Identification of corodded areas in an image

ARNAOUT, BASSAM
Dept. of Computer Science
Bishop's University
Sherbrooke, Canada
barnaout18@ubishops.ca

***Abstract*— Most of industries nowadays make use of iron materials in manufacturing their products. However, corrosion is a natural process which affects badly on the quality of the materials & hence could lead to a disastrous result. For this, to avoid unwanted consequences, it is very important to be able to detect corroded areas at their earlier stage. There are many techniques available that serve such purpose. In this paper, we will discuss one technique based on texture features extraction on an image using GLCM (gray level co-occurrence matrix) and applying WCCD (Weak-classifier Color-based Corrosion Detector) algorithm on it.**

Keywords—rust, corrosion detection technique, glcm

I. INTRODUCTION

Iron machines and materials are utilized in most of industries for producing product. In industries these iron materials are exposed to humidity and pollution, this will increase the oxidization of iron. Corrosion takes place once the mechanical materials are exposed to the humidness and pollution in industries. Because of the attack of the corrosion, those materials become weak and this will affect the integrity of the metallic surface. The rust accumulated by corrosion will lead to reduction in iron material efficiency and will increase the cost to maintain it. [1]. The following sections describe texture features derived form GLCM, introduce WCCD algorithm and discuss the steps implemented to detect corroded areas in an image. Then will show the performance against a varied set of test images.

II. MEASURING TEXTURE OF AN IMAGE [2]

A. Image Texture Characterization

Various tries to characterize image textures over the years are primarily based on extracting the primary and second order properties of the grey levels. By 1st order, we mean the properties that may be measured from the individual pixels themselves. The first-order properties are generally depend on the pixels' means, the variances, etc.. The second-order properties involve comparison of two pixels at same time. The second-order properties, therefore, investigate how one pixel at some reference location relates statistically to a different pixel at a location displaced from the reference location. In what follows, GLCM is an example of texture characterizations that depends on its second-order statistical properties.

Any numerical characterization of image textures should possess some essential properties so as to be helpful in sensible applications:

–To the most extent potential, it should be invariant to changes in image contrast which will be created by uneven illumination of a scene.

–To the most extent potential, it should be invariant to in-plane rotations of the image.

–It should lend itself to quick computation.

So it is best if the tactic used for characterizing a texture is generally freelance of the macro-level variations within the contrast in every image. That is, we need ways that extract texture connected info from the changes within the grey levels at the pixels level and their immediate neighborhoods.

B. Image Texture Characterization with grey-level co-occurrence matrix (GLCM)

The basic idea of GLCM is to estimate the joint probability distribution P[x1, x2] for the gray levels in an image, where x1 is the gray level at any randomly selected pixel in the image and x2 the gray level at another pixel that is at a specific vector distance d from the first pixel. After we have estimated P[x1, x2], the texture can be characterized by the shape of this joint distribution.

Therefore, thinking about the gray levels at two different pixels that are separated by a fixed displacement vector d is a good place to start for understanding the GLCM approach to texture characterization.

Let's say we raster scan an image left to right and top to bottom and we examine the gray level at each pixel that we encounter and at another pixel that at a displacement d with respect to the first pixel. As an image is being scanned, the (m, n)-th element of the GLCM matrix records the number of times we have seen the following event: the gray level at the current pixel is m while the gray level at the d-displaced pixel is n.

To illustrate with a toy example, consider the following 4 × 4 image with pixels whose gray levels come from the set {0, 1, 2}:

```
2 0 1 1
0 1 2 0
1 1 1 2
0 0 1 1
```

And let us assume a displacement vector of d = (1, 1).

As we scan the image row by row by visiting each pixel from top left to bottom right, if m is the gray level at the current pixel and n the gray level at the pixel one position to the right and one row below, we increment glcm[m][n] by 1. Scanning the 4 × 4 array shown above, we get the following 3 × 3 GLCM matrix:

```
      ----------> displaced pixel
     |              gray levels
     |
     |    0 1 1                          NOTE:  The glcm matrix
     |    2 3 0                                 is of size 3x3
     |    0 1 1                                 because we have
     |                                          ONLY 3 gray levels
     V                                          which are {0,1,2)
reference pixel
  gray levels
```

As we would expect, what we get is an asymmetric matrix as shown above. The matrix is asymmetric because, in general, the number of times the gray level at the reference pixel is m while the gray level at the displaced pixel is n will not be the same for the opposite order of the gray levels at the two pixels.

Nonetheless, as we will see later, in general one is interested primarily in the fact that two gray levels, m and n, are associated together because they occur at the two ends of a displacement vector, and one does not want to be concerned with the order of appearance of these two gray levels. When that is the case, it makes sense to create a symmetric GLCM matrix. This can easily be done by the simple expedient of incrementing the element glcm(n,m) when we increment glcm(m, n). For the above example, this yields the result:

```
0 3 1
3 6 1
1 1 2
```

Here are some interesting observations about GLCM matrices: If we sum the diagonal elements of a normalized GLCM matrix, we get the probability that two pixels in the image that are separate by the displacement vector d will have identical gray levels— assuming that the GLCM matrix was constructed for a given displacement d. Along the same lines, if we sum any non-diagonal entries in the normalized GLCM atrix that are on a line parallel to the diagonal, we get the probability of finding the gray level difference at any two pixels separated by d corresponding to the line on which the GLCM elements lie.

C. Summary of GLCM Properties

Here is a summary of the GLCM matrix properties:

• GLCM is of size M ×M for an image that has M different gray levels.

• The matrix is symmetric

• Typically, one does NOT construct a GLCM matrix for the full range of gray levels in an image. For 8-bit gray level images with its 256 shades of gray (that is, the shade of brightness at each pixel is an integer between 0 and 255, both ends inclusive), we are likely to create a GLCM matrix of size of just 16 × 16 that corresponds to a re-quantization of the gray levels to just 4 bits (for the purpose of texture characterization).

• Also typically, one may construct multiple GLCM matrices for the same image for different values of the displacement vectors. At the least, one uses the displacement vectors: (0, 1), (1, 0), and (1, 1).

To interpret a GLCM matrix as a joint probability distribution, we need to normalize the matrix by dividing its individual elements by the sum of all the elements. That gives we a legitimate joint (or bivariate) probability distribution over two random variables that represent the gray levels at the two ends of the displacement vector.

D. Deriving Texture Measures from GLCM

Each element of a GLCM matrix provides too "microscopic" a view of a texture in an image. What we need is a larger "macroscopic" characterization of the texture from the information contained in a GLCM matrix.

Here are some textures attributes that are derived from a GLCM Matrix:

$$Energy = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} P[i,j]^2$$

$$Entropy = -\sum_{i=0}^{M-1}\sum_{j=0}^{M-1} P[i,j]\log_2 P[i,j]$$

$$Contrast = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} (i-j)^2.P[i,j]$$

$$Homogeneity = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1} P[i,j]}{1+|i-j|}$$

where P[i, j] is the normalized GLCM matrix. As mentioned earlier, P[i, j] is the joint probability distribution of the gray levels at the two ends of the displacement vector in the image. We will talk about the first texture attribute "Energy" as we will use this only in our algorithm to detect corroded areas in an image. Energy, is also called "Uniformity." Its value is the smallest when all the P[i, j] values are the same — that is, for the case of a completely random texture. Note that since P[i, j] must obey the unit summation constraint, if the values are high for some values of i and j, they must be low elsewhere. In the extreme case, only one cell will have the value P[i, j] equal

to 1 and all other cells will be zero. For this extreme case, Energy acquires the largest value of 1. At the other extreme, each cell will have a value of 1/256 for the case a 16x16 GLCM matrix. In this case, Energy will equal 256 × (1/256)2 = 1/256. For all other cases of 16 × 16 GLCM matrices, Energy will range from the low of 1/256 to the max of 1.

III. Weak-classifier Colour-based Corrosion Detector (WCCD)

WCCD is a supervised classifier which has been built around a cascade scheme, although its two stages can be considered as weak classifiers. The idea is to chain different fast classifiers with poor performance in order to obtain a global classifier attaining a much better global performance [3].

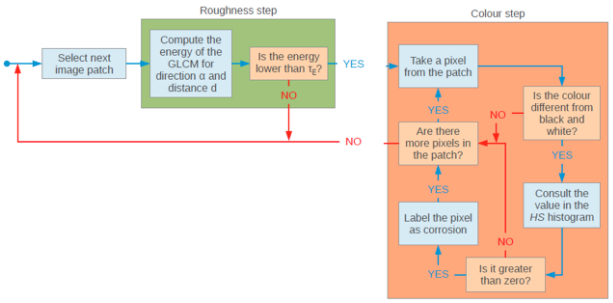


Figure 1. WCCD algorithm flowchart

A. Roughness stage

The first stage of the cascade is based on the premise that a corroded area presents a rough texture. Roughness is then related to the energy of the symmetric gray-level co-occurrence-matrix (GLCM), calculated for **downsampled intensity values between 0 and 31**, for a given direction a and distance d [3]. The energy is calculated from the formula mentioned in section II, D. Here it is again:

$$Energy = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} P[i,j]^2$$

where p(i, j) is the probability of the occurrence of gray levels i and j at distance d and orientations a or a + p. Patches with an energy lower than a given threshold tE, i.e. exhibit a rough texture, are finally candidates to be more deeply inspected [3].

To configure the parameters of the roughness stage, several experiments have been performed considering different values for d and a when calculating the GLCM and, consequently, its energy level. No significant differences have been observed among the output values, and so the parameter values are set to **d = (2,2) (pixels) and a = 0** (horizontal direction). As for the energy threshold tE, its value determines the algorithm performance in terms of computation time and number of false positives, since all patches with a high energy level are discarded and only those with a low value become input of the colour checking step [3].

The first stage operates over **15 × 15 -pixel image patches** since it depends on texture, which necessarily involves a pixel neighborhood.

Figure 2 provides classification outputs for the same input image using different energy thresholds tE. This parameter can be tuned to decrease false positives and just allow the detection of the most significant corroded areas. The best value detected for the energy threshold is found to be **tE = 0.05**

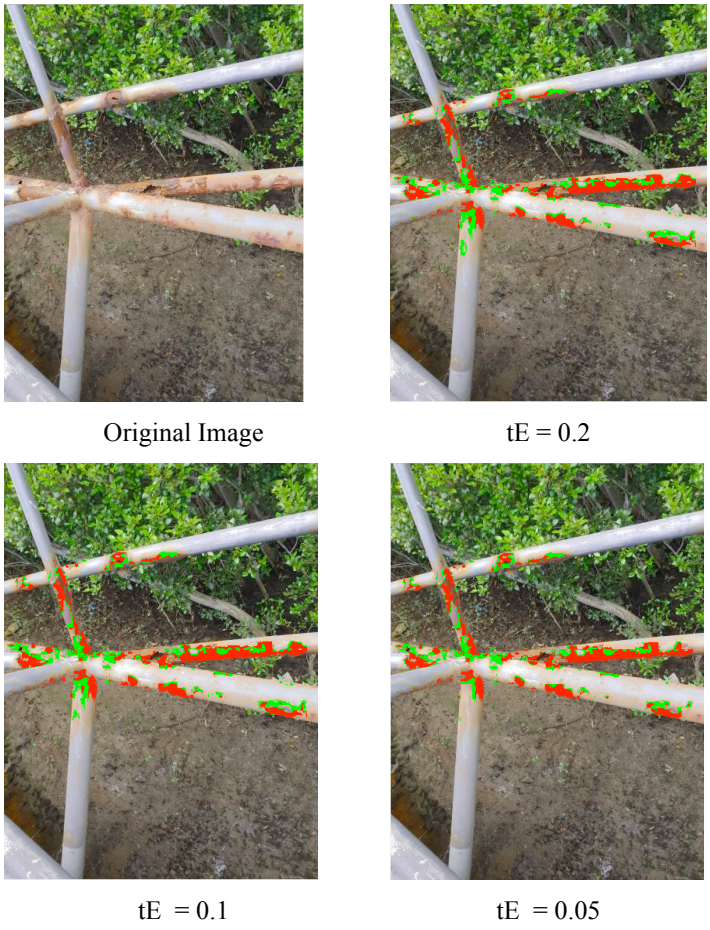


Figure 2. Corroded areas detected by WCCD for different energy threshold values

B. Color Stage

The second stage filters the pixels of the patches that have passed the roughness stage. This stage makes use of the color information that can be observed from corroded areas. More precisely, the classifier works over the Hue-Saturation-Value (HSV) space after the realization that HSV-values that can be observed in corroded areas are confined in a bounded subspace of the HS plane. Although the V component has been observed neither significant nor necessary to describe the color of corrosion, it is used to prevent the well-known instabilities in the computation of hue and saturation when color is close to white or black. In that case, the pixel is classified as non-corroded [3].

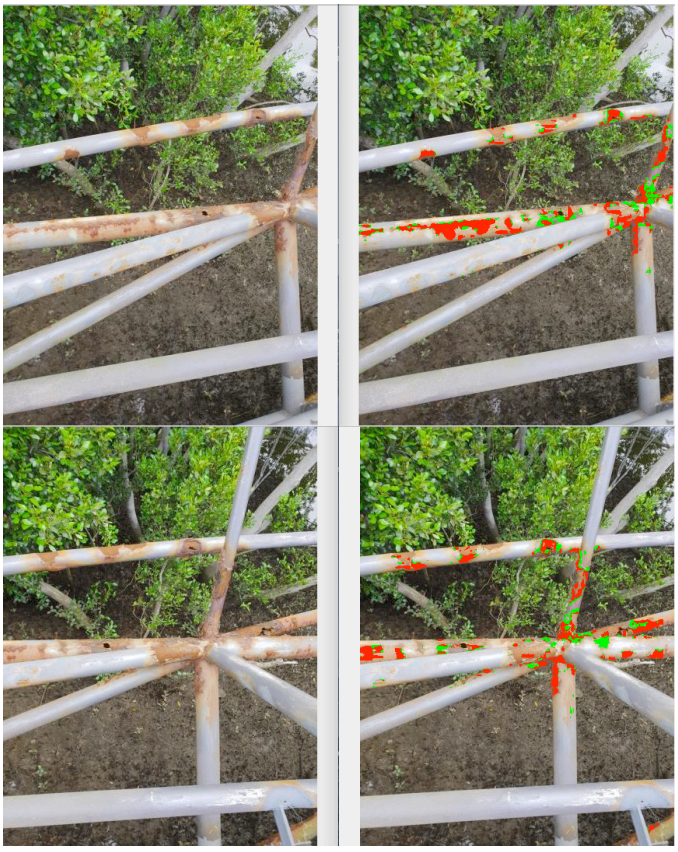
The classifier proceeds as follows for every 3-tuple (h, s, v):
1) pixels close to black, v < 0.5, or white, v > 0.5 ^ s< 0.2 and h <=0.07, are labeled as non-corroded,
and
2) for the remaining pixels, the HS histogram is consulted and the pixel is labelled as corroded if HS(h, s) > 0

In the images, pixels labelled as corroded are color-coded to indicate the probability of successful classification. To be more precise, the color depends on the height of the corresponding histogram bin in the following way:

• red if HS(h, s) ∈ [0.3HS, 1.00HS],
• green if HS(h, s) ∈ [0.15HS, 0.3HS] and
• blue if HS(h, s) ∈ [0.0HS, 0.15HS],
where HS = max{HS(·, ·)}.

IV. Testing the algorith WCCD

Examples of final classification outputs on sample images for WCCD are provided in Figure 3.



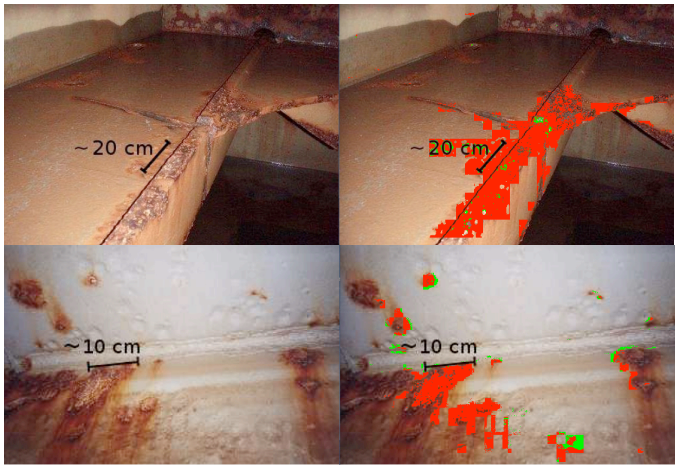


Figure 3. (1st column) Test images and (2nd column) corroded areas detected by WCCD

REFERENCES

[1] Mariana P. Bento, Fatima N. S. De Medeiros, Ialis C. De Paula Jr. and Geraldo L.B. Ramalho, "Image Processing Techniques applied for Corrosion Damage Analysis".

[2] https://engineering.purdue.edu/kak/Tutorials/TextureAndColor.pdf

[3] https://www.intechopen.com/download/pdf/46235.