Combination of Weak Classifiers for Metallic Corrosion Detection and Guided Crack Location*

Francisco Bonnin-Pascual and Alberto Ortiz
Dep. of Mathematics and Computer Science
University of Balearic Islands, Spain
{xisco.bonnin, alberto.ortiz}@uib.es

Abstract

Periodic visual inspection of the internal and external parts of vessels hull is typically performed by trained surveyors at great cost. Assisting them during the inspection process by means of mechanisms capable of defect detection would certainly decrease inspection cost. With this aim, this paper presents a corrosion detection algorithm built around a weak classifiers cascade scheme and reports on its performance. As a secondary contribution, a crack detection approach guided by the output of the corrosion detector is also proposed. As a result, false positives rate is reduced as well as the computation time.

1. Introduction

Humans and goods transportation by mean of ships and vessels is nowadays one of the most time and cost effective methods. In order to improve the transportation service quality, Classification Societies require achieving an always increasing safety standard level against maritime accidents. Structural failure, its major cause, is in many cases preventable with timely maintenance. To this end, vessels are periodically submitted to extensive inspection schemes.

Taking into account the huge dimensions of some vessels, this process can mean the visual assessment of more than 600,000 m^2 of steel. Besides, the surveys are on many occasions performed in hazardous environments for which the access is usually difficult and the operational conditions turn out to be sometimes extreme for human operation. Moreover, total expenses involved by the infrastructure needed for close-up inspection of the hull can reach up to one million euros for certain sorts of vessel (e.g. Ultra Large Crude Carriers, ULCC). Therefore, it is clear that any level of automation of the inspection process that can lead to a reduction of the inspection time, a reduction of the financial costs involved and/or an increase in the safety of the operation is fully justified.

With this aim, this paper presents a method for detecting corrosion in images as a support for surveyors during vessel inspection. The method adopts an approach based on a weak classifiers cascade scheme for discriminating pixels suspected to be affected by corrosion from those not affected. To the best of authors' knowledge, many strategies have been proposed about general defects detection, but only the work described by Xu and Weng in [4] refers specifically to detecting corrosion in metallic surfaces. In their paper, the authors adopt an approach based on the fractal properties of corroded surfaces.

A secondary contribution of this paper is a method combining the previous corrosion classifier and a crack detector to improve the performance of the latter, both in terms of processing time and classification success. The idea underlying this approach is that cracks in metallic surfaces typically present corrosion as well. Consequently, the computational effort can focus on areas suspected to be corroded.

The paper is organized as follows: Section 2 describes the corrosion detection algorithm, Section 3 presents some results obtained during performance testing using vessel surface images, Sections 4 and 5 describe and compare the two crack localization strategies —single and combined with the corrosion detector— and, finally, Section 6 concludes the paper.

2. Corrosion detection method

This section presents a corrosion detection algorithm that has been built around a weak classifier cascade scheme. The idea is to chain different fast classifiers with poor performance in order to obtain a global classifier attaining a much higher global performance. To this end, each weak classifier takes profit from different features of the items to classify, reducing the number of false positive detections at each stage. For a good global performance, the classifiers must present a false negative rate close to zero.

The cascade of the algorithm presented in this paper consists of two stages. The first one is based on the

*This work is partially supported by FP7 project SCP8-GA-2009-233715 (MINOAS). The authors are thankful to Lloyd's Register of Shipping (London, UK), and to RINA s.p.a. (Genova, Italy) for providing the test images used in this study.

premise that a corroded area presents a rough texture. Roughness is then measured as the energy of the symmetric *gray-level co-occurrence matrix* (GLCM), calculated for downsampled intensity values between 0 and 31, for a given direction α and distance d [3]:

$$E = \sum_{i=0}^{31} \sum_{j=0}^{31} p(i,j)^2 \,, \tag{1}$$

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

The second stage of the cascade operates over the pixels of the patches that have passed the roughness stage. Unlike the first, this stage makes use of the colour 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.

A training step is performed prior to the application of this second stage of the corrosion classifier. Training consists of building a bi-dimensional histogram of HS values downsampled between 0 and 31 for image pixels known to be affected by corrosion in the training image set. The resulting histogram is subsequently filtered by zeroing entries whose value is below 10% the highest peak. By way of example, Figure 1 shows the HS histogram calculated from our test images set.

Next, the classifier proceeds as follows for every 3-tuple (h,s,v):

(1) pixels close to black, $v < mV$, or white, $v > MV \wedge s < mS$, are labeled as non-corroded, and

(2) for the remaining pixels, the HS histogram is consulted and the pixel is labelled as corroded if $HS(h,s) > 0$,

for given thresholds mV, MV and mS.

Notice that the stages of the cascade here presented cannot be swapped since they do not work with the same kind of entities: while the second stage works at the pixel level, the first stage operates over 15×15 -pixel image patches since it depends on texture, which necessarily involves a pixel neighborhood.

3. Corrosion detection results

The performance of the corrosion detection algorithm has been tested using a collection of vessel images that present corroded areas.

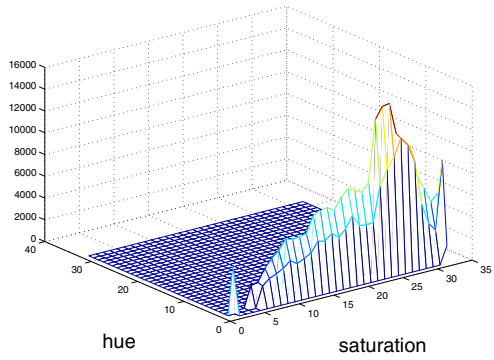
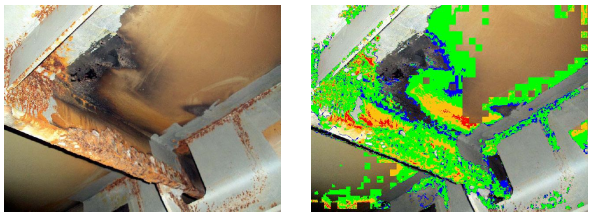


Figure 1. Hue-Saturation histogram for the corrosion detector



(a) Original image (b) $\tau_E = 0.2$

(c) $\tau_E = 0.1$ (d) $\tau_E = 0.05$

Figure 2. Classification output for different energy threshold values

First, several executions of the algorithm have been performed in order to determine a suitable value for the energy threshold τ_E. This threshold determines the patches (and their pixels) that are allowed to pass to the second stage. Similarly, several experiments have been performed using different values for d and α to choose the most appropriate distance and orientation. However, no significant differences have been observed in the obtained energy values, what suggests isotropy in the corrosion texture. Consequently, d and α have been set to, respectively, 5 pixels and $0°$, i.e. the horizontal direction.

Figure 2 provide the classification output for the same input image and different energy thresholds. As can be observed, τ_E can be tuned to decrease false positives and just allow the detection of the most significant corroded areas.

Classification output after the application of the second stage is provided in Figure 3. In the images, a pixel labeled as corroded is color-coded to indicate the prob-

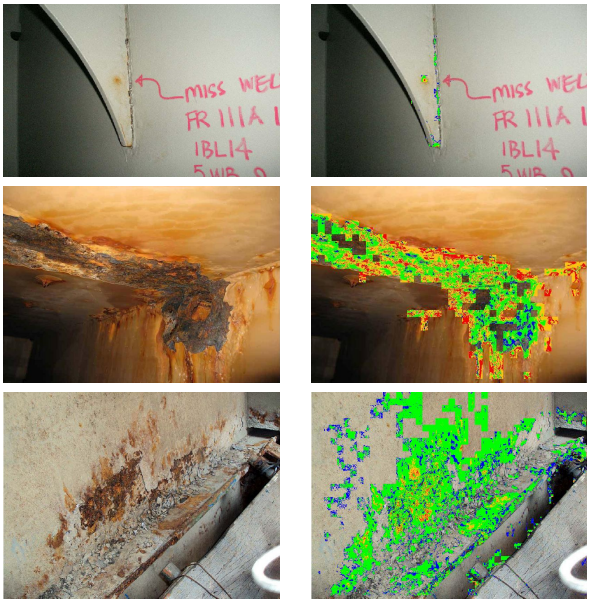


Figure 3. Test images and detected corrosion

ability of successful classification. To be more precise, the color depends on the height of the corresponding histogram bin in the following way:

- red if $HS(h,s) \in [0.75\mathbb{HS}, 1.00\mathbb{HS}]$,
- orange if $HS(h,s) \in [0.50\mathbb{HS}, 0.75\mathbb{HS}]$,
- green if $HS(h,s) \in [0.25\mathbb{HS}, 0.50\mathbb{HS}]$ and
- blue if $HS(h,s) \in [0.10\mathbb{HS}, 0.25\mathbb{HS}]$

where $\mathbb{HS}$ is the highest peak in the HS histogram.

The parameters mV, MV and mS have been set to prevent black and white pixels from being labeled as corrosion. Using 8-bit HSV values, the *minimum value* mV has been set to 50, as well as the *minimum saturation* mS. The *maximum value* MV has been set to 200.

Among the different classification outputs provided in Figure 3, special mention is done for the image in the first row, a specific kind of corrosion called *pitting*, affecting a very reduced fraction of the image, is successfully detected.

To finish, regarding execution times, the detector took between 10 and 40 ms for images ranging from 120.000 to 515.000 pixels along the different experiments performed. This is because the computation time does not depend as much on the size of the image but on the occurrence of the property checked by the first stage of the cascade, i.e. roughness: the more probable it is, the larger is the amount of attention that the pixels enclosed in the corresponding patch deserve.

4. Crack detection method

This section presents a crack detector based on a percolation model, as well as the algorithm by Yamaguchi and Hashimoto described in [5]. This latter method was, however, devised for detecting cracks in concrete, what makes the authors assume a geometrical structure that does not match exactly the shape of cracks that are formed in steel.

The percolation process consists in a region-growing procedure which starts from a seed element and propagates in accordance with a set of rules. In our case, the rules are defined to identify dark, narrow and elongated sets of connected pixels, which are then labelled as cracks. In order to optimize the detector, seed points are defined only at edges that have not yet been classified as crack pixels and whose gray level is below γ_s.

Once a seed has been located, the percolation process starts as a two-stage procedure: during the first stage, the percolation is applied inside a window of $N \times N$ pixels until the window boundary is reached; in the second step, if the elongation of the grown region is above ϵ_N, a second percolation is performed until either the boundary of a window of $M \times M$ pixels ($M > N$) is reached or the propagation cannot proceed because the gray level of all the pixels next to the current boundary are above a threshold T (see below). Finally, all the pixels within the grown region are considered as candidates to be crack pixels if the elongation is larger than ϵ_M. Elongation is calculated by means of Equation 2:

$$\epsilon = \sqrt{1 - \frac{\mu_{xx} + \mu_{yy} - \sqrt{4\mu_{xy}^2 + (\mu_{xx}^2 - \mu_{yy}^2)}}{\mu_{xx} + \mu_{yy} + \sqrt{4\mu_{xy}^2 + (\mu_{xx}^2 - \mu_{yy}^2)}}}, \quad (2)$$

where μ_{xx}, μ_{yy} and μ_{xy} are the normalized second central moments of the region [1].

Within the $N \times N$ or $M \times M$ pixel window, the percolation proceeds in accordance to the next propagation rules:

(1) all the 8-neighbours of the percolated area are defined as candidates and

(2) each candidate p is visited and included in the percolated area only if its gray level value $I(p)$ is lower than the threshold T, which has been initialized to the seed pixel gray level value.

Finally, at the end of the percolation process, the average gray level of the set of pixels is checked to determine if it is dark enough for the region to be considered as a crack, i.e. it is below a given threshold γ_{avg}. Otherwise, the set of pixels is rejected and nothing is marked as a crack in the current percolation.

Regarding the differences with [5], in Yamaguchi's crack detector:

(1) all pixels are candidate to be seed pixels,

(2) only the seed pixel is labeled as crack once the percolation finishes,

(3) it uses an acceleration parameter that allows the percolation of lighter areas and

(4) it is not required that the average gray level of the percolated region is below a certain threshold.

As a result of these differences, our algorithm is faster for crack detection in metal surfaces. For reducing even more the execution time, not all the edges are considered for starting the percolation, but only those at image places over a regular grid where the gap between points is q pixels. To ensure that the relevant edges are always considered, a dilation step follows the edge detection. Dilation thickness is in accordance with q.

5. Guided crack detection

In order to improve the performance of the crack detector of Section 4, this method has been combined with the corrosion detection algorithm presented in Section 2. The rationale behind this approach lies on the observation that most of the cracks in metallic surfaces appear in corroded areas. Therefore, a successful corrosion detection can be used to guide the localization of cracks in the image.

To implement the guided crack detection, the initial condition that starts a percolation has been slightly modified so that it starts at a given seed pixel only if it has been labeled as corroded.

Figure 4 provides the classification output for the unguided and guided versions of the crack detector. As can be observed, the computation time falls below 50% for the guided crack detector. Moreover, it is also worth observing that the time reported for guided detection does include the execution of the corrosion detector. The improvement in terms of time is thus even higher.

A second and very important enhancement that is obtained by means of the guidance is the reduction of the false positive detection rate, and consequently the improvement of the classifier reliability. As can be seen in Figure 4, both methods detect all the cracks from the images but the unguided version also marks other elongated, narrow and dark zones, e.g. shadows. The guided version, however, prevents de percolation of some of these false positives since corrosion has not been detected there.

6. Conclusions

A corrosion detection algorithm for support in vessel hull inspection has been presented in this paper. The method is based on a cascade of weak classifiers and its performance has been proved using test images whose corroded areas have been detected in close-to-real-time.

As a secondary contribution, the corrosion detector has been used to improve the execution of a crack locator also described in this paper. The results obtained prove that the guided approach saves more than 50% of the computation time with regard to the unguided version, while the number of false positive detections is also reduced, and the classifier reliability increased.

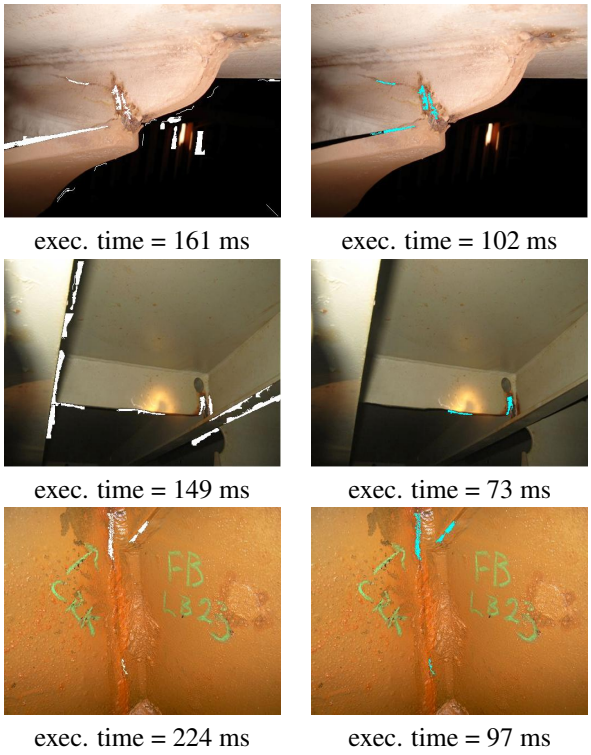


Figure 4. Detected cracks for (left) original and (right) guided detectors.

It is also important to mention that both algorithms are expected to show a better performance if the illumination during the imaging process was controlled.

As for future work, the corrosion detector is planned to be enhanced by means of an *Adaptative Boosting* (AdaBoost) [3] scheme to chain different weak classifiers in a more grounded way. Before each stage, this method computes a weighting distribution to give emphasis to the incorrectly classified samples of previous stages.

Another research line to be investigated is around the concept of *saliency* [2]. Considering corrosion and cracks as anomalies over metallic surfaces, saliency maps turn out to be relevant tools to improve their detection.

References

[1] B. Horn. *Robot Vision*. MIT Press, 1986.

[2] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1254–1259, 1998.

[3] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, 3rd Edition*. Academic Press, 2006.

[4] S. Xu and Y. Weng. A new approach to estimate fractal dimensions of corrosion images. *Pattern Recogn. Lett.*, 27(16):1942–1947, 2006.

[5] T. Yamaguchi and S. Hashimoto. Fast crack detection method for large-size concrete surface images using percolation-based image processing (in press). *Machine Vision and Applications*, 2010.