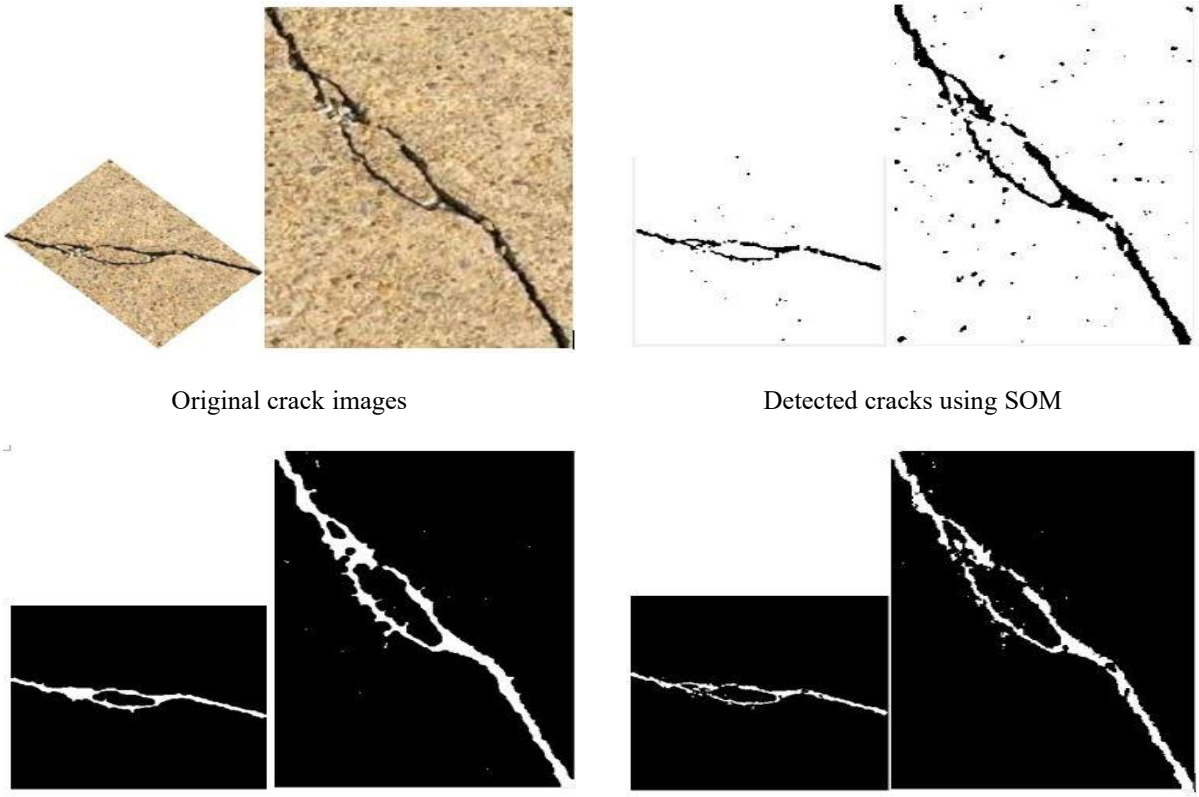
3. Experiments and Results

This section presents a series of experiments designed to evaluate the proposed methodology under varying conditions. While a concrete crack is employed as a representative example, it is important to note that the proposed algorithm is not limited to this material and can also be applied to cracks in brick structures or even shadow-induced cracks. The experiments are organized into three key scenarios: (1) detection of identical cracks, (2) detection of crack growth, and (3) identification of an unknown crack segment. In each case, a distinct experimental strategy is employed, and the results are analyzed to provide a comprehensive understanding of the method's performance.

3.1 Identical crack detection

In the first set of experiments, two images of the same crack are considered, each exhibiting differences in rotation and scale. By processing these images through the successive stages of SOM classification, labeling, dilation,

superimposition, and KAZE-based feature matching, the effectiveness of the proposed approach is demonstrated. Figure 8 illustrates the progression of this process. The method successfully identifies the same crack despite alterations in scale and rotation, confirming that the integrated SOM–KAZE framework reliably detects identical cracks under varying conditions.



Original crack images | Detected cracks using SOM

After Labelling | After dilation and superposition

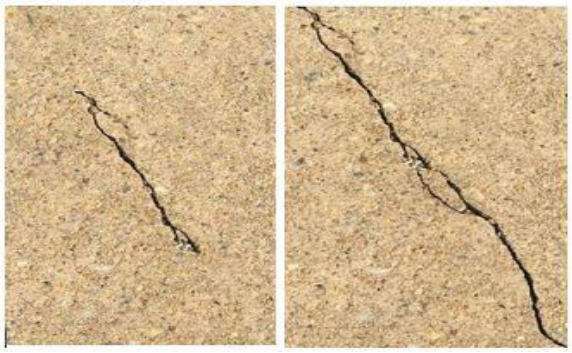
Feature match by KAZE | Detected crack

Figure 8: Identical crack detection

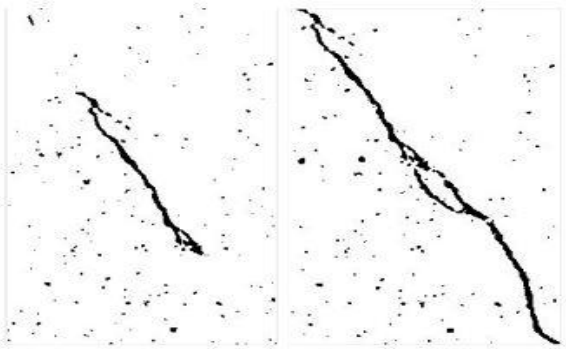
3.2 Crack growth detection

3.2.1 Crack growth with the same scale and rotation

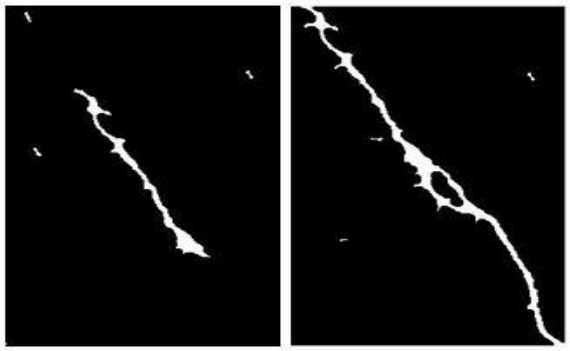
Next, the methodology is evaluated for its ability to detect crack growth while maintaining the same scale and rotation parameters. Two images are analyzed: one capturing the initial crack and another depicting its growth over time. The step-by-step outcomes, illustrated in Figure 9, confirm that the proposed method accurately identifies and quantifies crack growth when geometric parameters remain constant. The results highlight the potential of the approach to monitor structural integrity efficiently, enabling timely maintenance interventions.



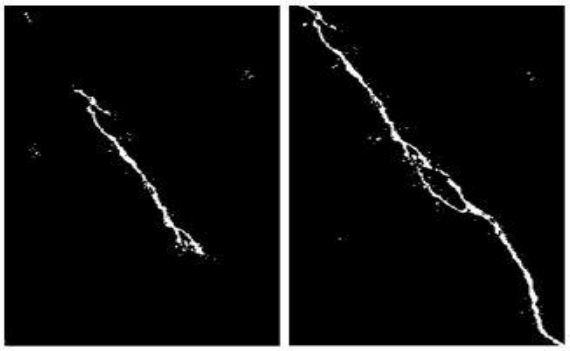
Original crack images



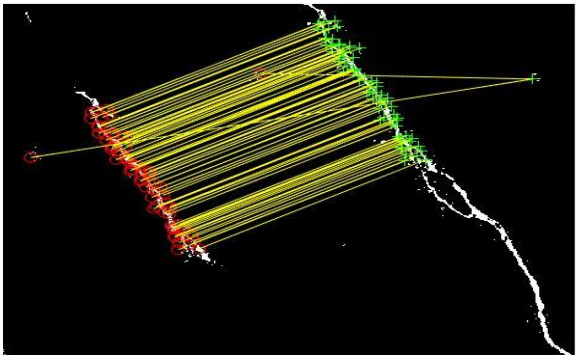
Detected cracks using SOM

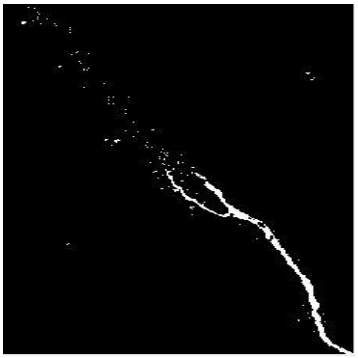


After Labelling



After dilation and superposition



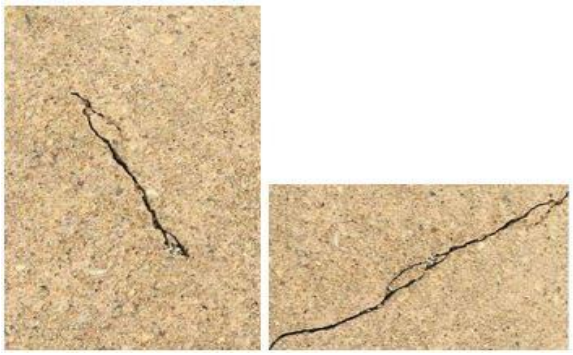


Feature match by KAZE Detected crack growth

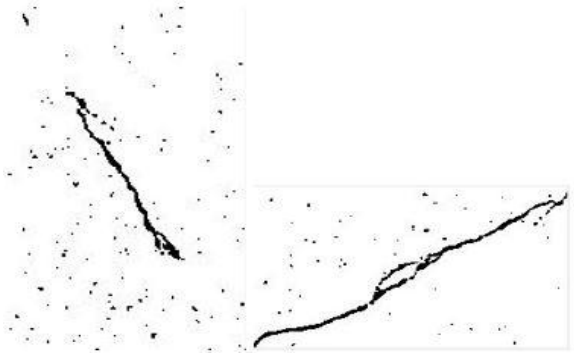
Figure 9: Crack growth detection without changes in scales and rotations

3.2.2 Crack growth with different scales and rotations

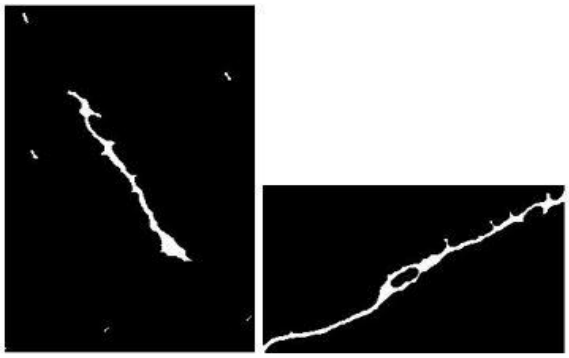
In a more challenging scenario, two images are analyzed where the crack has not only grown but also been captured under different rotations and scales. As shown in Figure 10, the methodology successfully adapts to these changes, detecting the growing crack segment even when subjected to 90-degree rotation and a 0.75 scale factor. This demonstrates the robustness and versatility of the algorithm in handling realistic field conditions, where cracks may be observed from varying perspectives and magnifications.



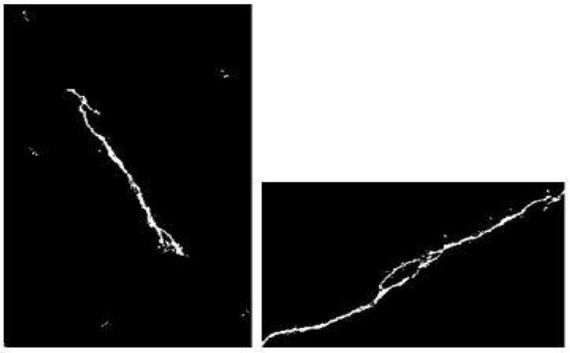
Original crack images



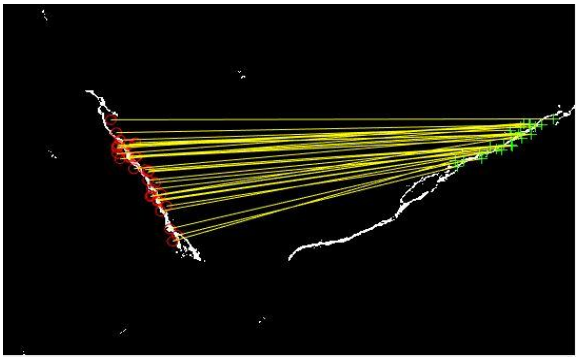
Detected cracks using SOM



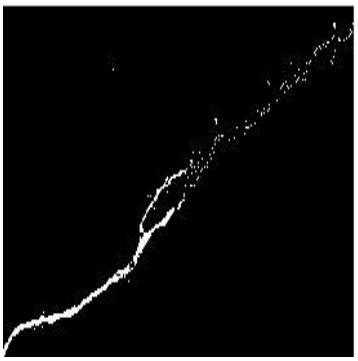
After Labelling



After dilation and superposition



Feature match by KAZE

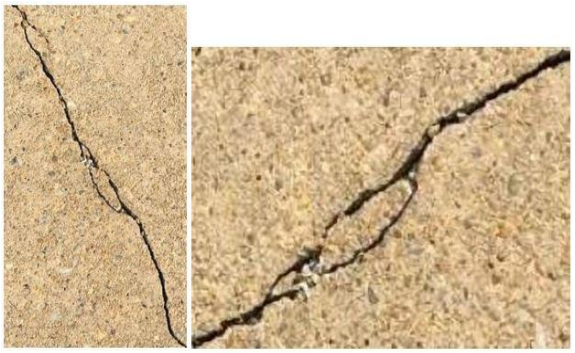


Detected crack growth

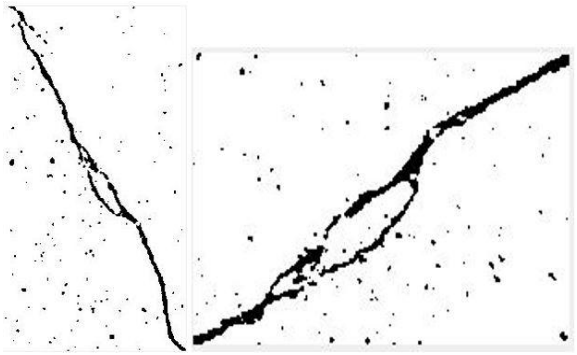
Figure 10: Crack growth detection with 90° rotation and scale of 0.75

3.3 Crack part positioning

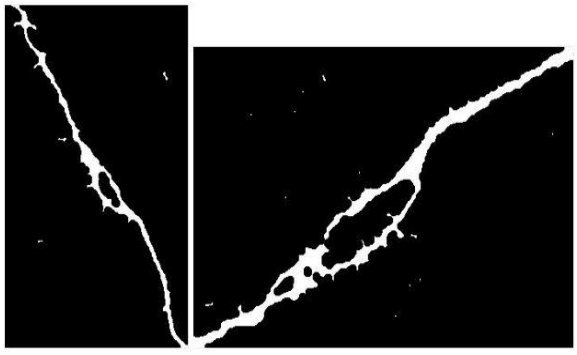
Finally, an experiment is conducted to identify a previously unknown crack segment. Here, the method evaluates a partial crack image characterized by indeterminate rotation and scale changes. Figure 11 presents the results, with a detected rotation of 90 degrees and a scale factor of 2.5. This confirms the algorithm's capacity to locate and characterize crack segments with limited prior information, further underscoring its applicability in complex inspection tasks.

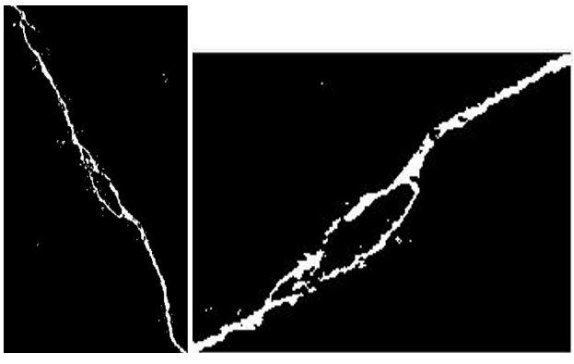


Original crack images

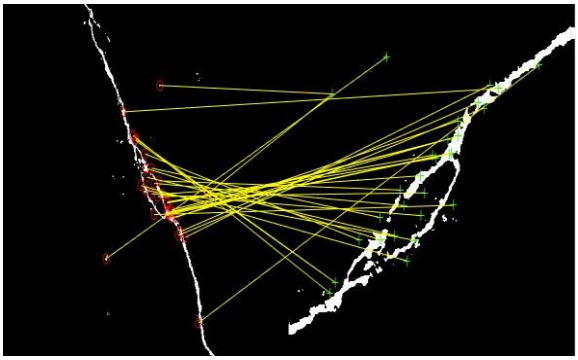


Detected cracks using SOM



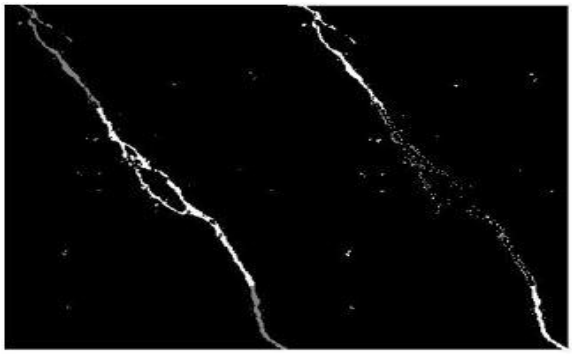


After Labelling



Feature match by KAZE

After dilation and superposition



Detected crack position

Figure 10: Detection of an unidentified crack segment

4. Conclusion

This study introduces a novel crack detection approach that integrates a SOM with the KAZE feature descriptor. By clustering the pixels of an initial crack image and storing the resulting weight vectors, subsequent images can be analyzed efficiently without retraining. Through the application of labeling, dilation, and superimposition, noise is reduced while preserving thin crack structures. KAZE-based feature matching further enhances the method's adaptability, allowing for reliable detection and analysis of cracks under varying scales and rotations.

The experimental evaluation, conducted using concrete crack images as a primary example, demonstrates the method's effectiveness in detecting identical cracks, monitoring crack growth, and identifying unknown crack segments. Notably, the approach is not limited to concrete cracks and can be extended to brick cracks or shadow-induced cracks, thereby offering broad applicability in structural health monitoring scenarios. Compared to conventional image processing techniques, the proposed method delivers improved accuracy, automation, and robustness against variable imaging conditions.

Overall, the integration of SOM classification with KAZE-based feature detection provides a powerful framework for crack analysis. Future research can expand upon these findings by considering different perspectives, lighting conditions, and environmental factors. The presented methodology represents a significant step toward more efficient and comprehensive image-based crack assessment systems, ultimately supporting proactive maintenance and enhanced infrastructure safety.

Reference

1. Gharehbaghi VR, Noroozinejad Farsangi E, Noori M, et al (2022) A Critical Review on Structural Health Monitoring: Definitions, Methods, and Perspectives. Arch Computat Methods Eng 29:2209–2235. https://doi.org/10.1007/s11831-021-09665-9

2. Yao Y, Tung S-TE, Glisic B (2014) Crack detection and characterization techniques—An overview. Structural Control and Health Monitoring 21:1387–1413. https://doi.org/10.1002/stc.1655

3. Farrar CR, Worden K (2006) An introduction to structural health monitoring. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 365:303–315. https://doi.org/10.1098/rsta.2006.1928

4. Hoang N-D (2018) Detection of Surface Crack in Building Structures Using Image Processing Technique with an Improved Otsu Method for Image Thresholding. Advances in Civil Engineering 2018:3924120. https://doi.org/10.1155/2018/3924120

5. Inman DJ, Farrar CR, Junior VL, Junior VS (2005) Damage prognosis: for aerospace, civil and mechanical systems. John Wiley & Sons

6. Munawar HS, Hammad AWA, Haddad A, et al (2021) Image-Based Crack Detection Methods: A Review. Infrastructures 6:115. https://doi.org/10.3390/infrastructures6080115

7. Rimkus A, Podviezko A, Gribniak V (2015) Processing digital images for crack localization in reinforced concrete members. Procedia Engineering 122:239–243

8. Dias-da-Costa D, Valença J, Júlio E, Araújo H (2017) Crack propagation monitoring using an image deformation approach. Structural Control and Health Monitoring 24:e1973

9. Fujita Y, Mitani Y, Hamamoto Y (2006) A method for crack detection on a concrete structure. IEEE, pp 901–904

10. Jahangir H, Esfahani MR (2012) Structural damage identification based on modal data and wavelet analysis

11. Zawad MRS, Zawad MFS, Rahman MA, Priyom SN (2021) A comparative review of image processing based crack detection techniques on civil engineering structures. Journal of Soft Computing in Civil Engineering 5:58–74

12. Gupta P, Dixit M (2022) Image-based crack detection approaches: a comprehensive survey. Multimed Tools Appl 81:40181–40229. https://doi.org/10.1007/s11042-022-13152-z

13. Khan MA-M, Kee S-H, Pathan A-SK, Nahid A-A (2023) Image Processing Techniques for Concrete Crack Detection: A Scientometrics Literature Review. Remote Sensing 15:2400. https://doi.org/10.3390/rs15092400

14. Mohan A, Poobal S (2018) Crack detection using image processing: A critical review and analysis. Alexandria Engineering Journal 57:787–798. https://doi.org/10.1016/j.aej.2017.01.020

15. Canny J (1986) A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence 679–698

16. Sobel I, Feldman G (1968) A 3x3 isotropic gradient operator for image processing. a talk at the Stanford Artificial Project in 1968:271–272

17. Xiao Y, Li J (2018) Crack Detection Algorithm based on the Fusion of Percolation Theory and Adaptive Canny Operator. In: 2018 37th Chinese Control Conference (CCC). pp 4295–4299

18. Chen J-H, Su M-C, Cao R, et al (2017) A self organizing map optimization based image recognition and processing model for bridge crack inspection. Automation in Construction 73:58–66

19. Dung CV, Anh LD (2019) Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction 99:52–58. https://doi.org/10.1016/j.autcon.2018.11.028

20. Nguyen SD, Tran TS, Tran VP, et al (2023) Deep learning-based crack detection: A survey. International Journal of Pavement Research and Technology 16:943–967

21. Prasanna P, Dana KJ, Gucunski N, et al (2016) Automated Crack Detection on Concrete Bridges. IEEE Transactions on Automation Science and Engineering 13:591–599. https://doi.org/10.1109/TASE.2014.2354314

22. Hamishebahar Y, Guan H, So S, Jo J (2022) A Comprehensive Review of Deep Learning-Based Crack Detection Approaches. Applied Sciences 12:. https://doi.org/10.3390/app12031374

23. Li J, Li X, Liu K, Yao Z (2022) Crack Identification for Bridge Structures Using an Unmanned Aerial Vehicle (UAV) Incorporating Image Geometric Correction. Buildings 12:1869. https://doi.org/10.3390/buildings12111869

24. Ai D, Jiang G, Lam S-K, et al (2023) Computer vision framework for crack detection of civil infrastructure—A review. Engineering Applications of Artificial Intelligence 117:105478. https://doi.org/10.1016/j.engappai.2022.105478

25. Kavakiotis I, Tsave O, Salifoglou A, et al (2017) Machine learning and data mining methods in diabetes research. Computational and structural biotechnology journal 15:104–116

26. Kohonen T (2013) Essentials of the self-organizing map. Neural networks 37:52–65

27. Mathavan S, Rahman M, Kamal K (2015) Use of a self-organizing map for crack detection in highly textured pavement images. Journal of Infrastructure Systems 21:04014052

28. Shifani SA, Thulasiram P, Narendran K, Sanjay D (2020) A study of methods using image processing technique in crack detection. IEEE, pp 578–582

29. Vesanto J, Alhoniemi E (2000) Clustering of the self-organizing map. IEEE Transactions on neural networks 11:586–600

30. Ghaseminezhad M, Karami A (2011) A novel self-organizing map (SOM) neural network for discrete groups of data clustering. Applied soft computing 11:3771–3778

31. Rozenberg G, Bck T, Kok JN (2011) Handbook of Natural Computing. In: Springer Berlin Heidelberg

32. Sathya R, Abraham A (2013) Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence 2:34–38

33. Rossi F, Conan-Guez B, El Golli A (2004) Clustering functional data with the SOM algorithm. Citeseer, pp 305–312

34. Furukawa T (2009) SOM of SOMs. Neural Networks 22:463–478

35. Berglund E, Sitte J (2006) The parameterless self-organizing map algorithm. IEEE Transactions on neural networks 17:305–316

36. Vesanto J, Himberg J, Alhoniemi E, Parhankangas J (1999) Self-organizing map in Matlab: the SOM Toolbox. Finland, pp 16–17

37. Van Kessel PF, Hornbeck LJ, Meier RE, Douglass MR (1998) A MEMS-based projection display. Proceedings of the IEEE 86:1687–1704

38. Vincent L (1993) Grayscale area openings and closings, their efficient implementation and applications. Citeseer, pp 22–27

39. Sinha SK, Fieguth PW (2006) Automated detection of cracks in buried concrete pipe images. Automation in construction 15:58–72

40. Gu K, Tao D, Qiao J-F, Lin W (2017) Learning a no-reference quality assessment model of enhanced images with big data. IEEE transactions on neural networks and learning systems 29:1301–1313

41. Kaseko MS, Ritchie SG (1993) A neural network-based methodology for pavement crack detection and classification. Transportation Research Part C: Emerging Technologies 1:275–291

42. Nagabhushana S (2005) Computer vision and image processing. New Age International

43. Zhang W, Zhang Z, Qi D, Liu Y (2014) Automatic crack detection and classification method for subway tunnel safety monitoring. Sensors 14:19307–19328

44. Phillips D (1994) Image processing in C: analyzing and enhancing digital images. R & D Publications. Inc, Lawrence

45. Gevers T, Smeulders AW (1999) Color-based object recognition. Pattern recognition 32:453–464

46. Zou J, Kim H (2007) Using hue, saturation, and value color space for hydraulic excavator idle time analysis. Journal of computing in civil engineering 21:238–246

47. Haralick RM, Shapiro LG (1993) Computer and Robot Vision. Addison-Wesley Publishing Company

48. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. Ieee, pp 886–893

49. Freeman WT, Roth M (1995) Orientation histograms for hand gesture recognition. Citeseer, pp 296–301

50. Harris C, Stephens M (1988) A combined corner and edge detector. Citeseer, pp 10–5244

51. Lowe DG (1999) Object recognition from local scale-invariant features. Ieee, pp 1150–1157

52. Shu C, Ding X, Fang C (2011) Histogram of the oriented gradient for face recognition. Tsinghua Science and Technology 16:216–224

53. Tomasi C (2012) Histograms of oriented gradients. Computer Vision Sampler 1–6

54. Cheon M-K, Lee W-J, Hyun C-H, Park M (2011) Rotation invariant histogram of oriented gradients. International Journal of Fuzzy Logic and Intelligent Systems 11:293–298

55. Derpanis KG (2004) The harris corner detector. York University 2:2

56. Khan RA, Islam S, Biswas R (2014) Automatic detection of defective rail anchors. IEEE, pp 1583–1588

57. Moravec HP (1980) Obstacle avoidance and navigation in the real world by a seeing robot rover. Stanford University

58. Shi J, Tomasi (1994) Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp 593–600

59. Shi Y, Cui L, Qi Z, et al (2016) Automatic road crack detection using random structured forests. IEEE Transactions on Intelligent Transportation Systems 17:3434–3445

60. Mainali P, Yang Q, Lafruit G, et al (2011) Robust low complexity corner detector. IEEE Transactions on Circuits and Systems for Video Technology 21:435–445

61. Tomasi C, Kanade T (1991) Detection and tracking of point. Int J Comput Vis 9:3

62. Mistry D, Banerjee A (2017) Comparison of feature detection and matching approaches: SIFT and SURF. GRD Journals-Global Research and Development Journal for Engineering 2:7–13

63. Tareen SAK, Saleem Z (2018) A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). pp 1–10

64. Truong MTN, Kim S (2016) A Review on Image Feature Detection and Description

65. Aanæs H, Dahl AL, Pedersen KS (2012) Interesting Interest Points: A Comparative Study of Interest Point Performance on a Unique Data Set. International Journal of Computer Vision 97:18–35. https://doi.org/10.1007/s11263-011-0473-8

66. Donoser M, Bischof H (2006) Efficient maximally stable extremal region (MSER) tracking. Ieee, pp 553–560

67. Matas J, Chum O, Urban M, Pajdla T (2004) Robust wide-baseline stereo from maximally stable extremal regions. Image and vision computing 22:761–767

68. Thosare H, Pandey M, Godse O, et al (2017) SURVEY PAPER ON TEXT RETRIEVAL AND TRANSLATION FROM SIGN BOARD IMAGES. International Journal of Approximate Reasoning 5:176–181

69. Turki H, Halima MB, Alimi AM (2017) Text detection based on MSER and CNN features. IEEE, pp 949–954

70. Bay H, Tuytelaars T, Van Gool L (2006) Surf: Speeded up robust features. Springer, pp 404–417

71. Choi D, Bell W, Kim D, Kim J (2021) UAV-Driven Structural Crack Detection and Location Determination Using Convolutional Neural Networks. Sensors 21:2650. https://doi.org/10.3390/s21082650

72. Oyallon E, Rabin J (2015) An Analysis of the SURF Method. Image Processing On Line 5:176–218. https://doi.org/10.5201/ipol.2015.69

73. Rosten E, Drummond T (2006) Machine Learning for High-Speed Corner Detection. In: Leonardis A, Bischof H, Pinz A (eds) Computer Vision – ECCV 2006. Springer, Berlin, Heidelberg, pp 430–443

74. Viswanathan D (2011) Features from Accelerated Segment Test (FAST)

75. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: Binary Robust Independent Elementary Features. In: Daniilidis K, Maragos P, Paragios N (eds) Computer Vision – ECCV 2010. Springer, Berlin, Heidelberg, pp 778–792

76. Calonder M, Lepetit V, Ozuysal M, et al (2012) BRIEF: Computing a Local Binary Descriptor Very Fast. IEEE Transactions on Pattern Analysis and Machine Intelligence 34:1281–1298. https://doi.org/10.1109/TPAMI.2011.222

77. Luo C, Yang W, Huang P, Zhou J (2019) Overview of Image Matching Based on ORB Algorithm. Journal of Physics: Conference Series 1237:032020. https://doi.org/10.1088/1742-6596/1237/3/032020

78. Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. Ieee, pp 2564–2571

79. Vinay A, Cholin AS, Bhat AD, et al (2018) An efficient ORB based face recognition framework for human-robot interaction. Procedia computer science 133:913–923

80. Leutenegger S, Chli M, Siegwart RY (2011) BRISK: Binary Robust invariant scalable keypoints. In: 2011 International Conference on Computer Vision. pp 2548–2555

81. Alcantarilla PF, Bartoli A, Davison AJ (2012) KAZE Features. In: Fitzgibbon A, Lazebnik S, Perona P, et al (eds) Computer Vision – ECCV 2012. Springer, Berlin, Heidelberg, pp 214–227

82. Liu C, Xu J, Wang F (2021) A Review of Keypoints' Detection and Feature Description in Image Registration. Sci Program 2021:8509164:1-8509164:25