International Journal of Scientific Advances
ISSN: 2708-7972
Volume: 6 | Issue: 3 | May – Jun 2025 Available Online: www.ijscia.com
DOI: 10.51542/ijscia.v6i3.11

Color-based Crack Detection using Image Analysis

Leul Deribe Abera*, Yancheng Li, Shaoqi Li

College of Civil Engineering, Nanjing Tech University, Nanjing, 211816, China

*Corresponding author details: Leul Deribe Abera; leul.deribe45@gmail.com

ABSTRACT

This paper explores the application of machine learning algorithms combined with image processing techniques for the detection of cracks in concrete structures. Specifically, it focuses on a color-based crack detection approach using image analysis to enhance the accuracy and efficiency of identifying structural defects. By leveraging advanced image processing methods, the system can extract visual features that differentiate between cracked and uncracked surfaces. The study also aims to develop an automated classification model capable of distinguishing between images of cracked and non-cracked concrete, thereby minimizing the need for manual inspection and improving the reliability of structural health monitoring.

Keywords: deep learning; machine learning; neural networks; crack detection.

INTRODUCTION

Infrastructure, including buildings and dams, deteriorates with time, and catastrophic events including quakes and landslides can accelerate this process; thus, structural monitoring is essential for predicting the lifespan of these edifices. Cracks not only diminish the aesthetic appeal of structures but also compromise their load-bearing capabilities.

Crack analysis can be conducted manually by specialists to create a drawing of the crack and assess its breadth and depth; however, this manual method is time-consuming and necessitates the specialist's presence on-site. Image processing has demonstrated itself as an appropriate substitute for traditional detection of cracks.

The challenges encountered in image-based identification arise from the unpredictable shapes and variable sizes of cracks, as well as the presence of shadowing in the captured images.

Identifying the crack location is essential for assessing the practicability and longevity of concrete. Physically identifying the fracture in an image is time-consuming and ineffective in large-scale situations.

This paper uses image processing to detect the existence of cracks, classifying images as cracked or not cracked, and also indicates the location of the crack.

Some experts have grouped crack detection techniques into 4 groups.

1) Integrated method
2) Real-world practical method
3) Method based on percolation
4) Approach based on morphology

A, C, and D on the list above are automatic, while the practical method based on the route finder algorithm is only partly automatic. We can use deep learning to find cracks. ImageNet is a collection of pictures that is structured based on the Wordnet structure and has hundreds of thousands of pictures in it.

METHODOLOGY

This section explains how the code works.

```
import cv2 as cv
import numpy as np
```

We utilize cv2 and numpy libraries to manipulate the images and the image data. Cv2 can handle images and video processing while numpy handles the numberic data/mathematical formulas for adjusting and identifying cracks.

```
def adjust_contrast(image, alpha, beta):
    # Image scaling
    result_image = cv.convertScaleAbs(image, alpha=alpha, beta=beta)
    return result_image
```

adjust_contrast utilizes the coverscalAbs function to perform scaling. So, the method is used to perform scaling, this function performs scaling and converts the resulting floating-point values to absolute values (unsigned integers). It ensures that the pixel values remain within the valid range (0 to 255 for 8-bit images).

```
def filter(image):
    # Histogram equalization
    result_image = cv.equalizeHist(image)

    # Bilateral filtering
    result_image = cv.bilateralFilter(result_image, d: 5, sigmaColor: 75, sigmaSpace: 75)
    return result_image
```

Purpose and functionality: This function performs two main operations: histogram equalization and bilateral filtering.

Histogram equalization enhances the contrast of the image by redistributing the intensity values. It improves the global contrast of the image.

Bilateral filtering is a non-linear, edge-preserving filter that smooths the image while preserving edges. It takes into account both spatial and intensity differences when filterin.

```
def gamma_correction(image, gamma):
    # Gamma correction
    lookUpTable = np.empty( shape: (1, 256), np.uint8)
    for i in range(256):
        lookUpTable[0, i] = np.clip(pow(i / 255.0, gamma) * 255.0, a_min: 0, a_max: 255)
    res = cv.LUT(image, lookUpTable)
    return res
```

Purpose and functionality: This function performs gamma correction on the input image. Gamma correction is a nonlinear operation used to adjust the brightness and contrast of an image. It involves raising the intensity values of the image to the power of the gamma value.

The lookup table (lookUpTable) is created to map the input intensity values to their corrected values based on the gamma value.

The cv.LUT() function applies the gamma correction using the lookup table.

```
def invert_image(image):
    # Image inversion
    return ~image
```

Purpose and functionality: This function inverts the intensity values of the image. It essentially subtracts each pixel value from the maximum intensity value (255 in this case), effectively inverting the brightness values.

```
def thresh_otsu(image):
    # Otsu thresholding
    th, image_otsu = cv.threshold(image, thresh: 0, maxval: 255, cv.THRESH_OTSU)
    return image_otsu
```

Purpose and functionality: This function performs Otsu's thresholding method to automatically determine the optimal threshold value for binarization. Otsu's method calculates the threshold that minimizes the intra-class variance of the black and white pixels in the image, effectively separating foreground from background.

```
def draw_contours(image):
    # Draw contours
    contours, hierarchy = cv.findContours(image, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

    return contours
```

Purpose and functionality: This function finds contours in a binary image. Contours are curves joining continuous points along the boundary of objects in the image. The cv.findContours() function detects contours using the specified retrieval mode (cv.RETR_TREE) and contour approximation method (cv.CHAIN_APPROX_SIMPLE).

```
def remove_contours(image, contours, contour_area):
    # Remove single-point noises from the image
    mask = np.ones(image.shape, dtype=np.uint8) * 255

    for c in contours:
        if cv.contourArea(c) < contour_area:
            cv.drawContours(mask, contours: [c], -1, color: 0, -1)

    image_result = cv.bitwise_and(image, image, mask=mask)
    return image_result
```

Purpose and functionality: This function removes small contours (noise) from the image based on their area. It creates a binary mask where areas corresponding to small contours are set to zero (black), effectively removing them from the image.

```
def remove_contours(image, contours, contour_area):
    # Remove single-point noises from the image
    mask = np.ones(image.shape, dtype=np.uint8) * 255

    for c in contours:
        if cv.contourArea(c) < contour_area:
            cv.drawContours(mask, contours: [c], -1, color: 0, -1)

    image_result = cv.bitwise_and(image, image, mask=mask)
    return image_result
```

Purpose and functionality: This function removes small contours (noise) from the image based on their area. It creates a binary mask where areas corresponding to small contours are set to zero (black), effectively removing them from the image.

```
def draw_contour_on_image(image, image_bw_marked):
    # Mark crack in the image
    contours, hierarchy = cv.findContours(image_bw_marked, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    image_result = cv.drawContours(image, contours, -1, color: (0, 0, 255), thickness: 2)

    return image_result
```

Purpose and functionality: This function draws contours on the original image to mark specific features or objects. It finds contours in a binary image (image_bw_marked) and draws them on the original image (image) with a specified color (red) and thickness. This is often used for visualizing detected objects or features in an image.

Algorithm

These techniques convert the vibrant picture into a binary image in which the remainder of the image is black and the areas of cracks are white highlights. First, turn the picture grayscale so you can accomplish this.

This phase reduces the three RGB channels into one channel image. The morphological technique of thresholding then is applied. Depending on a given thresholding value, the thresholding process converts a grayscale image into a binary image that is, each pixel has either 1 or 0 value. Thresholding creates an image in which the brighter pixels are colored black and the darker pixels are tinted white. This produces fissures and other dark areas that in the binary image seem white. Following preprocessing, Matlab's Euclidean Distance Transformation tool computes the crack width.

The image-based crack detection system begins with pixel analysis meant to find flaws or cracks. It starts with image preparation, which improves contrast and quality to help to distinguish features. Following that, edge detection techniques that concentrate on color or brightness variations help to find possible fissures. Texture analysis helps to differentiate cracks from other natural components such as shadows or surface patterns. Furthermore, machine learning methods can be taught on photos with known fracture sites, hence enhancing the accuracy of the detection. At last, the technique generates a binary picture pointing out the size and location of every found fracture. Working with different kinds of photographs allows one to improve its accuracy and dependability by means of adjustments and fine-tuning.

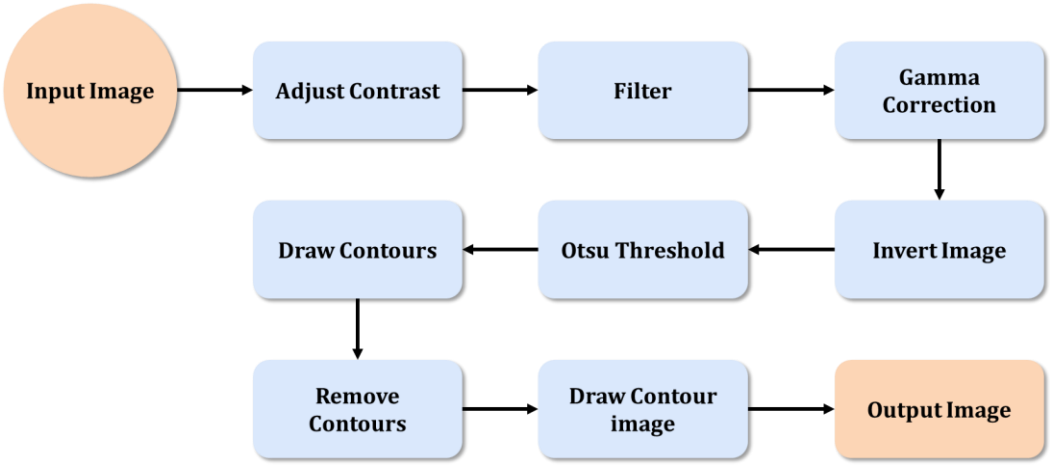

FIGURE 1: Image Processing Pipeline for Contour Detection and Enhancement.

RESULT

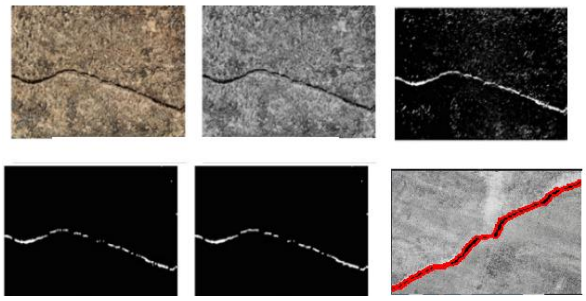

FIGURE 2: Crack Detection Process.

The original image was converted to greyscale after filtering, with the crack highlighted using red lines for clear visualization.

FURTHER SCOPE

The current approach utilizes Convolutional Neural Networks (CNN) and Artificial Neural Networks (ANNs) for crack detection in concrete through image processing. With future advancements, this technique can be extended beyond static image analysis to include:

a. video analysis
b. enabling real-time crack identification.
c. To monitor crack propagation, the code can be changed to provide an accurate crack position on the structure.

CONCLUSION

Structural inspection is crucial for determining the longevity of structures and even prolonging it. Cracks diminish both the visual appeal of a structure and its load-bearing capacity. This research investigates the possibilities of employing machine learning methods and algorithms for image processing to identify and categorize cracks in concrete as either cracked or NOT-cracked. Manual crack detection is labor-intensive and necessitates an expert, resulting in a deficiency of objectivity. Image processing is suggested as a viable option. The challenges encountered in image-based fracture detection encompass the unpredictable morphology and inconsistent dimensions of cracks, together with diverse interferences such as uneven illumination and shade. The difficulties in deep learning lie in the complexity of training extremely deep neural networks. The model may be further enhanced to detect cracks in real-time by merging it with cameras. Learning through reinforcement (where an algorithm identifies the optimal method to execute or accomplish an objective through trials and errors).

Crack detection based on image color is a method used to automatically identify and classify cracks in different surfaces such as roads, bridges, buildings, and pavements. This technique involves analyzing the color information in images to distinguish cracks from the surrounding surface area.

One significant benefit of using image color for crack detection is its ability to capture detailed aspects of a crack's appearance, including its shape, size, and severity. By analyzing color patterns within an image, even subtle variations can reveal the presence of a crack.

The process of crack detection through image color involves several key steps. Initially, the image undergoes preprocessing to enhance the contrast between the crack and its surrounding surface. This may involve adjusting brightness and contrast, applying edge-enhancing filters, or normalizing the color distribution across the image.

Following this, a segmentation algorithm isolates the crack from the background by identifying areas of the image that correspond to cracks based on their color. Since cracks may exhibit different color characteristics, it's crucial to employ a flexible segmentation technique that can adapt to these variations.

After segmentation, the cracks are classified according to their color features. This might involve comparing the crack's color distribution to a predefined color model or leveraging machine learning algorithms to automatically classify cracks based on their color properties.

In summary, crack detection through image color is a powerful method for recognizing and classifying cracks on various surfaces. This approach allows for greater accuracy and efficiency compared to traditional methods, aiding in better maintenance practices, prolonging infrastructure longevity, and improving public safety.

REFERENCES

[1] Munawar, H. S., Hammad, A. W. A., Haddad, A., Soares, C. A. P., & Waller, S. T. (2021). Image-based crack detection methods: A review. Infrastructures, 6(8), 115. https://doi.org/10.3390/infrastructures6080115

[2] Mohan, A., & Poobal, S. (2017). Crack detection using image processing: A critical review and analysis. Alexandria Engineering Journal, 56(4), 729–738. https://doi.org/10.1016/j.aej.2016.10.010

[3] Zhou, X. X., & Cui, X. Y. (2024). A review of computer vision-based crack detection methods in civil infrastructure. Remote Sensing, 16(16), 2910. https://doi.org/10.3390/rs16162910

[4] Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural networks. IEEE Transactions on Intelligent Transportation Systems, 17(8), 2279–2290. https://doi.org/10.1109/TITS.2015.2509260

[5] Hu, K., Ling, Y., & Liu, J. (2025). Image recognition technology for bituminous concrete reservoir panel cracks based on deep learning. PLoS ONE, 20(2), e0318550. https://doi.org/10.1371/journal.pone.0318550

[6] Li, W. B., & Yang, B. C. (2015). Detection of crack width of concrete bridge based on image processing technology. Hunan Communication Science and Technology.