Image Processing

(3EC401ME24)

Special Assignment

Crack Detection On Road

Neel Patel

Parin Patel

Institute of Technology, Nirma University, Ahmedabad

Email: 22bec076@nirmauni.ac.in

Email: 22bec090@nirmauni.ac.in

INDEX

Crack Detection On Road

1.Introduction

1.1 Abstract:

Road infrastructure is essential for transportation and safety but is vulnerable to get damaged from environmental factors and heavy usage. Cracks on road surfaces are early signs of wear that, if neglected, can lead to severe damage and costly repairs. Traditional inspection methods are time-consuming and error-prone, prompting the need for efficient, automated solutions.

This project leverages image processing in MATLAB to create a system capable of detecting road cracks quickly and accurately. By analyzing road images, this approach enables timely maintenance and improved safety. MATLAB's powerful image processing tools make it an ideal platform for implementing and refining this crack detection system.

1.2 Keywords:

- Road Crack Detection
- Image Processing
- MATLAB
- Road Infrastructure Maintenance
- Automated Detection System
- Edge Detection
- Morphological Operations
- Image Segmentation

1.3 Objective:

The main objective of this project on "Crack on Road Detection" is to understand and apply Image Processing concepts using MATLAB software to detect cracks on road surfaces accurately. The project aims to develop a system that leverages MATLAB's image processing capabilities to identify and analyze road cracks in real-time. This will involve implementing algorithms to process and enhance images for effective crack detection and classification, thereby aiding in proactive road maintenance and improved safety.

1.4 Motivation:

The motivation behind this project is the pressing need for efficient and reliable methods of maintaining road infrastructure. Manual road inspection methods are not only labor-intensive but also subject to delays and inaccuracies, often resulting in the degradation of road quality before repairs are made. Cracks, as the earliest indicators of road wear, provide crucial information about the state of the infrastructure, making their timely detection essential for preventive maintenance.

Advances in image processing offer a promising solution to this challenge. By developing a crack detection system using MATLAB, this project seeks to harness the power of automation in infrastructure monitoring. This work also offers hands-on experience with MATLAB's image processing tools, making it an educational venture into both software proficiency and the practical applications of image analysis.

2.Algorithm/Flow of Code

2.1 Methodology:

- **Image Preprocessing**: The image is loaded, resized, and converted to grayscale. Gaussian blur is applied to reduce noise.

- **Binary Thresholding**: An adaptive thresholding technique is used to create a binary image, isolating dark cracks from the road background.

- **Morphological Operations**: Clean the binary image using morphological operations to remove small artifacts that aren't part of cracks.

- **Edge Detection and Filtering**: Apply edge detection (Canny method) to outline crack edges, and remove any small, irrelevant components that may cause false detections.

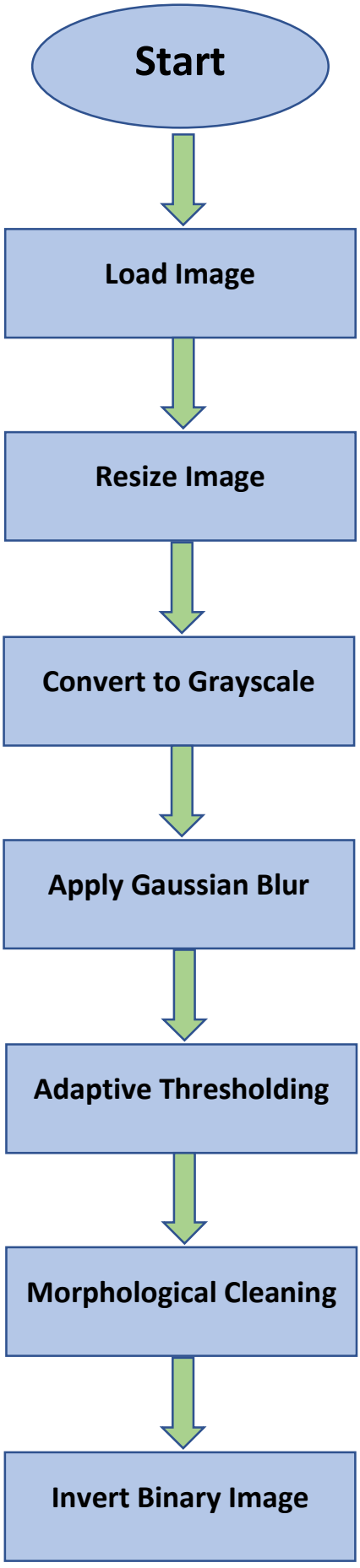
- **Bounding Box Detection**: Draw bounding boxes around detected cracks, marking the regions of interest in the original image.

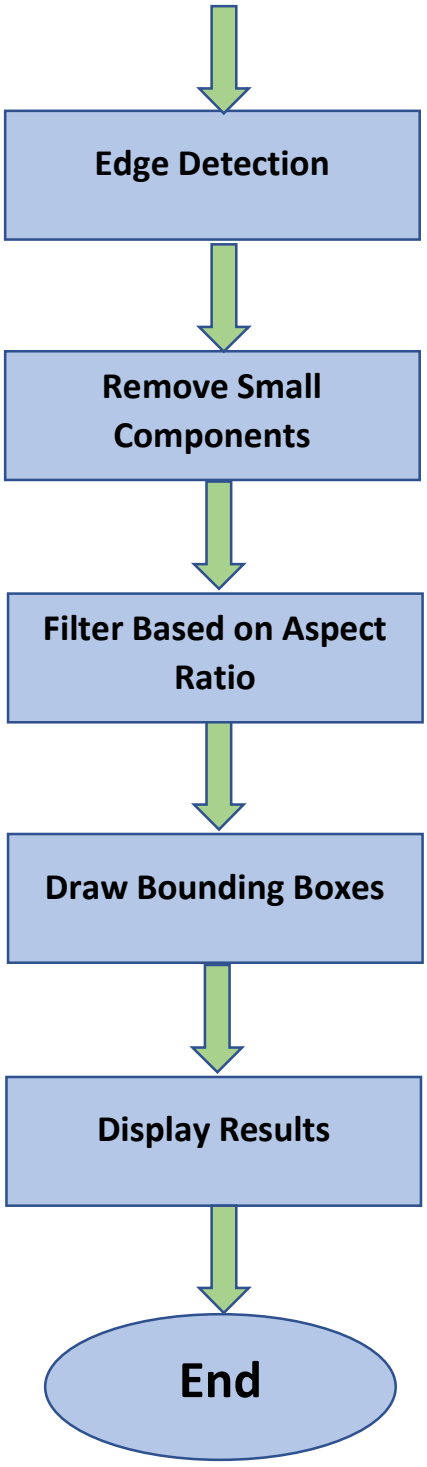
2.2 Flow of Code:

The code follows a sequence of steps to detect cracks on a road surface from an image using MATLAB. Here's a simplified flow:

1. **Load and Display Image**: Load the road image and display it.
2. **Resize (Optional)**: Resize the image to reduce processing time.
3. **Convert to Grayscale**: Convert the image to grayscale for easier processing.
4. **Apply Gaussian Blur**: Reduce noise by applying a Gaussian blur.
5. **Thresholding**: Convert the image to binary, highlighting potential crack areas.
6. **Morphological Operations**: Clean the binary image to remove small, irrelevant areas.
7. **Edge Detection**: Use the Canny edge detector to find edges in the image.
8. **Filter Small Components**: Remove small noise components to improve accuracy.
9. **Bounding Box Detection**: Draw bounding boxes around detected crack regions.

2.3 Flow Chart:



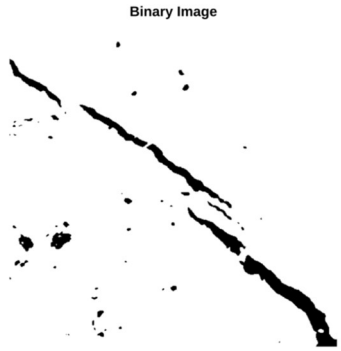

Edge Detection
Remove Small Components
Filter Based on Aspect Ratio
Draw Bounding Boxes
Display Results
End

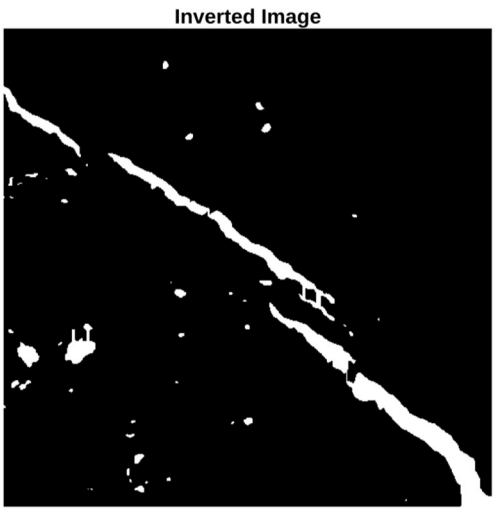
3.Literature Review

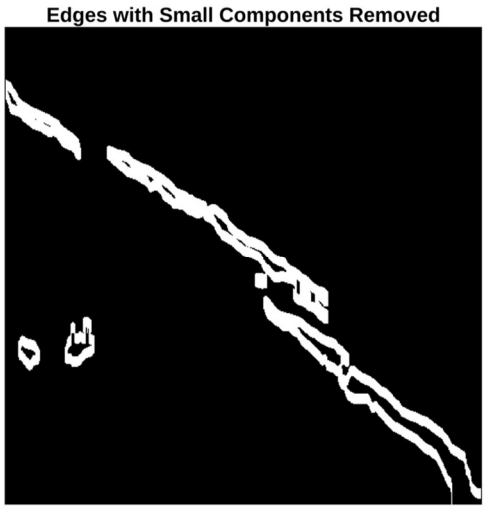
Crack detection in roads has evolved from manual inspection to automated methods using image processing and machine learning. Traditional techniques, like edge detection and thresholding, are effective but sensitive to lighting and noise. Advanced methods, such as CNNs and deep learning, offer higher accuracy but need large datasets and computational power. MATLAB's image processing tools provide a practical approach, balancing simplicity with effectiveness, ideal for educational crack detection project

4.Experimental Result

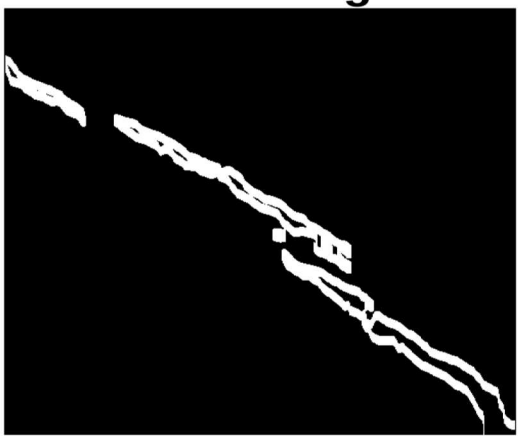
4.1 Step Wise Results:







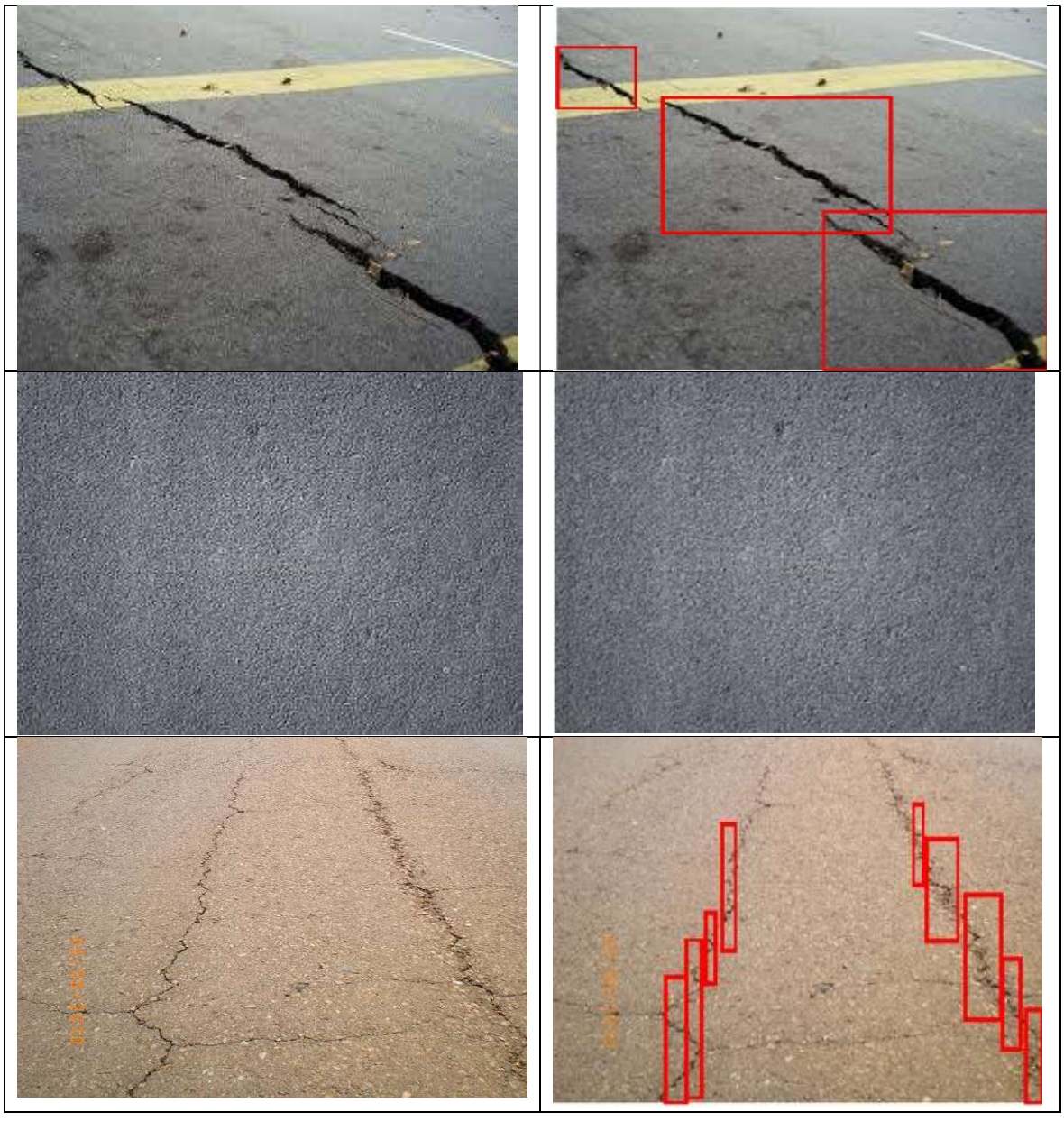
Filtered Edges



Detected Cracks with Bounding

Input Images

Output Images



5.Conclusion

This project highlights the potential of MATLAB-based image processing in effectively detecting road surface cracks. By leveraging techniques like grayscale conversion, Gaussian filtering, and edge detection, the developed system simplifies crack identification, supporting timely maintenance and improved road safety. Though effective for educational and controlled applications, future enhancements, such as incorporating machine learning, could enhance accuracy and adaptability, making it suitable for more complex, real-world conditions.

6.References

Internet sources

Textbooks based on Image processing

7.Appendix

```
% Load the image
image = imread("C:\Users\neelk\Downloads\Transverse-crack-of-road-pavement-Source-7.png");
imshow(image);
title("Original Image");
% Resize image for faster processing (optional)
image = imresize(image, [500, 500]);
% Convert the image to grayscale
grayImage = rgb2gray(image);
%imshow(grayImage);
%title("Grayscale Image");
% Apply Gaussian blur to reduce noise, with a slightly lower sigma value
blurredImage = imgaussfilt(grayImage, 1.8);
%imshow(blurredImage);
%title("Filtered Image");
% Perform adaptive thresholding to separate cracks from background
binaryImage = imbinarize(blurredImage, 'adaptive', 'ForegroundPolarity','dark', 'Sensitivity', 0.4);
imshow(binaryImage);
```

```
title("Binary Image");
% Use morphological operations to clean up the image
se = strel('line', 10,90); % Using disk structuring
element for better noise
removal
cleanImage = imopen(binaryImage, se);
%imshow(cleanImage);
%title("Cleaned Image");
% Invert the binary image so cracks become white
invertedImage = imcomplement(cleanImage);
imshow(invertedImage);
title("Inverted Image");
% Detect edges using the Canny method
edges = edge(invertedImage, 'canny');
%imshow(edges);
%title("Edges");
% Remove small components from the edges to reduce false
positives
minComponentSize = 500; % Minimum size of components to
keep
edges = imdilate(edges, se); % Dilate edges to connect
small cracks
edges = bwareaopen(edges, minComponentSize);
imshow(edges);
title("Edges with Small Components Removed");
% Further filter edges based on aspect ratio to reduce
non-crack detections
labeledImage = bwlabel(edges);
stats = regionprops(labeledImage, 'Area',
'MajorAxisLength','MinorAxisLength');
% Filter based on area and aspect ratio
for k = 1 : length(stats)
% Remove components that are too small or not elongated
enough
if stats(k).Area < minComponentSize ||
stats(k).MajorAxisLength /stats(k).MinorAxisLength < 2
edges(labeledImage == k) = 0;
end
end
imshow(edges);
title("Filtered Edges");
% Detect bounding boxes for the filtered cracks
```

```
boundingBoxes = regionprops(edges, 'BoundingBox');
% Display the detected cracks with bounding boxes on the
original image
figure, imshow(image), title('Detected Cracks with
Bounding Boxes');
hold on;
for k = 1 : length(boundingBoxes)
rectangle('Position', boundingBoxes(k).BoundingBox,
'EdgeColor', 'r','LineWidth', 2);
end
hold off;
```