# Image Analysis and Object Recognition

## Summer Semester 2023

## Project Report

## Under the supervision of

## Prof. Volker Rodehorst, Mariya Kaisheva

**Submitted by:**

**Ramakrishnareddy Parvatareddy(126028)**

# Declaration

We, hereby declare that we have worked on this Image Analysis and Object Recognition project report independently and have used only the specified sources and programs which are referred to. This project report represents our original work and does not contain any plagiarized content or unauthorized use of external materials.

We affirm that the results presented in this report are based on our diligent research, understanding, and application of the principles and techniques learned. We have thoroughly examined and verified the accuracy of the results presented.

Ramakrishnareddy Parvatareddy(126028)

# Acknowledgement

With great pleasure, we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project.

We extend our sincere and deep sense of pleasure to acknowledge of Prof. Volker Rodehorst, Christian Benz, Mariya Kaisheva for there constant support, guidance and for keen interest evinced at all stages of our project. We convey our sincere and earnest thanks for your continuous guidance and encouragement for the project.

We would like to thank Prof. Volker Rodehorst for explaining in better way in the class.

Sincerely,

Ramakrishanreddy Parvatareddy (126028)

# Abstract

This project addresses the critical task of automated crack detection and analysisthrough computer vision techniques. The process begins with data acquisition. Images of cracks are collected and annotated, forming a labeled dataset for supervised learning. The dataset is split into training and test sets, with data augmentation techniques enhancing diversity.

In the core of the project, a thresholding method adapts to varying brightness levels, producing an initial segmentation. Morphological operations refine results by enhancing connectivity and eliminating noise. Connected component analysis isolates individual crack regions, while feature engineering extracts informative attributes for classification.

Performance is evaluated using the intersection-over-union (IoU) metric, and thinning techniques represent cracks as line-like structures. A dedicated function calculates crack lengths in the test dataset.

# Contents

# Table of Contents:

# Task 1 – Data Engineering

1. Data is the fuel for nearly any kind of image-based detection challenge nowadays. If not used for training (i.e. the learning of suitable model parameters), a dataset must be present at least for evaluating the performance of a proposed approach. Thus, the acquisition and preparation of data forms a vital step in designing and developing a detection system.

a) Data acquisition: Take a walk around an urban area and look out for cracks in buildings1 Use your camera, e.g. the camera of your phone, to take pictures of cracks. Important: respect the legal restrictions when taking pictures of other peoples' property. For buildings that primarily refers to2:

- Stay on public ground (do not enter private ground without permission).
- Do not invade the owner's privacy by using aids such as drones, sticks,ladders, etc.
- Make sure to not capture artwork in a way violating copyright law.
- Preserve peoples' anonymity by not capturing other people, the doorbellnameplates, etc.

Document where you captured the image with a precise textual description. Capture atleast 10 images.

b) Data annotation: Supervised learning requires annotations (also referred to as labels or ground truth) for each input image. For semantic segmentation these labels are of same height and width as the input image. The pixel values represent the true class of the pixel. It is recommended to use values 0 for no-crack and 255 for crack. Figure 1 shows input images alongside their respective annotations. For annotation you can use basic image manipulation software such as GIMP3 or more advanced tools such as theComputer Vision Annotation Tool (CVAT)[4].

c) Data split: In order to evaluate the performance of your approach, you keep the dataset used during development separate from a test set. Thus, split your dataset into a subset for training/development (= 80%) and testing (= 20%). The test set should only be used once on the final version of your approach.

d) Data augmentation: It can optionally be helpful to use data augmentation on the acquired dataset. By applying basic transformations such as rotation, flipping, contrast shrinking, brightness shift, etc. the dataset can artificially be expanded without need of additional data collection or annotation.

e) Datasets statistics: Present basic statistics of the dataset such as the number of images, number of pixels, intensity distribution, the number of labeled pixels, etc. in tables and/or plots.

## Solution:

## Data acquisition:

Figure 1: Type 1 Crack(India) Figure 2: Type 2 Crack(friend room) Figure 3:Type 3 crack(dorm)

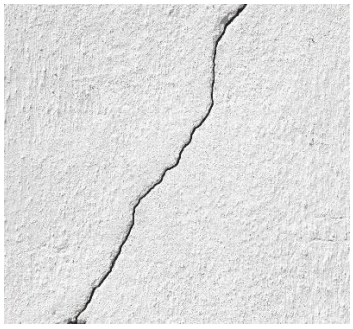

Figure 4: Type 4 Crack(Dorm) Figure 5: Type 5 Crack(India) Figure 6: Type 6 Crack(Road)

Figure 7: Type 7 Crack(India)

Figure 8: Type 8 Crack(My Home)

Figure 9: Type 9 Crack( road )

Figure 10 : Type 10 Crack(Google)

Captured a total of 10 images with cracks from different places such as university dorms and from open source(e.g: Google images).
The reason for collecting images from opensource is because the pictures collected from dorms are more or less similar cracks and couldn't be able to find deep cracks. To work on more diverse cracks we tried to collect few pictures from open source.

## Data annotation:

As recommended in the project description for semantic segmentation the labels were maintained at same height and width as per the input image. Also, considered to give 0 for no cracks region and 255 for crack region.
Refer task 1.a pictures with task 2.a pictures.

We used Computer Vision Annotation Tool (CVAT) to perform data annotation task.

In our project focused on annotating cracks in wall images, we carefully considered the choice of annotation software. Ultimately, we selected the Computer Vision Annotation Tool (CVAT) for the following reasons:

- CVAT is purpose-built for computer vision tasks, making it well-suited for our specific objective of annotating cracks. Its features are more or less similar to tasks like semantic segmentation, ensuring precise annotations.
- We found CVAT to be efficient in the annotation process. Its user-friendly interface, along with tools for polygon-based annotations and segmentation masks, allowed us to annotate images quickly and consistently.

- CVAT's support for multiple annotators working simultaneously was a valuable feature, especially for our project with a limited dataset. It facilitated teamwork and speeded up the annotation process.

- CVAT provided effective data management tools, simplifying the organization and preparation of our annotated dataset for machine learning model training.

- Importantly, CVAT is open-source and free, aligning with our project's budget constraints while delivering the required functionality.

In summary, our decision to use CVAT over GIMP was based on its alignment with our task, efficiency, support for collaboration, data management capabilities, automation potential, and cost-effectiveness. These factors collectively ensured that CVAT was the ideal choice for our project, enabling us to achieve accurate and high-quality annotations of cracks in wall images.

## Ground Truth

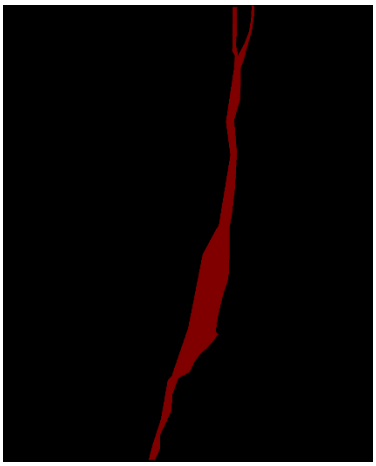

Figure 11: Type 1 Ground truth

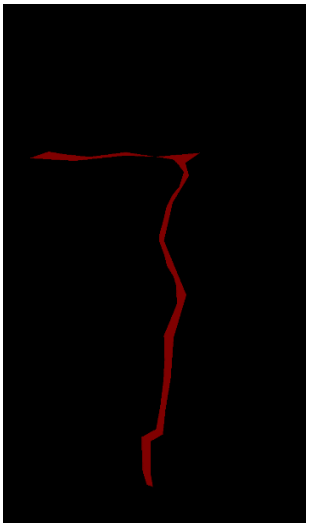

Figure 12: Type 2 Ground truth

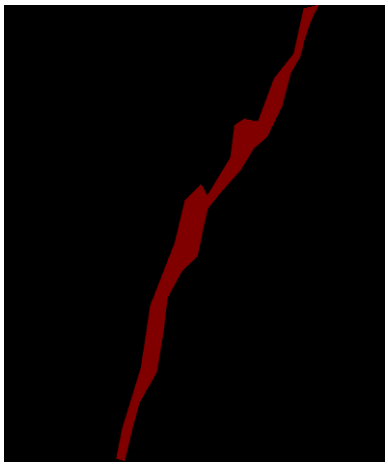

Figure 13: Type 3 Ground truth

Figure 14: Type 4 Ground truth

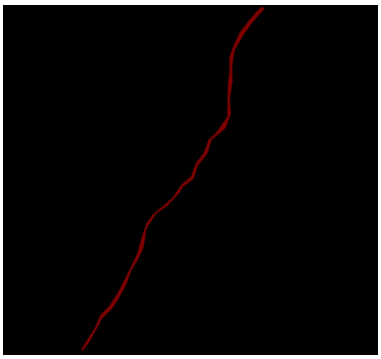

Figure 15: Type 5 Ground truth

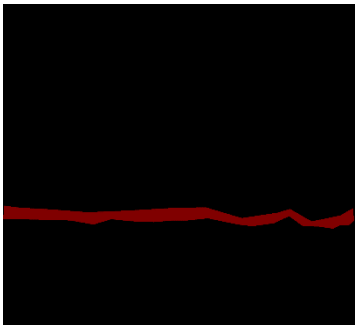

Figure 16: Type 6 Ground truth

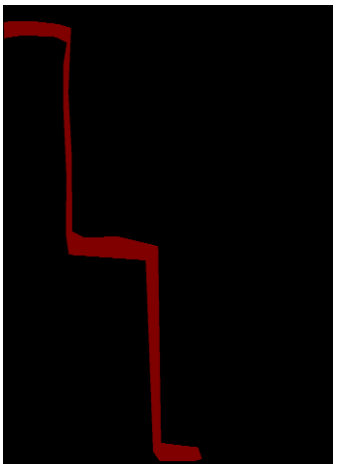

Figure 17: Type 7 Ground truth

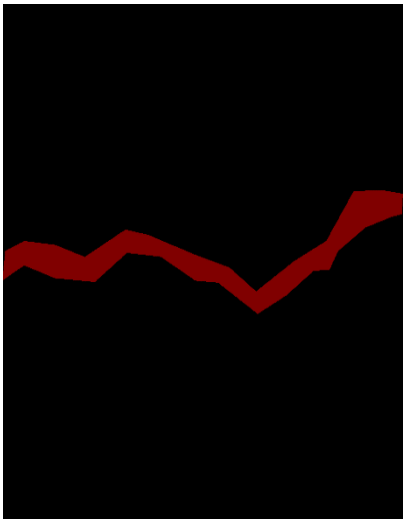

Figure 18: Type 8 Ground truth

Figure 19: Type 9 Ground truth

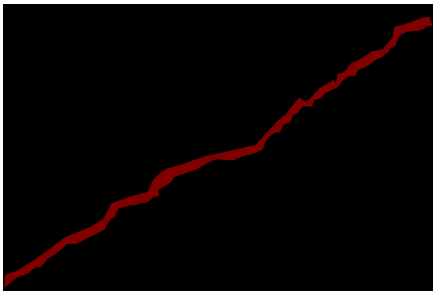

Figure 20 : Type 10 Ground truth

## Data split:

The data of 10 annotated images where splited into 4:1 ratio. Which means we divided the dataset into 80% for training/development and remaining 20% for testing as suggested in the inputs. For this Data split task we used Roboflow tool, which is also a open source and free tool.

## Data augmentation:

Also performed a data augumentation task by applying features such as rotation, flipping, contrast shrinking, brightness shift.

Flip: Horizontal, Vertical
90° Rotate: Clockwise, Counter-Clockwise, Upside Down

Figure 21: Crack1 Contrast shrink

Figure22:Crack 1 brightness change

Figure23: Crack3 Flipping

Figure24: Crack2 Contrast shrink

Figure25: Crack2 Rotation

Figure26: Crack2 Flipping

Figure27:Crack3 Rotation

Figure28:Crack3 Brightness

Figure29: Crack3 flipping

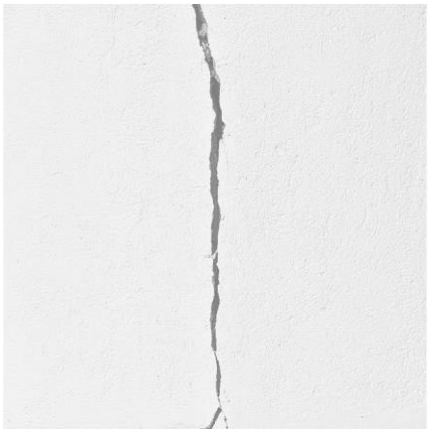

Figure 30: Crack 4 with contrast shrinking

Figure 31: Crack 4 with rotation and brightness shift

Figure 32: Crack 4 with flipping

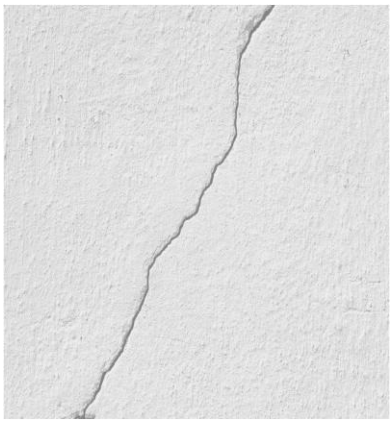

Figure 33: Crack 5 with contrast shrinking

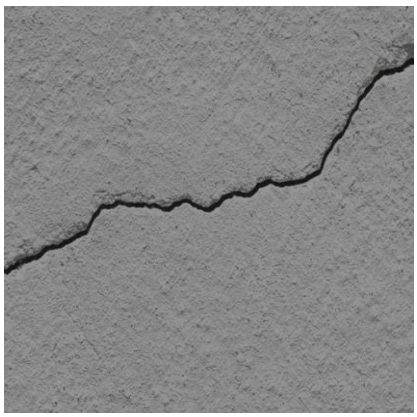

Figure 34: Crack 5 with rotation and brightness shift

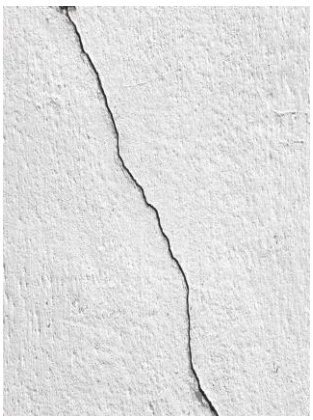

Figure 35: Crack 5 with flipping

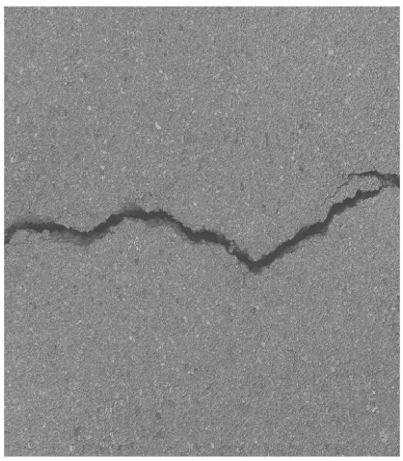

Figure 36: Crack 8 with contrast shrinking

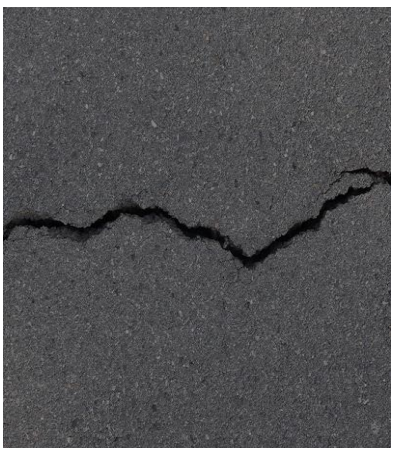

Figure 37: Crack 8 with rotation and brightness shift

Figure 38: Crack 8 with flipping

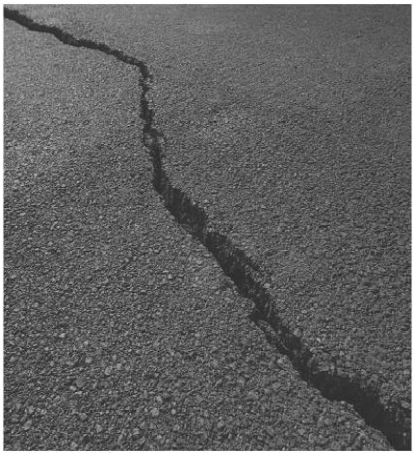

Figure 39: Crack 9 with contrast shrinking

Figure 40: Crack 9 with rotation and brightness shift

Figure 41: Crack 9 with flipping

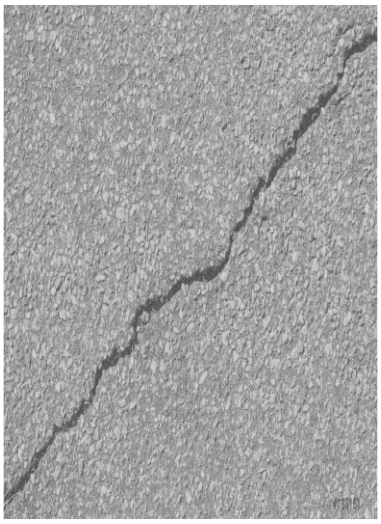

Figure 42: Crack 10 with contrast shrinking

Figure 43: Crack 10 with rotation and brightness shift

Figure 44: Crack 10 with flipping

## Datasets statistics:

| Dataset | Values |
|---|---|
| No. of Images | 10 |
| No. of pixels | 6,030,357 |
| Average pixels per image | 603035 |

Table 1: Dataset Statistics

## Intensity distribution of 10 images:

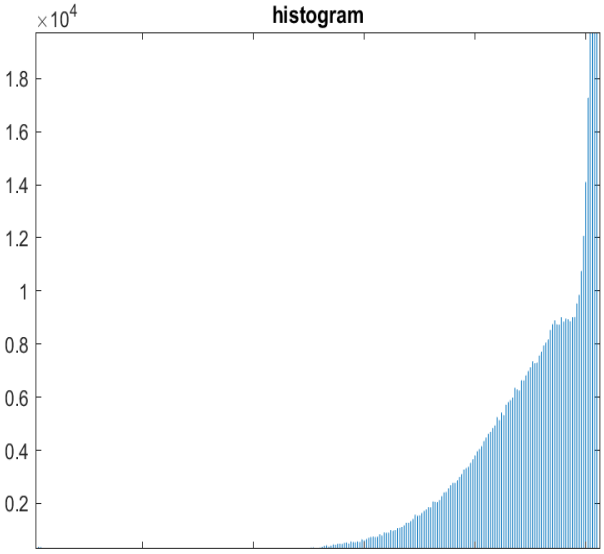


Figure 45: Histogram of Crack 1

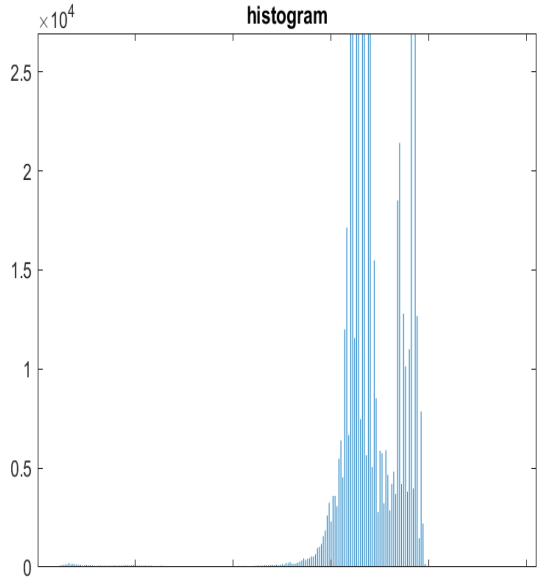


Figure 46: Histogram of Crack 2

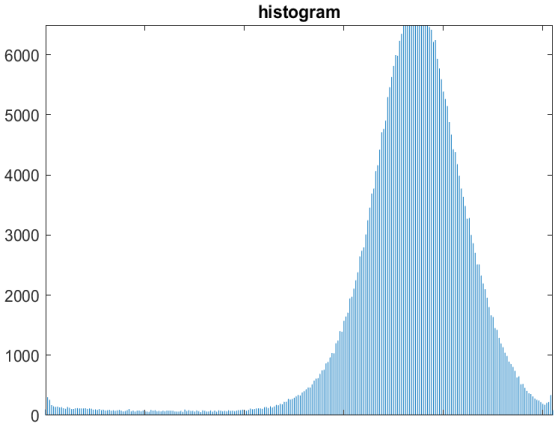


Figure 47: Histogram of Crack 3

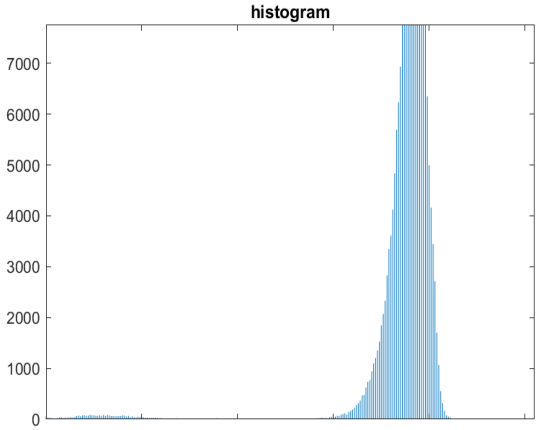


Figure 48: Histogram of Crack 4

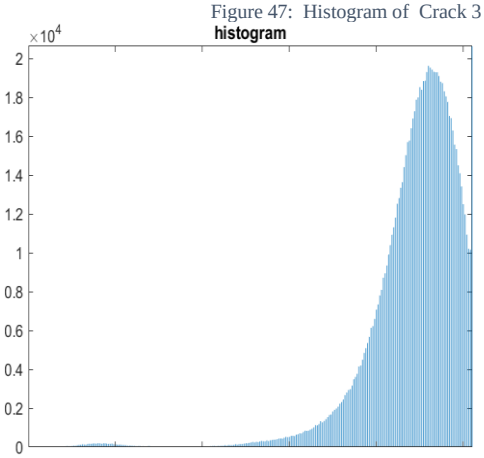


Figure49: Histogram of crack 5

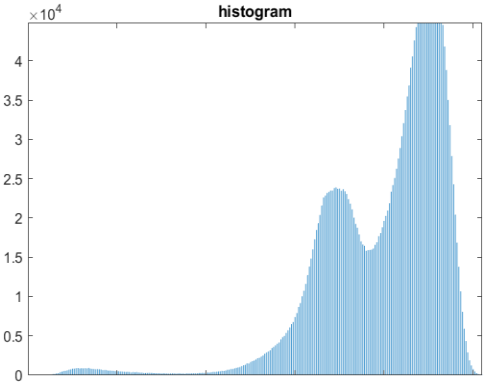


Figure50: Histogram of crack 6

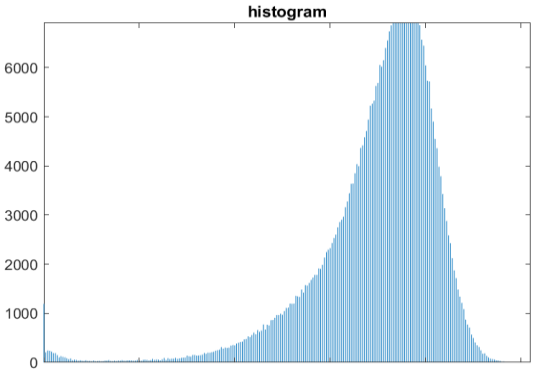


Figure51: Histogram of crack 7

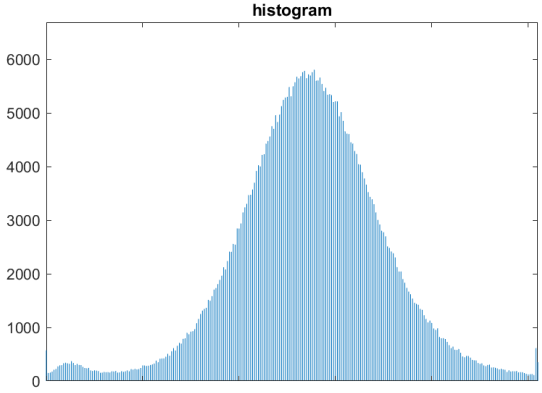


Figure52: Histogram of crack 8

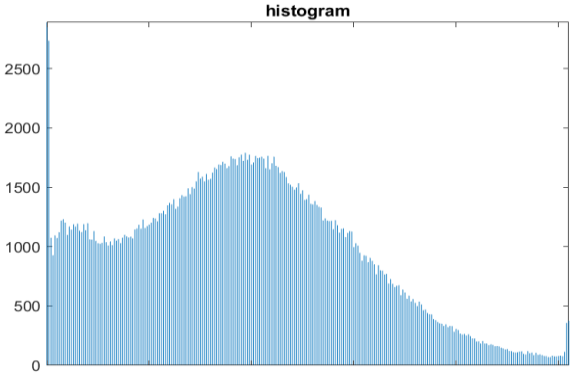


Figure53:Histogram of crack 9

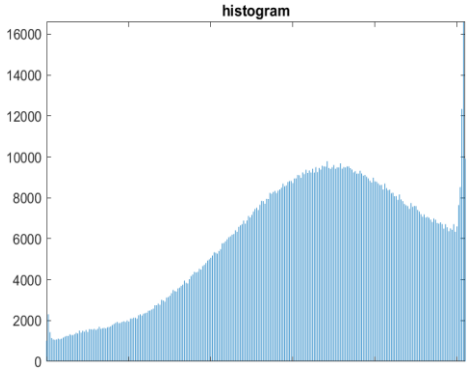


Figure54: Histogram of crack 10

## Task 2 – Crack Segmentation

In this task you propose an approach to semantically segment the cracks in the image. Semantic segmentation is the task where every pixel in the input image is assigned a class label in the output. Crack segmentation is a binary task with the classes no-crack and crack. As shown in Figure 1 it is recommended to use value 0 for no-crack and 255 for crack.

a) Cracks typically appear as dark line-like structures on brighter background. Thus, thresholding can be an intuitive first processing step for crack segmentation. Design and implement a thresholding method suitable for segmenting the cracks. Try to incorporate adaptiveness towards images of different brightness levels. The thresholding procedure usually yields many false positives, i.e. regions which do not represent cracks but are classified as crack.
b) Play around with morphological operators in order to clean up the result (e.g. by removing isolated points and ensuring a better connectedness of the cracks). You are allowed to use built-in functions.
c) From the resulting segmentation mask extract discrete regions by implementing connected component analysis as presented in the lecture.
d) Feature engineering: Create a feature vector for each instance of potentially useful features (region properties, circularity, color, ...) – be creative. Based on the annotated ground truth, assign a label to each region whether it represents a crack or no-crack. These feature vectors alongside the true labels will serve as input to the next step.
e) Classifier: Implement a classifier that uses the prepared features in order to filter out non-crack regions. You are allowed to implement a heuristic rule-based approach. Furthermore, you are welcome to experiment with support vector machines (SVM), decision trees, or any other concept taught in the course, which you consider suitable. Explain your reasons and ideas for choosing your approach.

   Heuristic rules are easy to implement and understand. They can serve as a baseline for classification. The SVM classifier is more accurate classification method.

# Task 3 – Crack Analytics

Crack segmentation unfolds its power only when the results undergo further processing. Information such as the length or the number of branches of a crack are crucial for the assessment of a structure's condition.

a) Implement a metric to assess the performance of the implemented approach. The standard metric for semantic segmentation is intersection-over-union (IoU). Report the performance of the detector on the test set and discuss the adequacy of the metric for crack detection.

   IoU is a metric for crack detection as it measures the core objective of segmenting and aligning predicted cracks with the ground truth.

b) For further processing use thinning in order to reduced the segmentation results to a line-like representation of the crack. You are allowed to use built-in functions.
c) Implement a function to compute the length of the detected cracks in the test set. Report the lengths.
d) Comment of the usefulness of the implemented approach for practical crack detection. What are the strengths and weaknesses of your approach?

**Code:**

```
clc;
close all

% Reading Input Image
input_Img=imread("image6.4.jpeg");

% CONVERTING INTO GRAY IMAGE
gray_Img = rgb2gray(input_Img);

% Visualizing Grey Image and Its Histogram
figure,imshow(gray_Img);
title('Gray Image');

figure,imhist(gray_Img);
title('histogram'),

% Applying Contrast Streaching with boundatry values of histogram ..
 enhImg = imadjust(gray_Img,[0.2,0.8],[0.00,1.00]);
% Visualizing Enhanced Image and Its Histogram
 figure,imshow(enhImg);
 title('Enhanced Image');
 % Binarization
b = gray_Img;
[x,y]=size(b);

% Using Nested for loops to seperate Foreground and Background Pixel
% Values
for i=1:x
   for j=1:y
       f=b(i,j);
       % Using 80 as threshold
      if f<=80
         b(i,j)=255;
      elseif f>80 && f<255
         b(i,j)=0;
      end
   end
end
```

```
binimg = b;
figure,imshow(binimg);
title('Binary Image');

% Assigning the variable to Binary Image Variable
% APPLYING MORPHOLOGICAL OPERATIONS
% 1.Opening

SE=strel('disk',4);
open_Img = imopen(binimg,SE);


% 2.Closing

closing_Img = imclose(open_Img,SE);
figure,imshow(closing_Img);
title('Filtered Mask')

% Now, perform connected component analysis (CCA) on the binary image
'cloimg'
cca = bwconncomp(closing_Img);  % Calculate connected components

% Get connected component properties, such as area and bounding box
areas = regionprops(cca, 'Area', 'BoundingBox');

% Extract the areas as the feature vector
featureVector = cat(1, areas.Area);

% Display the original image with bounding boxes around detected cracks
figure;
imshow(input_Img);
title('Original Image with Detected Cracks');

% Loop through each connected component and draw bounding boxes
for i = 1:length(areas)
   bb = areas(i).BoundingBox;
   rectangle('Position', bb, 'EdgeColor', 'r', 'LineWidth', 2);
end

% Displaying the number of detected cracks
```

```
Cracks = length(areas);
fprintf('Number of detected cracks: %d\n', Cracks);

% Now, you have 'featureVector' with the area of connected components as the
feature for further analysis.
```

**For contrasting the image code :**

```
clc;
close all
% Load the image
image = imread("image8.5.jpg");

% Display the original image
imshow(image);
title('Original Image');

gray_Img = rgb2gray(image);
% Convert the image to double for calculation
double_image = im2double(gray_Img);

% Calculate the mean intensity
mean_intensity = mean(double_image(:));

% Shrink the contrast using a custom formula
alpha = 0.5; % Adjust alpha to shrink the contrast (0 < alpha < 1)
contrast_shrunk_custom = mean_intensity + (double_image - mean_intensity) *
alpha;

% Display the result
figure;
imshow(contrast_shrunk_custom);
```

## Sample Output:

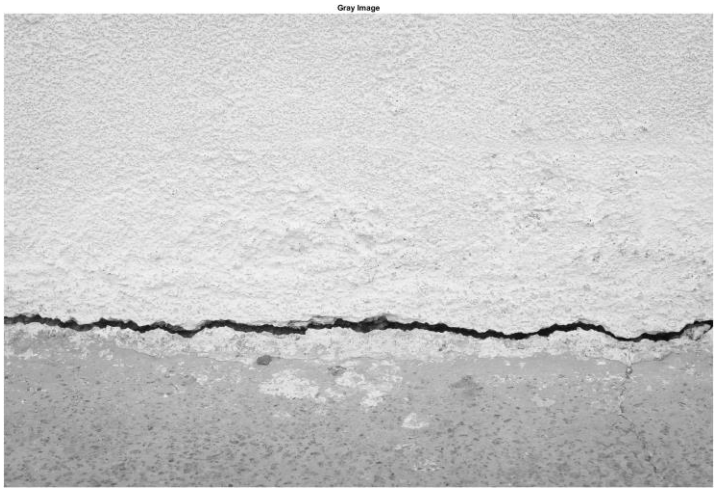

Figure 55: Gray image

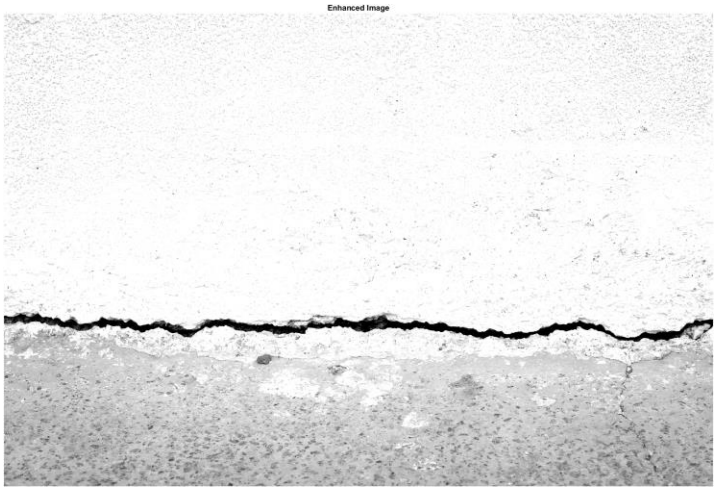

Figure 56: Enhance image

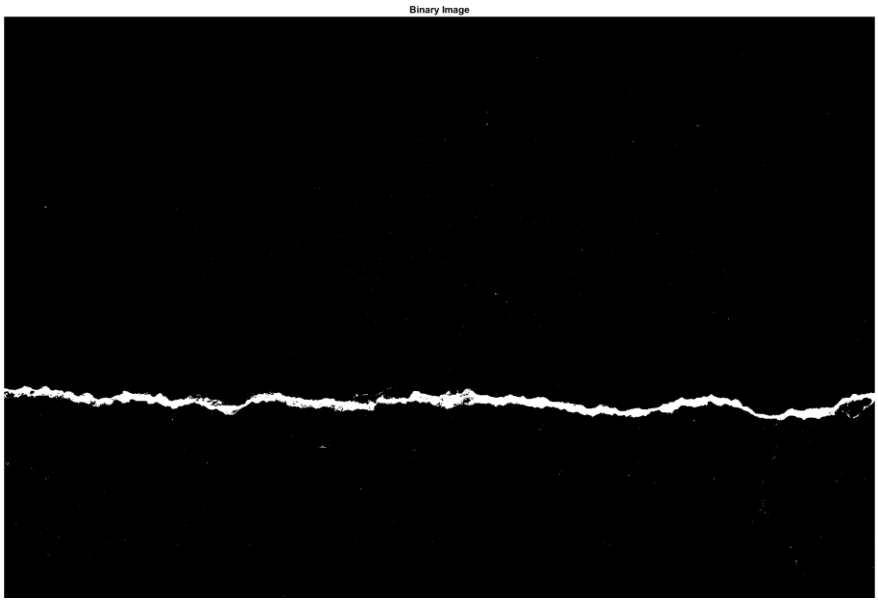


Figure 57:Binary image

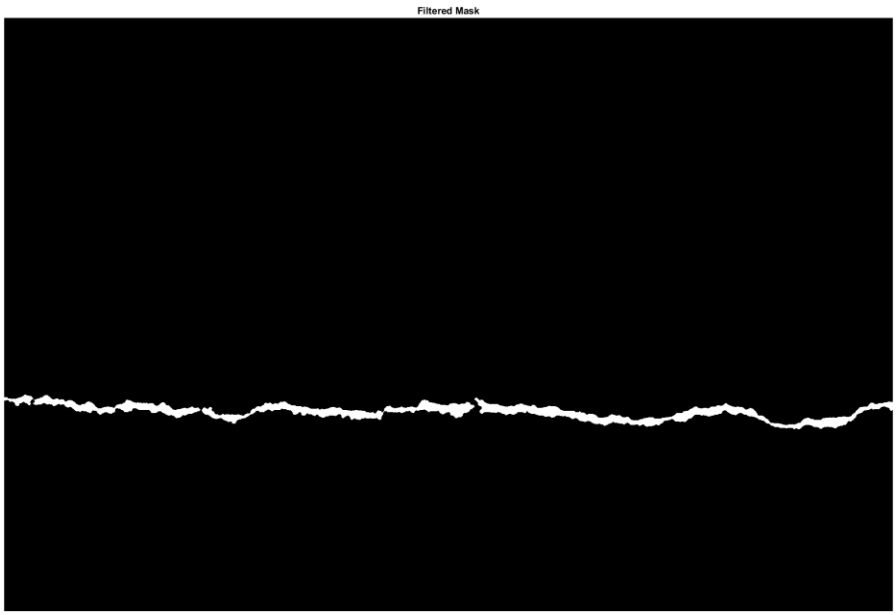


Figure 58:Filtered Mask


Figure 59: Original Image with number of cracks

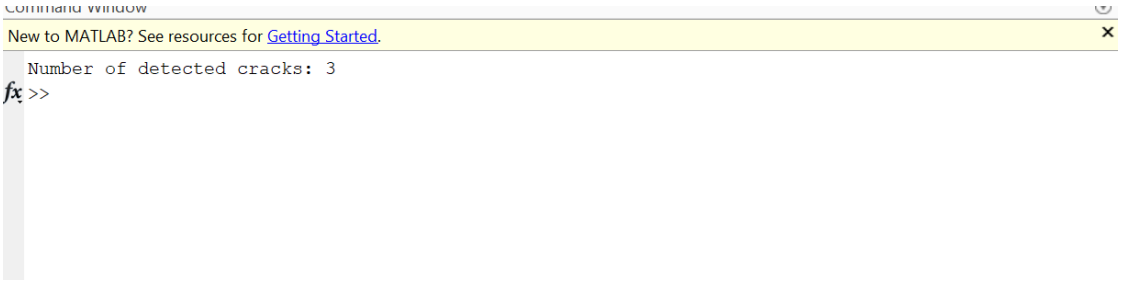


Figure 60: Output of the image

| **Image** | **Number of Cracks** |
|---|---|
| Image 1 | 12 |
| Image 2 | 3 |
| Image 3 | 4 |
| Image 4 | 7 |
| Image 5 | 11 |
| Image 6 | 3 |
| Image 7 | 17 |
| Image 8 | 4 |

| | |
|---|---|
| Image 9 | 9 |
| Image 10 | 23 |

Table 2: Output table for images

## Strengths:

Thresholding: It uses a simple thresholding technique to binarize the image. While basic, thismethod can be effective when there's a clear contrast between cracks and the background.

Morphological Operations: Morphological operations like erosion, dilation, and opening areemployed to refine the binary image and remove noise. These operations are useful for cleaning up binary images and improving the quality of detected objects.

Connected Component Analysis: The approach identifies connected components within thebinary image, which can help in segmenting individual cracks. It also attempts to merge connected components that are part of the same crack.

Feature Engineering: Region properties such as area, perimeter, eccentricity, and circularity are extracted for each connected component. These features provide valuable information fordistinguishing cracks from non-crack regions.

Crack Detection: Calculating the number of cracks using the connected component analysis.

## Weakness:

My code is matching the expectations but it some images it is not fully calculating the cracks.

Connected component analysis is giving the best answer but the some images it is providing the long cracks.

In Ground Truth we used 255 for crack but still it is visible in red color.