"Crack Property Analysis Using Image Processing Techniques"

A Minor Project Report

Submitted in Partial Fulfillment of the Requirements for the degree

of

Bachelor of Technology

IN

INSTRUMENTATION AND CONTROL ENGINEERING

By

Deep Khut (19bic008)

Karan Padariya (19bic033)

Under the Guidance of

Harsh Kapadiya

ELECTRONICS AND INSTRUMENTATION ENGINEERING DEPARTMENT
SCHOOL OF TECHNOLOGY
NIRMA UNIVERSITY
Ahmedabad 382 481

December 2022

CERTIFICATE

This is to certify that the project report entitled "**Crack Property analysis using Image Processing Techniques**" submitted by Deep Khut (19bic008) & Karan Padariya (19bic033) towards the partial fulfillment of the requirements for the award of the degree in bachelor of Technology (INSTRUMENTATION & CONTROL ENGINEERING) Of School of Technology is the record of work carried out by them under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this project work to the best of my/our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

"Harsh Kapadiya"

Assistant Professor
Dept. of Electronics and Instrumentation Engineering
Institute of Technology, Nirma University

Head of Department (EI)

DATE: 13th Dec 2022

ACKNOWLEDGEMENT

The development of this project was assigned under the guidance of Prof. Harsh Kapadia who guided us through this project from the beginning to the end.

He helped us in optimizing our code as well as gave indepth knowledge of some concepts related to image processing.

We would also like to thank Dr. Dipak Adhyaru and Dr. Ankit Sharma for helping and guiding us to develop some special algorithm to get an proper idea for measurement of crack.

ABSTRACT

Structures that have stood for a long time and is still operational now for several years are referred to be aging structures. The crack is an early signal of an aging structure along with its analysis of measure, can be determine the life prediction of an structure. Crack Monitoring is usually done by visual inspection, which can occasionally result in inconsistent data. The Authors have worked to extract the length, width with area and depth of the crack from the image. The Authors have put forth an algorithm for measuring and analysing cracks that make use of bilateral filtering, connected-component labelling, and thresholding, among other image processing methods. The main goal of this development is to reduce human error while spotting cracks in diverse structures. For the most accurate findings, analysis of the collected data was done by removing the background noise to accurately measure the crack's length and width. To check the viability of this algorithms, authors performed experiment using pictures of genuine concrete surface used to examine crack properties.

Keywords: Crack Extraction, Image Segmentation, Crack width, Crack Length, Thresholding

Concepts:

Computing methodologies → Image Processing

LIST OF FIGURES

CONTENTS

Chapter 1 Introduction

In recent years we have witnessed a rise in interest in the measurement of crack on concrete structures. Important indicators of the structural integrity and health of a concrete element include the creation, breadth, and propagation of cracks. Traditionally, crack inspection is done by specialists visually, using a crack scale and a loupe. Methods of automatic crack detection were done conventionally through image processing [4]

Important details regarding a concrete block's durability and reasons of degradation can be learned via crack width analysis. Manual inspection, however, is frequently time-consuming, making quantitative analysis impractical. As a result, the significance of applying image processing for visual assessment has continuously grown.

Related works in automatic crack detection using image processing have been reported in [2-4]. Also, in [5,6], proposed an image processing technique for crack detection using percolation model which considers the relationship with neighbouring pixels. The region and border of images can now be directly and naturally expressed using image segmentation techniques based on the level set. As a result, these techniques are more in line with the notion of picture segmentation. Additionally, level set results do not require post-processing such as region merging and edge joining[7].

Non-destructive testing is an inspection technique that allows materials or components to be examined without changing their original shapes or functions. It also refers to all inspection methods to examine a product's properties, conditions, and internal structures without disassembling or destroying it. Non-destructive testing is an inspection technique related to the measurement of crack depth in structures. Non-destructive testing methods include radiographic testing(RT), ultrasonic testing (UT), magnetic practice testing (MT), penetrant testing (PT), eddy current testing (ET), leak testing (LT), and infrared thermography (IRT), and so on, and the demand is fast growing with the development of high-value-added industries [8].

The objective of this work is to develop an algorithm which will help in the inspection and maintenance of concrete surfaces that extract information like measurement of crack length and width. This work targets to design an measurement algorithm that uses different image processing techniques. Engineers and inspectors' manual work will be replaced by automated work in the hopes that this will produce results that are consistent. The system will analyse the data and present the findings. Engineers and inspectors of old buildings will gain a lot from this study because automation will replace their manual labour and produce correct results quickly.

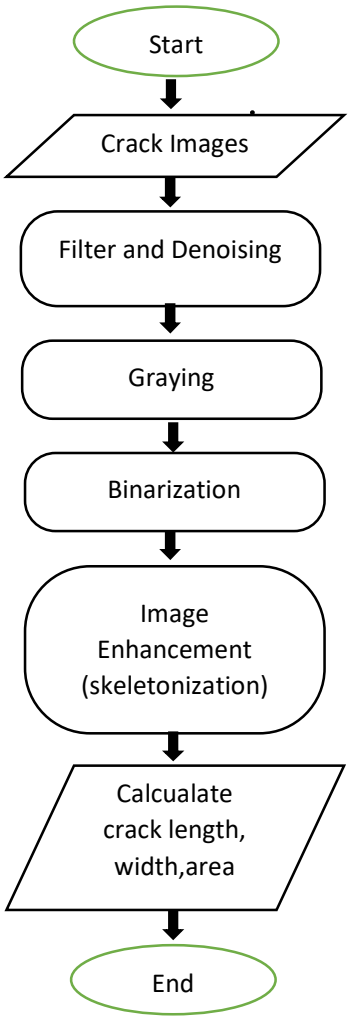
Chapter 2: Research Methodology

In this project the reseach methodology we have used kinf of Quantitative Methodology. This type of research methods gather numerical data and measures things as they are. It aims to summarise the state of a specified variable right now. The introduction, method, results, and conclusion are the four main sections that make up the quantitative methodology.

The difficulty in finding fractures in aged buildings when utilising solely visual examination is the research challenge. This is the most typical technique used when inspecting an old building, which can lead to differences in the information gathered by different engineers. Additionally, there is currently no hardware implementation specifically for fracture detection on ageing buildings, despite the ongoing advancements in technology.

The theoretical framework provides an overview of the study's fundamental methodology. To accomplish the study's goals, the researchers suggest that many concepts be combined. The input will be a sample image of the wall's structure, which will then be processed using the hardware to apply the suggested crack identification and measuring method. The structural health information, which includes the fracture width and the number of cracks on that particular picture, will be the system's final output after processing the image.

Chapter 3: Flow Chart



3.1 Pre-processing

In order for subsequent operations to produce better results, a number of image pre-processing techniques must be used.

1) RGB to Grayscale Conversion: An image is considered to be a multi-dimensional and multi-channel signal. Since a grayscale picture just has one channel and can represent each pixel with a single value, processing may be made simpler by converting colourful images to grayscale.

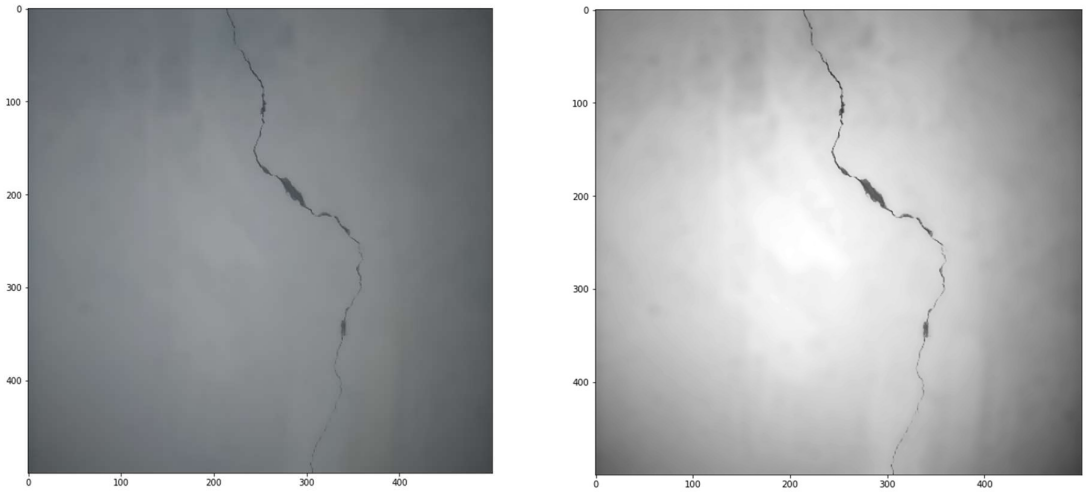


Fig2. RGB to Gray image

2) Smoothening filter and Inversion: Since the surface is not always smooth, edge recognition algorithms may mistakenly identify the roughness as an edge. As a result, the edge detection process is performed before the Gaussian smoothening filter. This will guarantee that the uneven surface is smoothed out, reducing noise detection.
3) Histogram Equalization: This method is used to adjust image intensities to properly process the image. This will make the image clearer.

3.2 Crack Extraction

1) Binary Thresholding: To simplify the output image into abinary image ith, a threshold value ith is compared to the gradient magnitude.

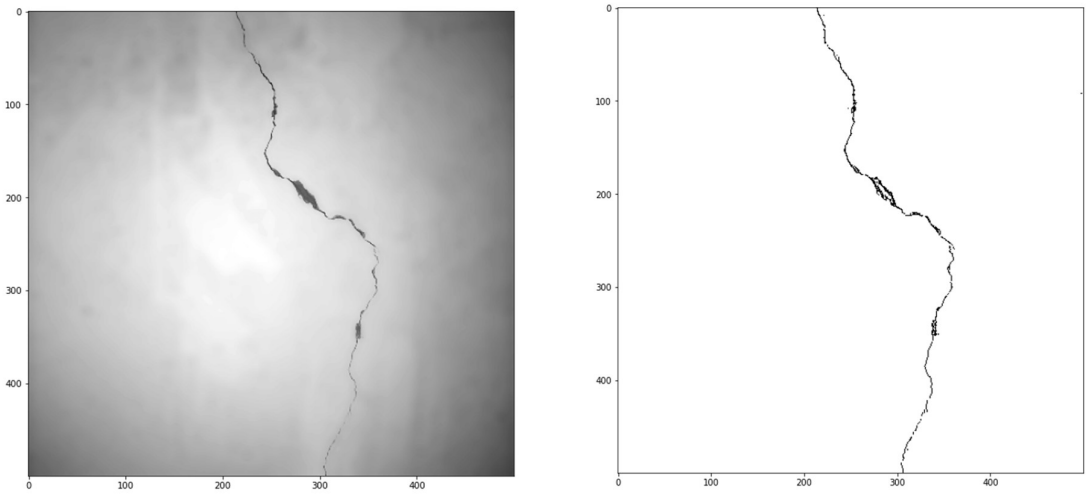


Fig3. Adaptive thresholding

2) Edge Detection: The edge detection is done by the Sobel filter. While [9] suggested a number of edge detection techniques, the Sobel filter is selected for this study due to it being computationally inexpensive.

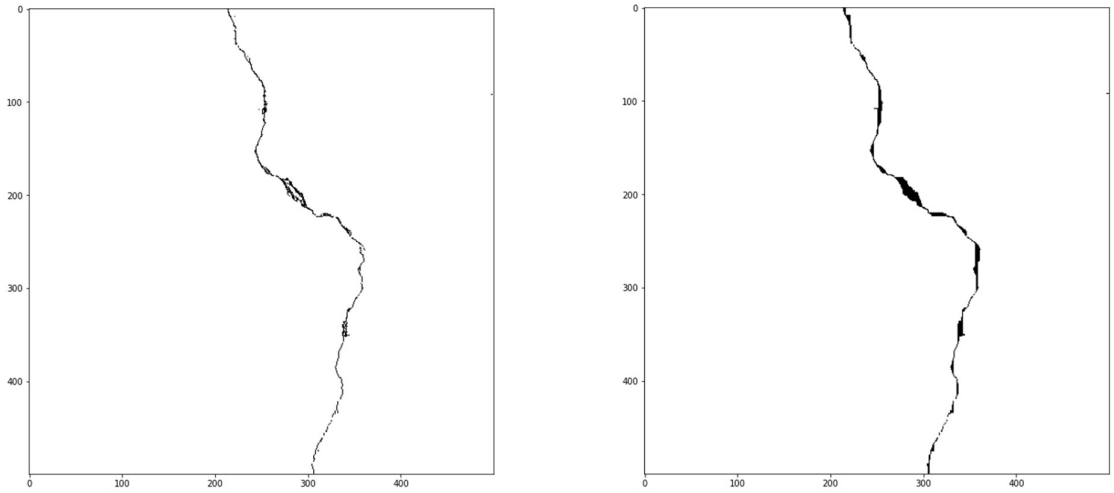


Fig4. Morphological operation(Opening)

3) Labeling: Labeling is an operation where the connected pixels are considered as one component. Through labeling, the number of components can be counted and identified. Each detected component can be considered a single crack which can be further processed to calculate its width.

Chapter 4: Implementation of algorithm

Psuedo-code of the proposed algorithm

1. Read the RGB original cracked concrete surface image.

Fig5. Crack RGB Image

2. Smoothing using Bilateral filter in CV2

```
img = cv2.bilateralFilter(img, 101, 500, 500)
```

Bilateral filter is used for smoothening images and reducing noise, while preserving edges

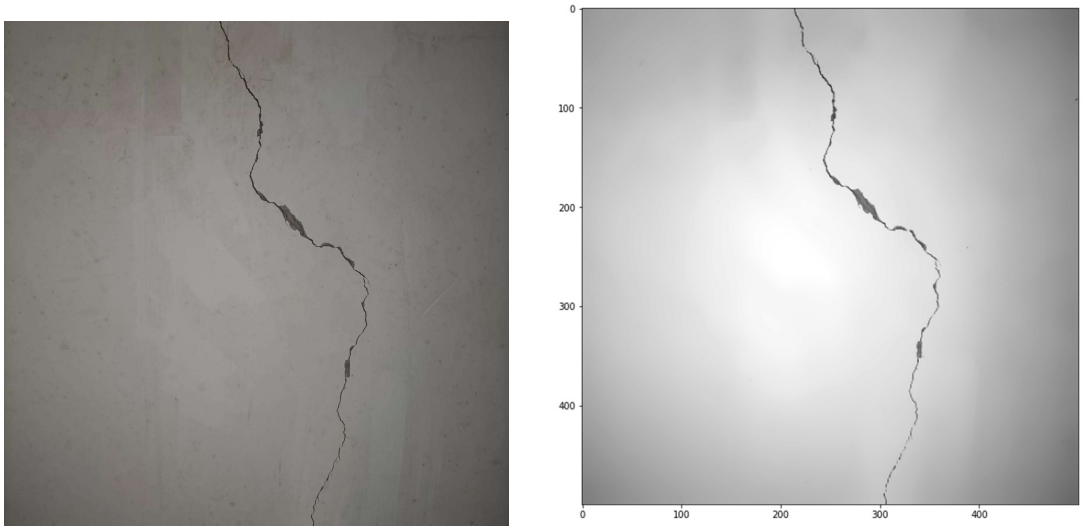


Fig6. RGB to Gray

3. Converting into gray scale

```
gray_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

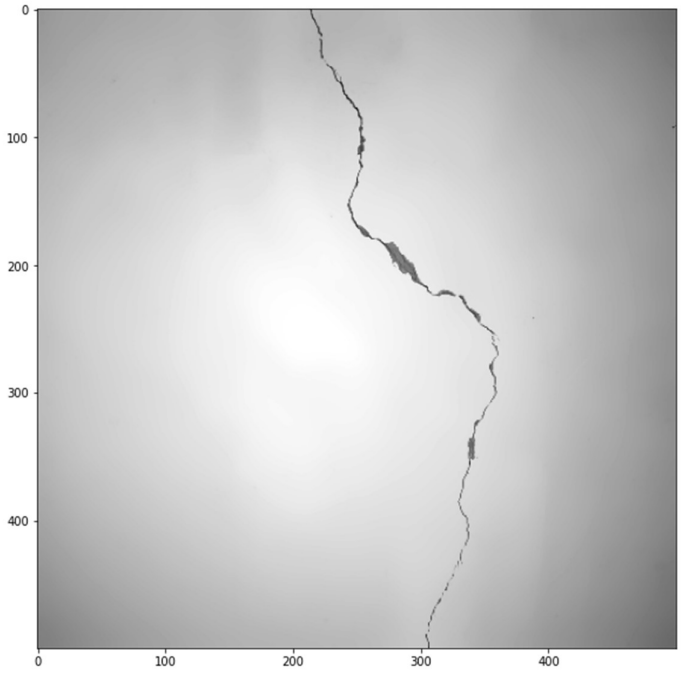


Fig7. Gray Image

4. Binary Thresholding using adaptive threshold using Adaptive_threshold

```
bin_img = cv2.adaptiveThreshold(gray_img,255,cv2.ADAPTIVE_THRESH_GAU
SSIAN_C,cv2.THRESH_BINARY,31,2)
```

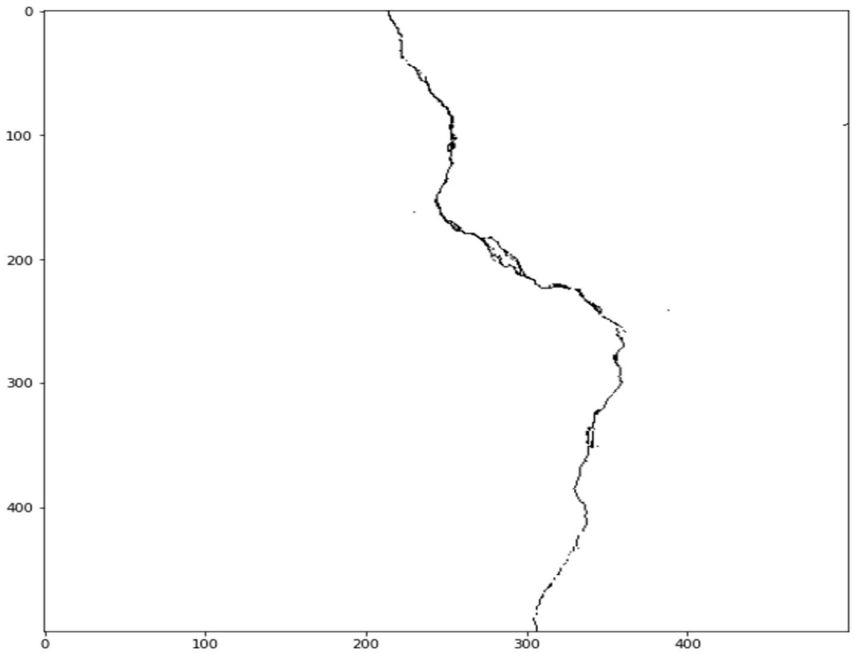


Fig8. Thresholded Image

5. Morphological Operation – Opening for better highlighting the shape of actual crack.

```
kernel = np.ones((11,11), np.uint8)

opening = cv2.morphologyEx(bin_img, cv2.MORPH_OPEN, kernel)
opening = cv2.morphologyEx(opening, cv2.MORPH_OPEN, kernel)
plt.figure (figsize= (10,10))
```

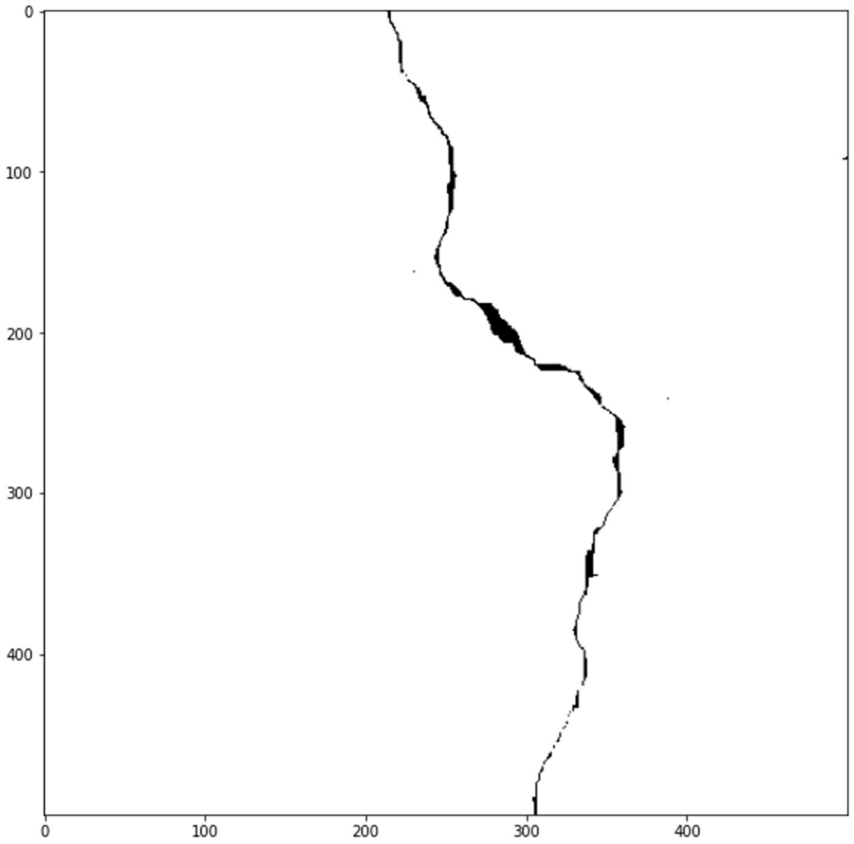


Fig9. Opening Image

6. Skeletonization using coustom made function.

```
def find_skeleton1(thresh):
    skeleton = np.zeros(thresh.shape,np.uint8)

    # _,thresh = cv2.threshold(img,127,255,0)

    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))

    iters = 0
    while(True):
        eroded = cv2.erode(thresh, kernel)
        temp = cv2.dilate(eroded, kernel)
        temp = cv2.subtract(thresh, temp)
        skeleton = cv2.bitwise_or(skeleton, temp)
        thresh = eroded.copy()
```

```
        iters += 1
        if cv2.countNonZero(thresh) == 0:
            return (skeleton,iters)
```

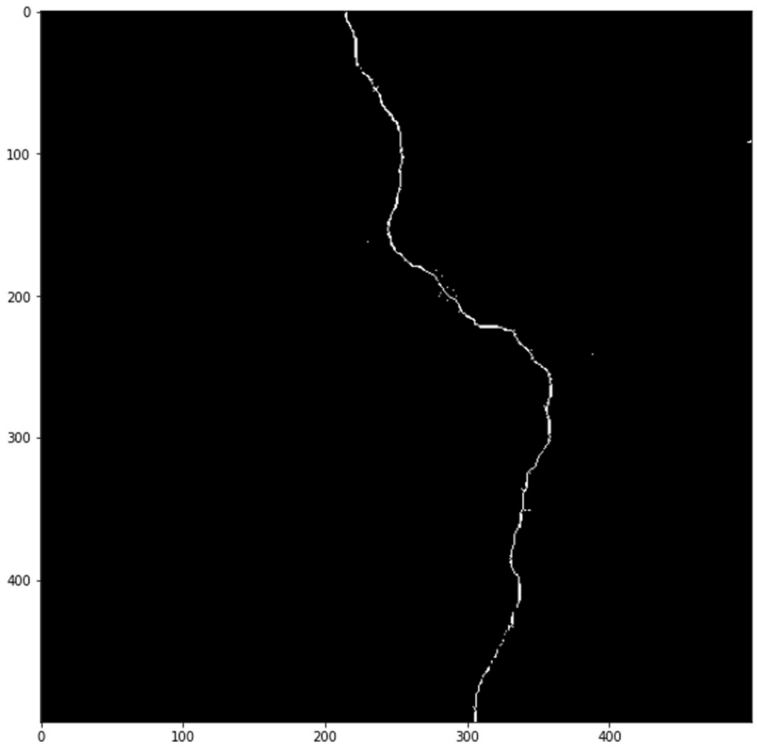
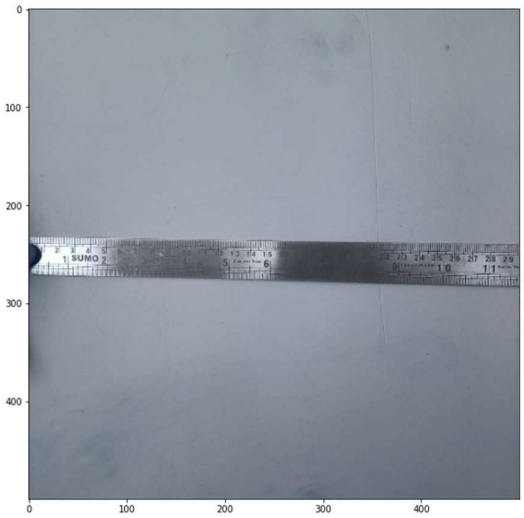


Fig10. Skeletonized Image

7. Length in terms of Pixels and mm

```
print("length:",np.sum(area_array),"pixels")
print("length:",round(np.sum(area_array)*0.592,2),"mm")
```

```
length: 690.0 pixels
length: 408.48 mm
```



➔ For measurement of length we first need the width of one pixel.
➔ For that we took an image of scale having 29.6 cms which is equal to 500 pixels.
➔ So, by default our calibration factor for measuring length is 296mm/500pixels = 0.592 which is the length of one pixel.

8. For measuring Width in terms of mm and pixels

```
edges = cv2.Canny(opening,100,200)
# for i in range(0,500):
#   for j in range(0,1):
#       edges[i,j]

width_array=[]
for i in range(0,500):
  line = edges[i,0:500]
  x=[]
  for j in range(0,len(line)):
        if line[j]==255:
          x = np.append(x, j)
  if len(x)>1:
    crack_width = x[len(x)-1]-x[0]
    width_array = np.append(width_array,crack_width)
  else:
    width_array = np.append(width_array,0)
```

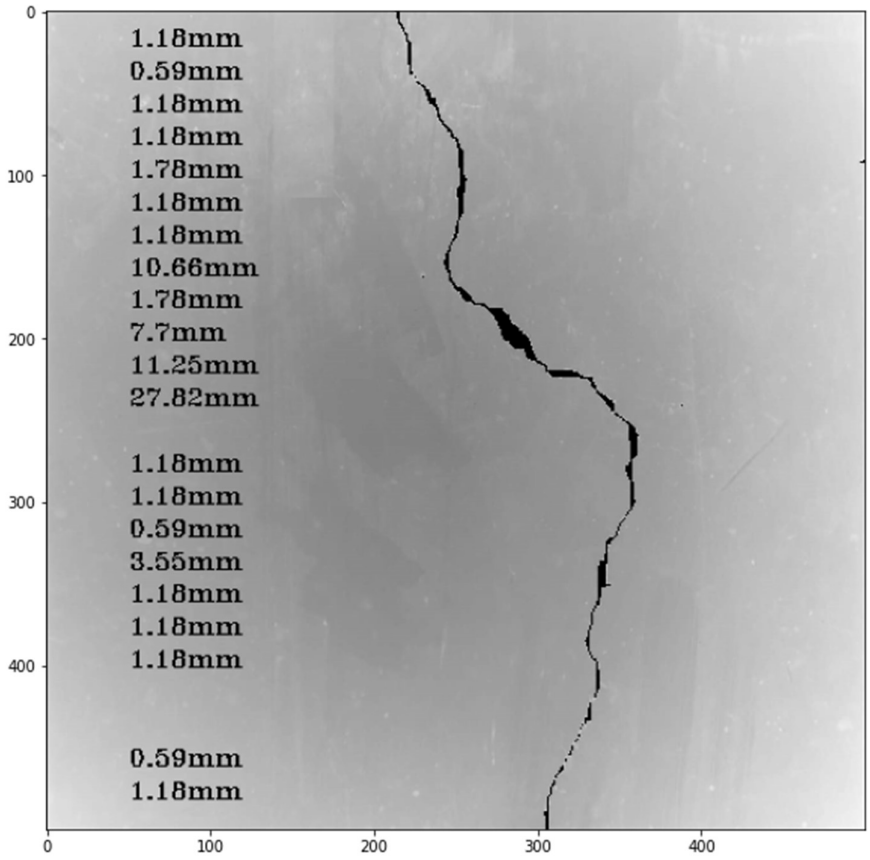


Fig11. Image with row-wise width

For calibrating width in terms of mm

```
final_img = img.copy()
for i in range(1,500,20):
  if width_array[i]>0:
    cv2.putText(img=final_img, text=str(round(int(width_array[i])*0.592
,2))+"mm", org=(50, i), fontFace=cv2.FONT_HERSHEY_TRIPLEX, fontScale=0.
5, color=(0, 0, 0),thickness=1)
    # 1 pixel = 0.592 mm
plt.figure (figsize= (10,10))
plt.imshow(final_img)
```

➔ For highlighting the crack we use countour overlay

```
final_img =cv2.imread("/content/drive/MyDrive/Crack Images/8.jpg",1)

gray = cv.cvtColor(final_img, cv.COLOR_BGR2GRAY)
ret, binary = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV | cv.THRE
SH_OTSU)

se = cv.getStructuringElement(cv.MORPH_RECT, (10, 10), (-1, -1))
binary = cv.morphologyEx(binary, cv.MORPH_OPEN, se)
# cv.imshow("binary", binary)
```

```
contours,hierachy=cv.findContours(binary,cv.RETR_EXTERNAL,cv.CHAIN_APPR
OX_SIMPLE)
height, width = final_img.shape[:2]
for c in range(len(contours)):
    x, y, w, h = cv.boundingRect(contours[c])
    area = cv.contourArea(contours[c])
    if h > (height//2):
        continue
    if area < 150:
        continue
    cv.rectangle(final_img, (x, y), (x+w, y+h), (0, 0, 255), 1, 8, 0)
    cv.drawContours(final_img, contours, c, (0, 255, 0), 1, 8)

# cv.imshow("result", final_img)
# cv.imwrite("result.jpg", final_img)

# cv.waitKey(0)
# cv.destroyAllWindows()
plt.figure (figsize= (10,10))
plt.imshow(final_img)
```

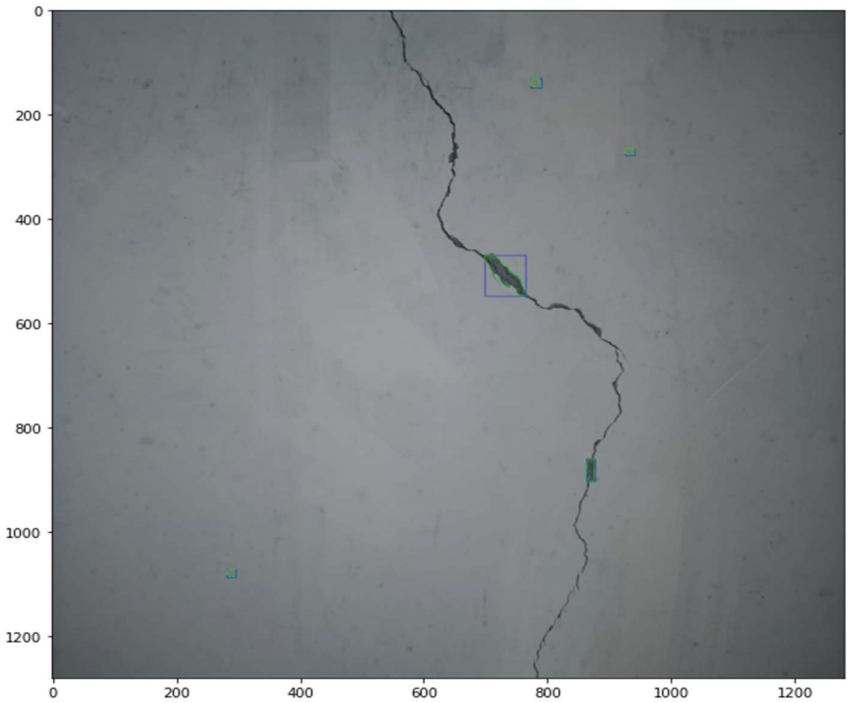


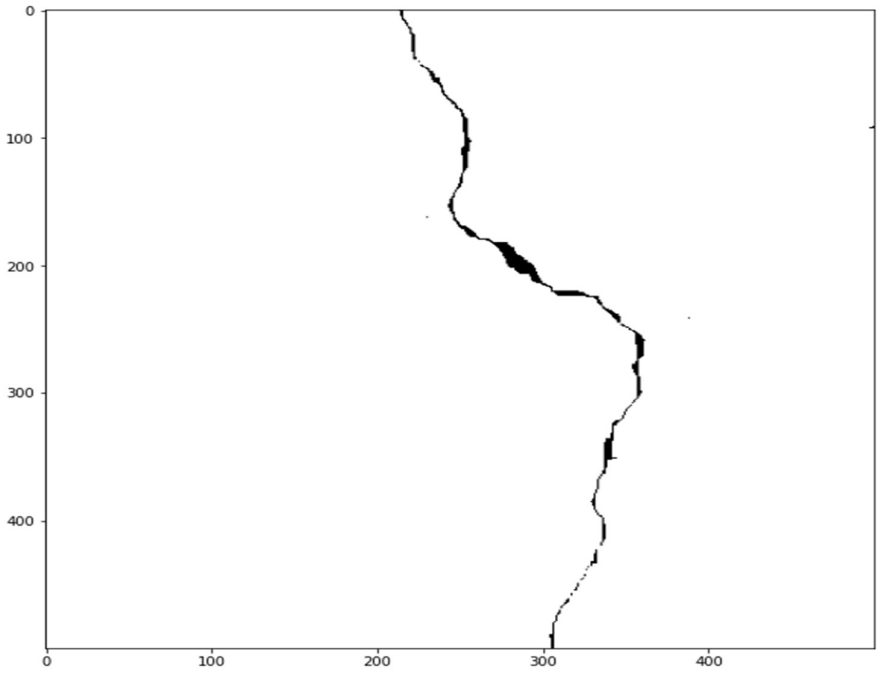
Fig12. Highlighted crack using Overlay

9. For calculating area in terms of mm

```
import math
area = 0
```

```
for i in range(0,500):
  for j in range(0,500):
    if bin_img[i,j]==0:
      area = area+1

print("Area:",round(area*0.592*0.592,2),"mm sq")
```

Area: 503.97 mm sq



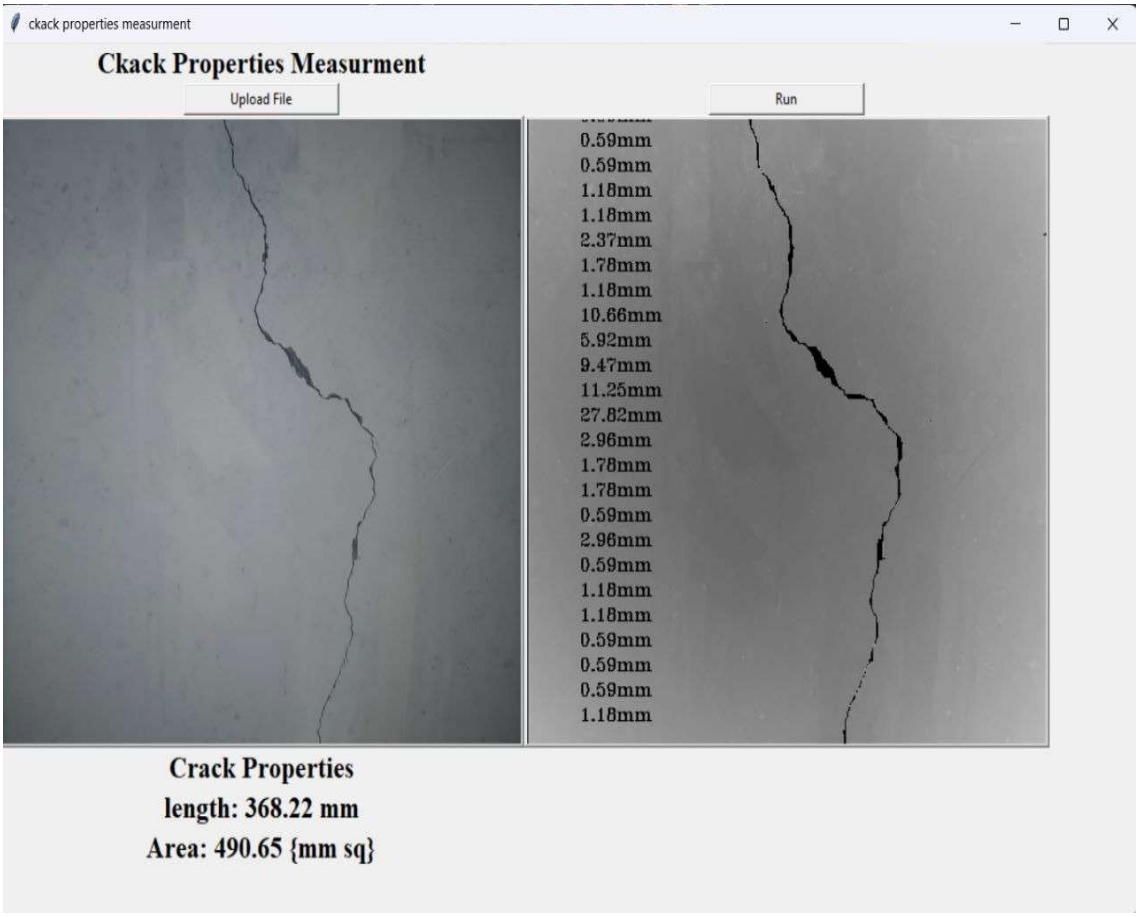
GUI Implementation

- ➔ Developed an GUI Application in which we can directly select the image from desired location or click the image using camera.
- ➔ To get accurate results, user need to provide the distance through which the image is been taken. As it will be needed to calibrate the crack Properties in terms of milimeter.
- ➔ Through this application user can get details of crack properties and also get the statistics of Average, Max and Min width.

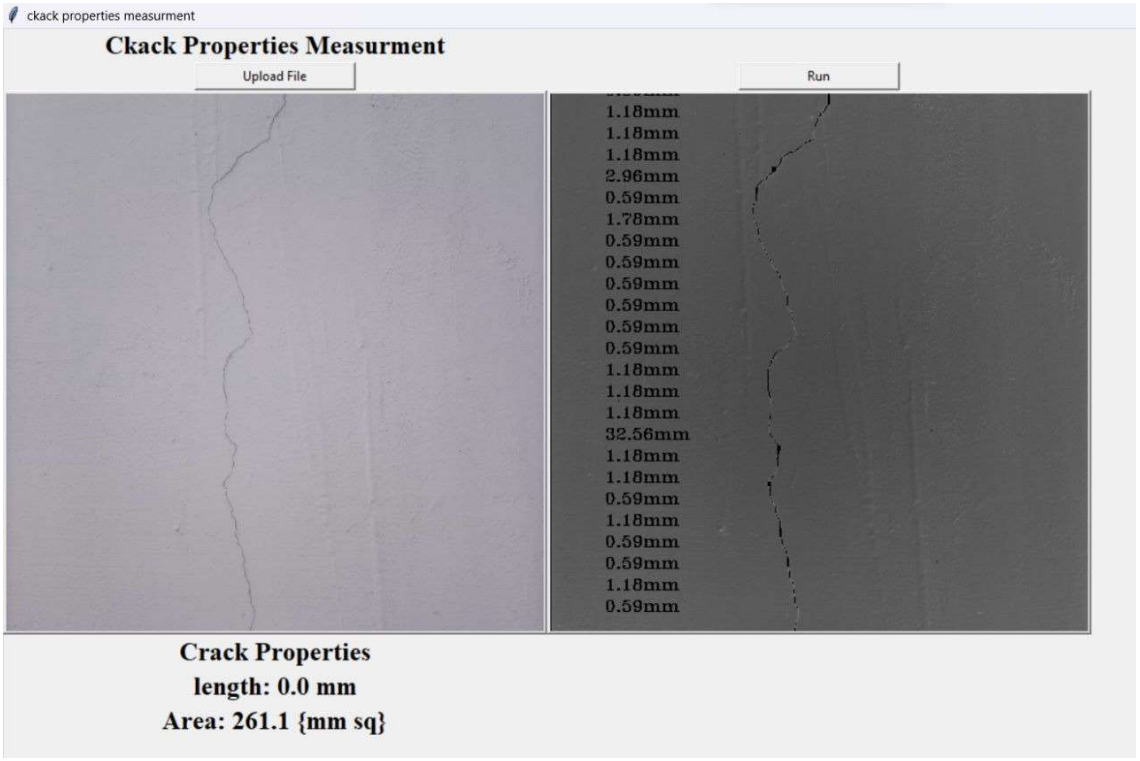
```
from typing import Any

import tkinter as tk

from tkinter import filedialog

from tkinter.filedialog import askopenfile

from PIL import Image, ImageTk

import numpy as np

import cv2

import matplotlib.pyplot as plt

import math

my_w = tk.Tk()

my_w.geometry("1000x600")  # Size of the window

my_w.title('ckack properties measurment')

my_font1=('times', 18, 'bold')

l1 = tk.Label(my_w,text='Ckack Properties Measurment',width=30,font=my_font1)

l1.grid(row=1,column=1)

b1 = tk.Button(my_w, text='Upload File',

  width=20,command = lambda:upload_file())

b1.grid(row=2,column=1)

def upload_file():

   global img,imge

   f_types = [('Jpg Files', '*.jpg')]

   global filename

   filename = filedialog.askopenfilename(filetypes=f_types)

   img = cv2.imread(filename)
```

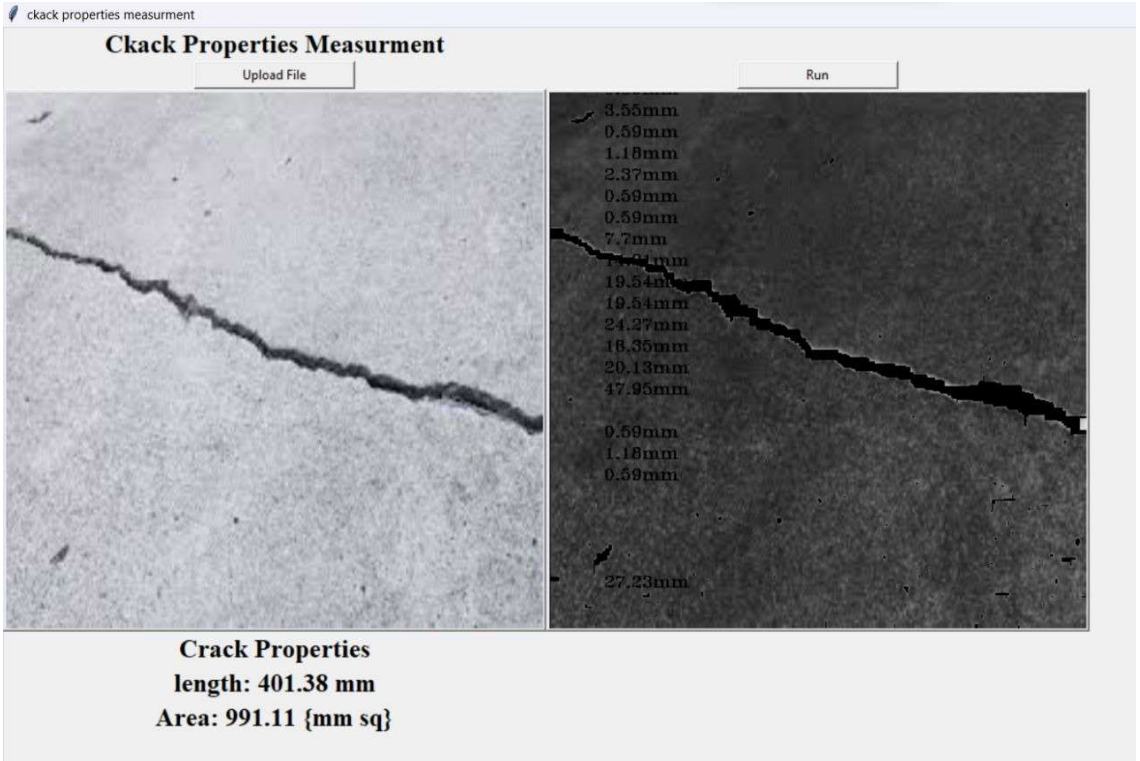
```
        img = cv2.resize(img, (500,500))
        imge =  Image.fromarray(img.astype('uint8'), 'RGB')
        imge=ImageTk.PhotoImage(imge)
        b2 =tk.Button(my_w,image=imge) # using Button
        b2.grid(row=3,column=1)
    b3 = tk.Button(my_w, text='Run',
               width=20,command = lambda:output_img())
    b3.grid(row=2,column=2)
    def output_img():
        global img, length, area

……….# image processing code as mentioned previously……..

    img=ImageTk.PhotoImage(img)
        b4 =tk.Button(my_w,image=img) # using Button
        b4.grid(row=3,column=2)

        my_font1 = ('times', 18, 'bold')
        l1 = tk.Label(my_w, text='Crack Properties', width=30, font=my_font1)
        l1.grid(row=4, column=1)
        # length = str(5)
        # length =("length:",str(round(np.sum(area_array)*0.592,2)),"mm")
        l1 = tk.Label(my_w, text=length, width=30, font=my_font1)
        l1.grid(row=5, column=1)
        # area = str(15)
        # area = ("Area:",str(round(area*0.592*0.592,2)),"mm sq")
        l1 = tk.Label(my_w, text=area, width=30, font=my_font1)
        l1.grid(row=6, column=1)
my_w.mainloop()  # Keep the window open
```

Results


ckack properties measurment
Ckack Properties Measurment
Upload File
Run
0.59mm
0.59mm
1.18mm
1.18mm
2.37mm
1.78mm
1.18mm
10.66mm
5.92mm
9.47mm
11.25mm
27.82mm
2.96mm
1.78mm
1.78mm
0.59mm
2.96mm
0.59mm
1.18mm
1.18mm
0.59mm
0.59mm
0.59mm
1.18mm
Crack Properties
length: 368.22 mm
Area: 490.65 {mm sq}


ckack properties measurment
Ckack Properties Measurment
Upload File
Run
1.78mm
1.78mm
2.37mm
33.15mm
1.78mm
2.37mm
1.78mm
31.5mm
23.68mm
1.78mm
14.8mm
12.43mm
2.37mm
1.78mm
25.46mm
2.96mm
31.97mm
Crack Properties
length: 433.94 mm
Area: 921.37 {mm sq}


ckack properties measurment
Ckack Properties Measurment
Upload File
Run
1.18mm
1.18mm
1.18mm
2.96mm
0.59mm
1.78mm
0.59mm
0.59mm
0.59mm
0.59mm
0.59mm
0.59mm
1.18mm
1.18mm
1.18mm
32.56mm
1.18mm
1.18mm
0.59mm
1.18mm
0.59mm
0.59mm
1.18mm
0.59mm
Crack Properties
length: 0.0 mm
Area: 261.1 {mm sq}


ckack properties measurment
Ckack Properties Measurment
Upload File
Run
3.55mm
0.59mm
1.18mm
2.37mm
0.59mm
0.59mm
7.7mm
19.54mm
24.27mm
18.35mm
20.13mm
17.95mm
0.59mm
1.18mm
0.59mm
27.23mm
Crack Properties
length: 401.38 mm
Area: 991.11 {mm sq}

Conclusion

In this work, the method of determining crack length width and area using the intensity values is successfully validated and verified by implementing the algorithm in 25+ images. Additionally we have successfully implemented an image-based non contact inspection method by developing an GUI in which user can get details within few clicks.

Reletively the traditional methods, sketching the crack pattern, and using crack guages, the main advantage of this algorithm are significant increase in: speed, efficiency, information, and reliability as automatic image processing is applied.

The drawback of this algorithm is, it will only suitable for laboratory test monitoring because the spatial resolution required for large structures, such as bridges and dams, does not allow low cost commercial cameras. And sometimes some surfaces are identified wrongly as cracks.

Future scope

The performed algorithm can replace the traditional method of measuring crack length width and area of the crack by hand using crack guages and through that information taken from it is transformed into drawing sheets, which is also labor-intensive and may be less accurate.

Future work needs focus on increasing the crack width detecting algorithm's precision and lowering false-negative errors. For further improvement of this algorithm, calibration can be done manually for different cases and for various distances between camera and the crack.

References

[1]. Japan Concrete Institute. Guideline of concrete crack inspection, repair, and reinforcement. Japan Concrete Institute; 2003.

[2]. Kawamura K, Miyamoto A, Nakamura H, Sato R.Proposal of a crack pattern extraction method from digital images using an interactive genetic algorithm. JSCE J 742/vi-60, p 115–141, 2003.

[3]. Roli F. Measure of texture anisotropy for crack detection on textured surfaces. Electronics Letters 1996;32:1274–1275.

[4]. Abdel-Qader I, Abudayyeh O, Kelly ME. Analysis ofedge detection techniques for crack identification in bridges. Journal of Computing in Civil Engineering 2003;17:255–263.

[5]. Yamaguchi T, Hashimoto S. Image processing based on percolation model. IEICE Trans Inf Syst 2006;E89-D:2044–2052.

[6]. Yamaguchi T, Hashimoto S. Automated crack detection for concrete surface image using percolation model and edge information. Proc IECON2006, p355–360.

[7]. M.Y. Yang, H. Ding, B. Zhao, Chan–Vese model image segmentation with neighborhood information, J. Comput. Aided Des. Comput. Graph. 23 (3) (2001) 413–418.

[8]. L. Cartz, Nondestructive Testing, ASM International, Geauga County, OH, USA, 1995.

[9]. L. S. Calderon and J. Bairan, “Crack Detection in Concrete Elements from RGB Pictures using Modified Line Detection Kernels,” in Intelligent Systems Conference, London, UK, 2017.

minor

ORIGINALITY REPORT

14%	10%	6%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

2%

★ repository.upenn.edu

Internet Source

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		