Project Report of CS590 Challenge 1

Table of Contents

1. Report Revision History

Name	Date	Reason for changes	Version
Xiaoping Yu	2019.06.18	Initial draft	1.0
Yunxiu Zhang	2019.07.18	More details and enhancements	1.5
Tianye Zhao, Xiaoping Yu	2019.08.05	Style, editing, additional information after research	2.0

2. Team members and Contribution

Name	Contribution Description
Xiaoping Yu	1.Research and paper reading. 2.Coding(getDownSampledGrayScale, User interface and menu) 3. Dataset , images preparation and testing 4. Write report.
Yunxiu Zhang	1.Research and technology selection. 2. Coding(getGLCM, GetEnergy, color step and integration of the 3 steps) 3. Testing, code review 4. Write report.
Tianye Zhao	1.Research and write proof of concept code. 2. Coding(getHistogram,Images read and write) 3. Testing, code review 4. Write report.

3. Objective

The aim of this challenge is to use a data driven approach to evaluate corrosion spread across a large dataset of high-quality dense image data.

4. Solution

The solution we use is based on the Corrosion Detection for Automated Visual Inspection, Weak-classifier Color-based Corrosion Detector (WCCD). The solution consists of 3 major steps: preparation step, roughness step and color step [1].

4.1 Preparation step

This is the preparation step for the third step. We cut some of the known corroded area from some sample images and save them in a folder to serve as image data set. Then we load them to construct a HS based 2-dimensional histogram. After that, we get the max value from histogram and update all values less than 10%*max to 0.

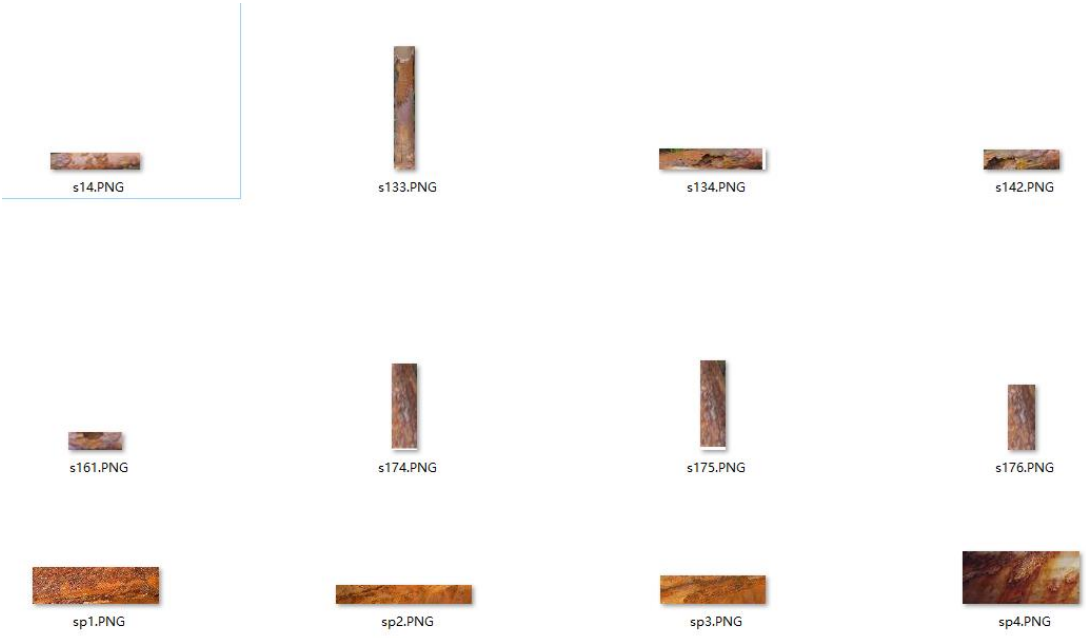
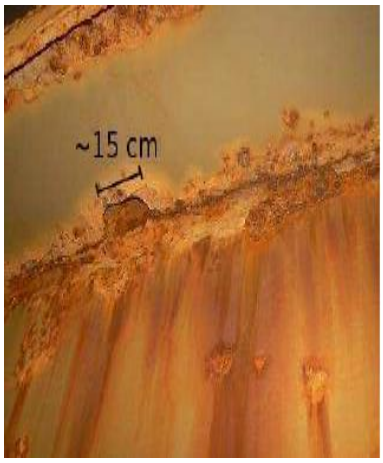
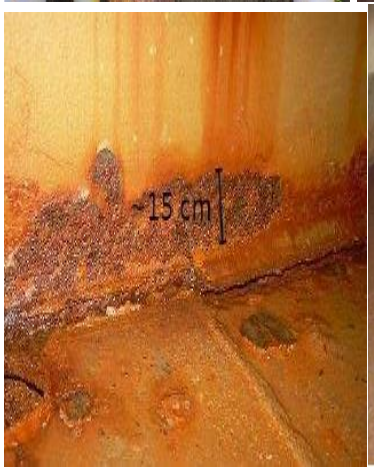


Figure 1. Dataset (Files provided separately)


~10 cm


~15 cm


~15 cm


~20 cm

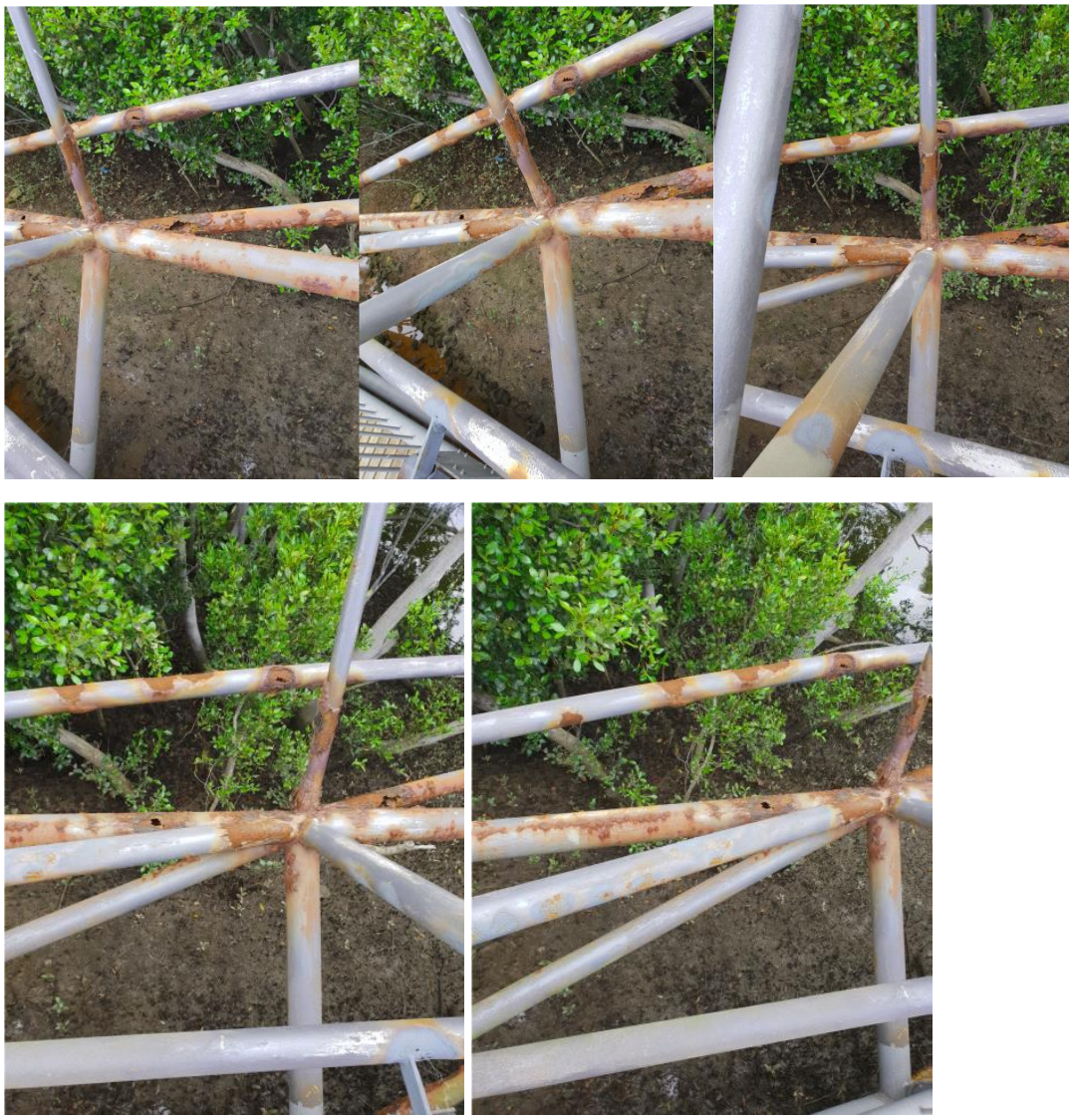


Figure 2. Images to detect (Files provided separately)

4.2 Roughness step

The basic idea is that a corroded area presents a rough texture and the roughness can be evaluated with the energy of the symmetric gray-level co-occurrence-matrix (GLCM).

To calculate the energy for all 15*15 patches of each image, we read the image and get its gray scale presentation array, and then calculate the GLCM and energy for each 15*15 patch. If an patch has energy less than a threshold (0.05), which means it's rough, we pass it to next color step for further analysis.

4.3 Color Step

In this step, we read the image and get their HSV presentation array.

Then, we ignore (label as non-corroded) the pixels close to black or white, which are very unlikely to be in corroded area.

After that, we check the HS of the pixel to see if its histogram value is 0. If not 0, we label it as corroded with different colors based on its histogram value:

red if HS(h, s) 2 [0.75HS, 1.00HS],

orange if HS(h, s) 2 [0.50HS, 0.75HS],

green if HS(h, s) 2 [0.25HS, 0.50HS] and

blue if HS(h, s) 2 [0.10HS, 0.25HS],

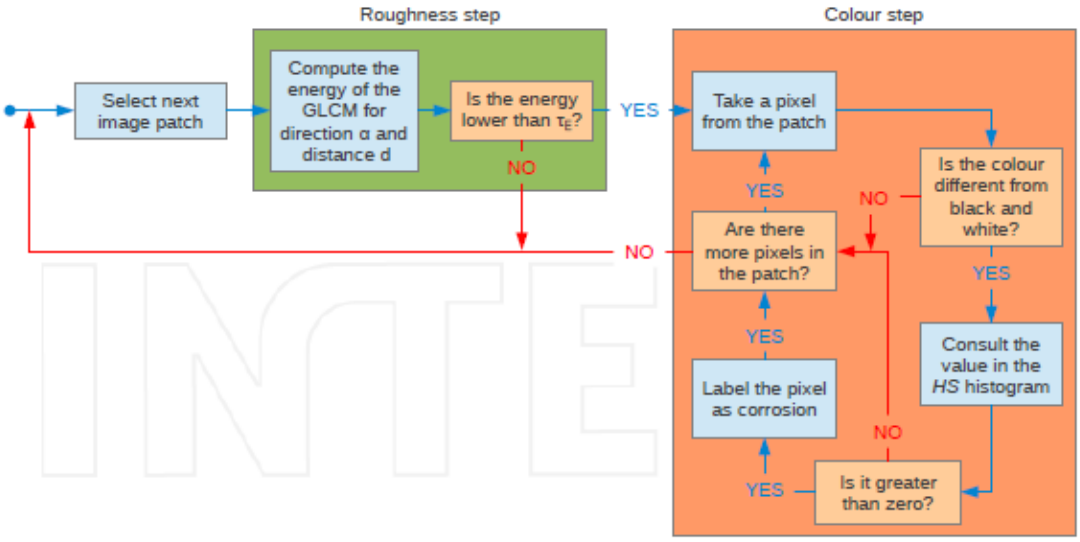


Figure 3. Algorithm flowchart [1]

5. Technical implementation

Our solution is implemented in Java (JDK8) without external framework, library or tool. Source code will be provided separately.

Steps to run the program

(1) Put all dataset files in one folder

(2) Put all images-to-be-detected in another folder

(3) Install JDK 8 if not yet

(4) Run CorrosionDetector.java as Java application in IDE(e.g. Eclipse) Or Run *mvn exec:java* on DOS command line under project folder.

(5) Do as per the menu suggests(e.g. Provide the location of the 2 folders)

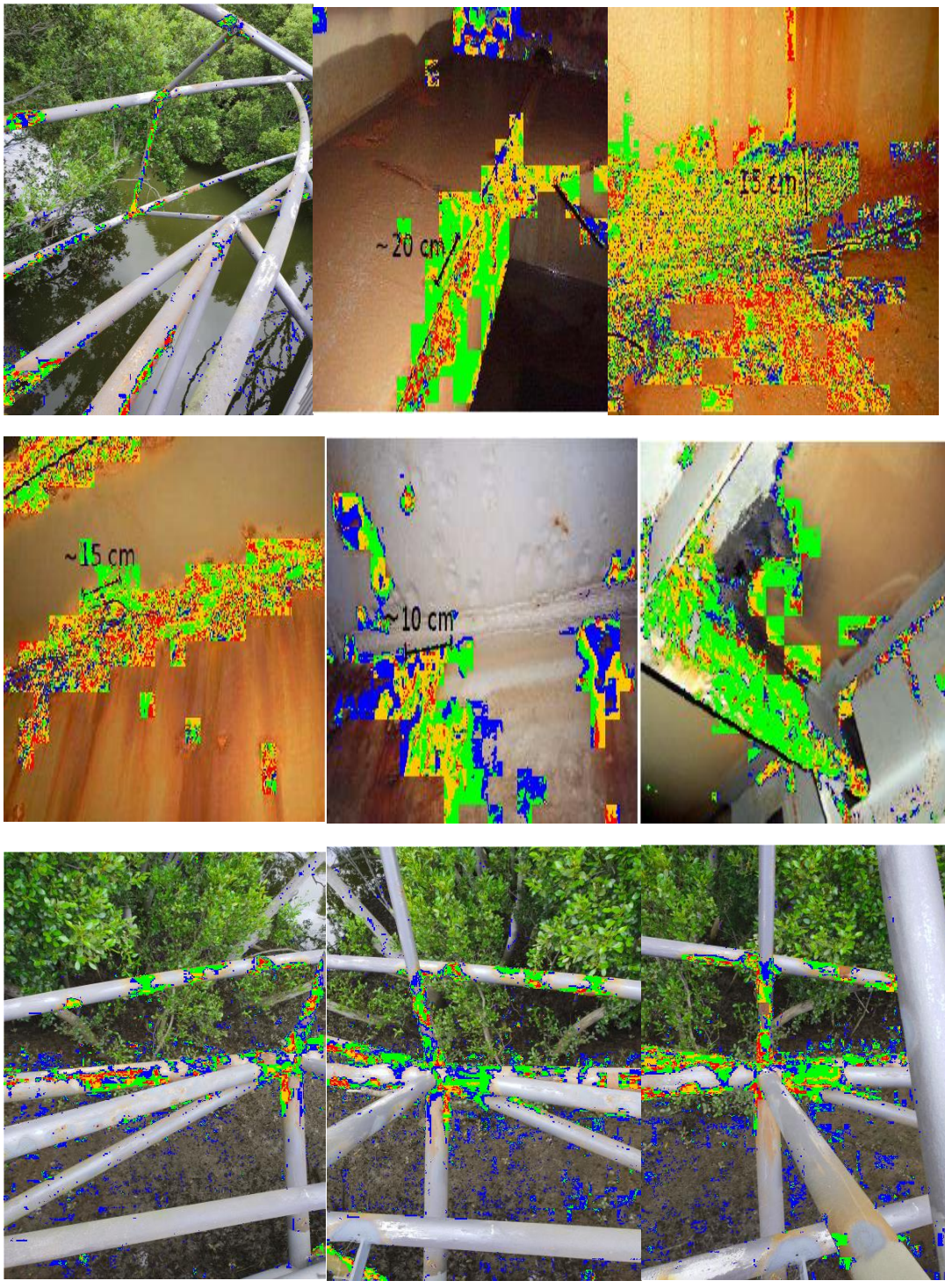
(6) Check out the result images on image folder(result images all have a name which ends with _result).

6. Results

The program is a standalone Java application with simple command line user interface and menus.

```
**********Corrosion Detection Application**********
******By Yunxiu Zhang, Xiaoping Yu, Tianye Zhao*********

Good Evening! Welcome to Corrosion Detection Application.

This application is the solution of CS590 Challenge 1

Please choose below menu to continue...
********** Menu **********
*****1. Inspect pictures*****
*****0. Exit*****

Please input a number to choose a menu

1
Please input the full location of the image data set for training

C:\BS\SummerProject\set
Please input the full location of the pictures for corrosion detection

C:\BS\SummerProject\pics
Success! Labeled images are stored at C:\BS\SummerProject\pics

Please choose below menu to continue...
********** Menu **********
*****1. Inspect pictures*****
*****0. Exit*****

Please input a number to choose a menu
```

Figure 4. User Interface


~20 cm
~15 cm
~15 cm
~10 cm

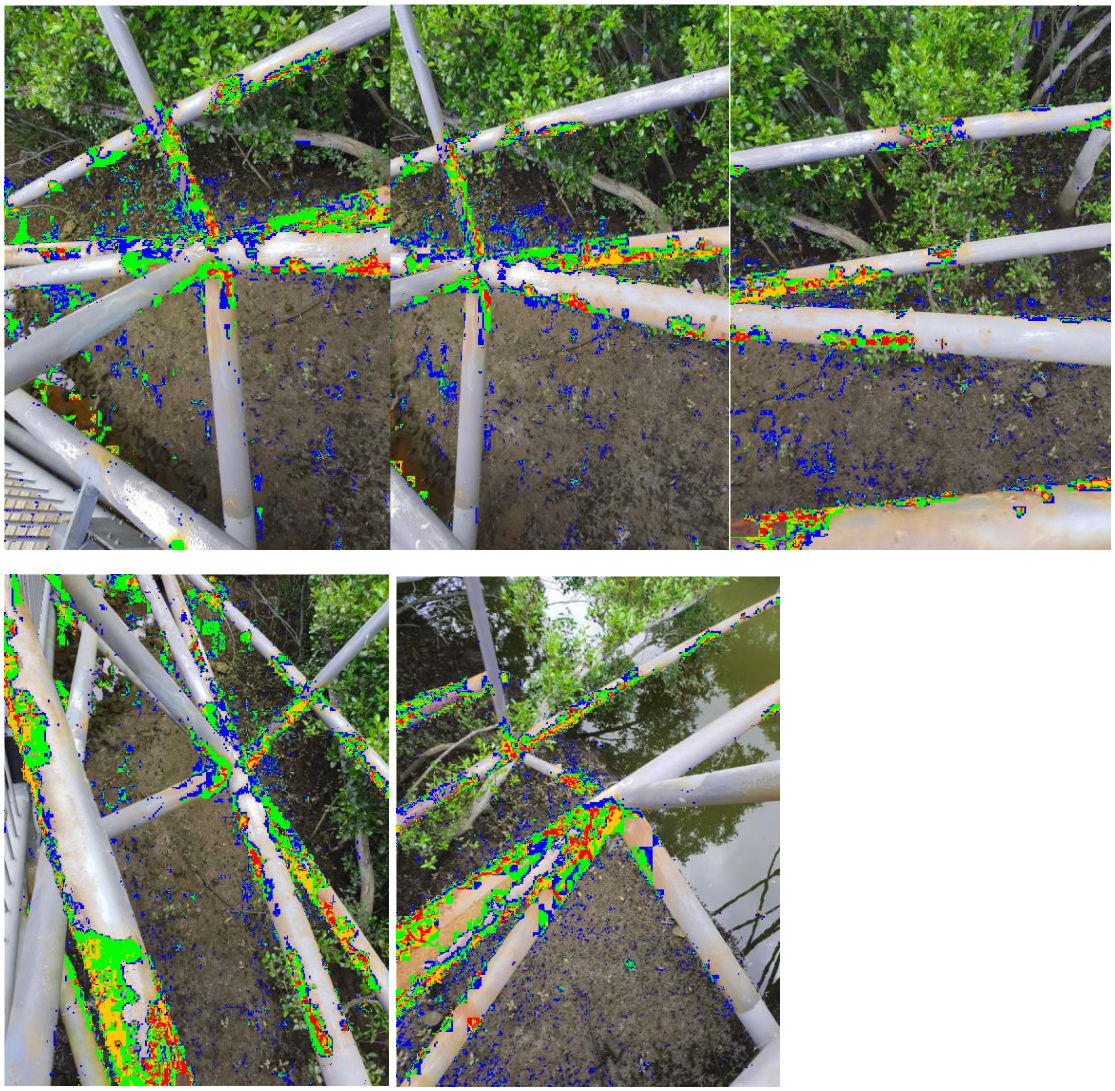


Figure 5. Corrosion Detection Result (Files provided separately)

Meaning of colors in Corrosion Detection Result.

Color	**Meaning**
Red	Very severely corroded
Orange	Severely corroded
Green	Corroded
Blue	A little corroded or something close to corrosion

7. Conclusion

To tackle this challenge, we have designed and developed an interactive Java standalone application which gets information from prepared dataset and then successfully detects corrosion on multiple images in a folder and generate the result images under the same folder.

We have presented this implementation to Professor Mohammed Ayoub Alaoui Mhamdi in person and he has validated our results.

References

[1] Francisco Bonnin-Pascual , Alberto Ortiz. Corrosion Detection for Automated Visual Inspection