Detection of Cracks and Corrosion for Automated Vessels Visual Inspection

Francisco Bonnin Pascual

xisco.bonnin@uib.es

Abstract

Despite the efforts on reducing maritime accidents, they still occur and, from time to time, have catastrophic consequences both in personal, environmental and financial terms. Structural failure is the major cause of ships wreckages and, as such, Vessel Classification Societies impose extensive inspection schemes for assessing the structural integrity of vessels.

The external and internal parts of the hull can be affected by different kinds of defects typical of steel surfaces and structures, such as cracks and corrosion. Nowadays, to detect these defects, visual hull inspections are carried out at a great cost. The goal of the EU-funded FP7 MINOAS project is to develop a fleet of robots for automating as much as possible the aforementioned inspection and maintenance operations.

Within this general context, the work presented constitutes a first attempt towards the remote visual inspection and documentation of hull surfaces. In this regard, the two main defective situations, cracks and corrosion, are expected to be autonomously or semi-autonomously detected by means of computer vision techniques.

In this work, several algorithms are presented for visual detection of the above mentioned two kinds of defects. On the one hand, a crack detector is described, which is based on a percolation process that exploits the morphological properties of cracks in steel surfaces: dark, narrow and elongated sets of connected pixels.

On the other hand, two different approaches for corrosion detection are introduced and compared. While the first one takes profit from the distribution of color in corroded areas, the second one has been built around a weak classifier cascade scheme, separating the spatial and colour analysis in two different steps. As an final contribution, the crack detector is combined with the corrosion detector in order to guide the crack location and improve its performance. The obtained detectors have shown promising rates of detection as well as close to real-time performance.

Key words: Vessel inspection, Crack detection, Percolation, Corrosion detection, Classification

Contents

1. Introduction

Vessels constitute one of the most cost effective forms of transporting bulk goods around the world. However, despite the efforts on reducing maritime accidents, they still occur and, from time to time, have catastrophic consequences both in personal, environmental and financial terms. Structural failure is the major cause of ships wreckages and, as such, Classification Societies (also known as Shipping Registers, Flag States, etc.) impose extensive inspection schemes for assessing the structural integrity of vessels.

An important part of the vessel maintenance has to do with the visual inspection of the external and internal parts of the vessel hull. They can be affected by different kinds of defects typical of steel surfaces and structures, such as cracks and corrosion. These two kinds of defects are indicators of the state of the metallic surface and, as such, an early detection prevents the structure from buckling and/or fracturing.

Nowadays, to perform a complete hull inspection, the vessel has to be emptied and situated in a dockyard, where typically temporary staging, lifts, movable platforms, etc. need to be installed to allow the workers for close-up inspection (and repair if needed) of all the different metallic surfaces and structures. 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, while the operational conditions turn out to be sometimes extreme for human operation. For large tonnage vessels, such as Ultra Large Crude Carriers (ULCC), total expenses can be as high as one million euros.

Corrosion and the presence of cracks are clear indicators of the state of the hull metallic structures, and, thus, are of great interest for the surveyor (see Figure 1). On the one hand, cracks generally develop at intersections of structural items or discontinuities due to stress concentration, although they also may be related to material or welding defects. If the crack remains undetected and unrepaired, it can grow to a size where it can cause sudden fracture. Therefore, care is needed to visually discover fissure occurrences in areas prone to high stress concentration.

On the other hand, different kinds of corrosion may arise: *general corrosion*, that appears as non-protective friable rust which can occur uniformly on uncoated surfaces; *pitting*, a localized process that is normally initiated due to local breakdown of coating and that derives, through corrosive attack, in deep and relatively small diameter pits that can in turn lead to hull penetration in isolated random places; *grooving*, again a localized process, but this time characterized by linear shaped corrosion which occurs at structural intersections where water collects and flows; and *weld metal corrosion*, which affects the weld deposits, mostly due to galvanic action with the base metal, and are likelier in manual welds than in machine welds.

To determine the state of a corroded structure it is common to estimate the corrosion level as percentage of affected area. Traditional methods quantify corrosion by visual comparison of the area under study with dot patterned charts (see Figure 1[bottom right]).

The goal of the EU-funded FP7 MINOAS project is to develop a fleet of robots for automating as much as possi-

Figure 1: Defective situations to be detected.

ble the aforementioned inspection (and maintenance) operations. Within this general context, the work that is presented in this paper constitutes a first attempt towards the remote visual inspection and documentation of hull surfaces. In this regard, the two main defective situations, cracks and corrosion, are expected to be autonomously or semi-autonomously detected by means of the algorithms presented in this paper.

The following sections describe several methods and asses their performance. Tests have been performed over a laptop with an Intel Core2 Duo processor running at 2.20GHz, with 4 GB of RAM and executing Windows Vista. It is also important to notice that execution times do not comprise image preprocessing, thus they just refer to the execution of the proposed algorithms.

The paper is organized as follows: Section 2 reviews the state of the art related to hull inspection and computer vision approaches for defect inspection, in Section 3 a crack detection method is described and its performance is assessed, in Sections 4 and 5 two different corrosion detection algorithms, and the results obtained, are discussed and compared, Section 6 presents an improved crack detection method which combines a crack and a corrosion detectors. Finally, Section 7 concludes the paper.

2. State of the art

A number of proposals of automated or semi-automated hull inspection can be found in the scientific literature, both commercial and non-commercial. However, most of them refer to inspection of the external part of the hull by means of Remotely Operated Vehicles [1, 9, 13, 32, 41], while just a few proposals use Unmanned Vehicles, e.g. see [14]. The main goal of those systems is to assist with the detection of the loss of the external coating due to corrosion, detection of life beings attached to the hull which in turn contribute to accelerating corrosion, detection of artificial objects attached to the hull (to avoid sabotages), weld inspection, etc. Most part of those systems do not use visual sensors to perform the inspection and if they use them, the output of the system is a mosaic to be surveyed by a human expert, so that no on-line defect inspection is performed.

Talking about defect detection in general, the vision literature contains a number of proposals. The starting point for the search that has been performed is the Keith Price annotated computer vision bibliography list [1]. This list comprises main conferences and journals on pattern recognition and image processing: Proceedings of the IAPR International Conference on Pattern Recognition (ICPR), Proceedings of the IEEE International Conference on Image Processing (ICIP), Proceedings of the IAPR Workshop on Machine Vision Applications (WMVA), Springer Journal of Machine Vision and Applications (MVA), Springer Journal of Real Time Imaging (RTI), IEEE Transactions on Systems, Man and Cybernetics (TSMC), Elsevier Journal of Image and Vision Computing (IVC), etc. The search performed does not intend to be complete, but it pretends to provide a representative view of the different approaches and techniques applied in automatic defect inspection through image processing.

The different proposals reviewed can be classified depending on the defect that they try to detect. Some methods are dedicated to general defect inspection, although an important amount is devoted to crack detection in varied materials.

A different classification can be performed depending

[1] http://iris.usc.edu/Vision-Notes/bibliography/contents.html

on the technique applied for the detection. Some methods make use of general image processing techniques selected and combined to detect a specific kind of defect in a specific material, while others rely on a previous learning stage using techniques such powerful as Neural Networks, Support Vector Machines, Genetic Algorithms, etc. Table 1 shows the proposals reviewed classified according to this last criterion.

Some of these methods have been used as starting point to develop the solutions that are presented, described and assessed in the next sections.

3. Percolation-based Crack Detector

3.1. Description of the algorithm

This section presents a crack detector based on a percolation model, similarly to the algorithm by Yamaguchi and Hashimoto described in [47]. 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. Besides our method, which will be referred as PCD from now on, has been speeded up so that the algorithm can run as close as possible to real-time onboard a robotic platform.

For a start, a percolation model derives into 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.

Once a seed has been located, the percolation process starts as a two-step procedure: during the first step, 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 classified as crack pixels if the elongation is larger than ϵ_M. Elongation is calculated by means of Equation 1:

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

where μ_{xx}, μ_{yy} and μ_{xy} are the normalized second central moments of the region [18].

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.

At the end of the percolation process, the mean gray scale level of the set of pixels is checked to determine if it is dark enough to be considered as a crack. Otherwise, the set of pixels is rejected and nothing is marked as a crack in the current percolation.

For reducing the execution time of PCD, seed points are defined only at edges that have not yet been classified as crack pixels and whose gray level is below γ_s. Furthermore, 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 Δ pixels. To ensure that the relevant edges are always considered, a dilation step follows the edge detection, where dilation thickness is in accordance with Δ.

A flowchart of the complete algorithm can be found in Figure 2 and pseudocode is presented as Algorithm 1. This flowchart does not comprise a pre-processing step that has been incorporated to reduce the noise of the image. Due to its excellent properties for preserving relevant edges, this step is implemented as a Bilateral filter [39] (see the Appendix for more details).

Regarding the differences with [47], 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

Table 1: Defect detection techniques

Approach	Particular technique		References
General image processing	Model-based texture analysis		[4–6, 26, 34]
	Filtering	FIR	[22]
		Gabor filters	[40]
		Wavelets	[33, 36]
		Multi-stage non-linear	[44]
	Morphology		[15, 37, 47, 48, 51]
	Edge-based (thresholding + thinning)		[12, 16]
	Characterization of defect geometrical structure	Bayesian reasoning over a network	[7]
		Tortuosity	[49]
		Others	[2, 3, 29, 31, 35]
Learning-based	Neural Network (NN)		[10, 23, 25]
	Backward propagation Neural Network (BPNN)		[50]
	Convolutional Neural Network (CNN)		[28]
	Neural Fuzzy Interference Network (NFIN)		[8]
	Self-Organizing Neural Network (SONN)		[11]
	Statistical mixture model		[45]
	Support Vector Machine (SVM)		[17, 23, 25]
	Self-Organazing Map (SOM)		[19]
	Genetic Algorithm		[28, 50, 51]
	Logistic Regression Tree		[21]

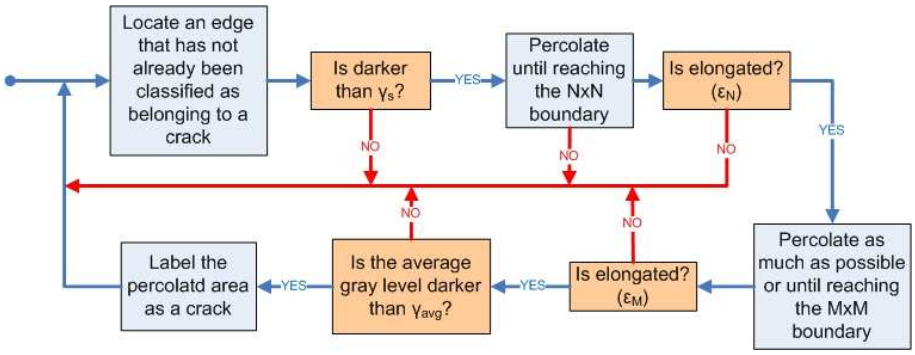


Figure 2: PCD algorithm flowchart

(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, PCD has become faster than Yamaguchi's for crack detection in metal surfaces.

3.2. Performance of PCD

The performance of PCD algorithm depends on the particular setting of the percolation parameters as well as on the size of the gap left between percolations. As a general rule during the selection of the final parameter values, false negatives have been penalized more than false positives. The elapsed time during the entire process has been considered an important factor as well, in the sense of trying to reduce it as much as possible.

Regarding specific parameters, ϵ_N and ϵ_M, related with the expected elongation of cracks, and the gray level thresholds γ_s and γ_{avg}, have all been tuned so as to reduce as much as possible the number of false positives over all the test images, while the value for N and M, related with the size of the percolation boundaries, has been determined using the mean value of Pratt's FOM measure [30] calculated for all the test images.

Once all the parameters have been configured, the detector performance has been assessed with regard to ground truth by means of the false positive and false negative rates, respectively *FP/(FP+TN)* and *FN/(FN+TP)*. After analyzing 2244960 pixels from 15 different images, the global rates result to be 2.31% for false positives and 55.60% for false negatives. The false positive rate is not null because of some dark narrow areas, e.g. shadows, that sometimes are classified as cracks. The high value obtained for the false negative rate is due to the different intensity levels that can be found inside a specific crack: i.e. the percolation process tends to tag only the darkest areas inside the crack, not the lightest ones. Thus, there can be a number of pixels initially labelled as belonging to a crack in the ground truth which finally are not marked by the algorithm.

In order to avoid the effect of ground truth subjectivity over the false negative rate, we decided to make use of another set of performance figures not so much affected by the reduced number of image pixels potentially involved in a crack. In this regard, performance has also been analyzed from the point of view of the false negative and false positive percentages, respectively *FP / #pixels* and

Algorithm 1 Percolation-based Crack Detector (PCD)

Require: Bilateral-filtered gray-scale image I is available. No pixel is labelled as a *crack*

Compute the edge map for I using Sobel operator
for all pixels $p = I(u, v)$ such $u = \lfloor u/\Delta \rfloor \Delta$ **and** $v = \lfloor v/\Delta \rfloor \Delta$ **do**
  $P = \emptyset$ {Currently percolated area}
  if p is darker than γ_s **and** p is an edge **and** p is not labelled as a *crack* **then** {new seed definition}
    $P = P \cup \{p\}$
    T = gray level of p
    while P does not reach $N \times N$ boundary **do** {first percolation process}
      for all neighbours q of P **do**
        if q is darker than T **then**
          $P = P \cup \{q\}$
        end if
      end for
      if any non-percolated neighbour is darker than T **then** {force percolation}
        d = darkest neighbour of P
        $P = P \cup \{d\}$
      end if
    end while
    while P does not reach $M \times M$ boundary **and** there are neighbours to visit **and** *Elongation (P)* $> \epsilon_N$ **do** {second percolation process}
      for all neighbours q of P **do**
        if q is darker than T **then**
          $P = P \cup \{q\}$
        end if
      end for
    end while
    if *Elongation (P)* $> \epsilon_M$ **then**
      if the average gray level of P is darker than γ_{avg} **then**
        Label all the pixels from P as a *crack*
      end if
    end if
  end if
end for

FN / #pixels. Observe that these performance figures are not affected by the percentage of pixels labelled as corrosion since the number of missclassifications are both divided by the same amount of cases (i.e. pixels).

The global percentages result to be 2.29% for false positives and 0.47% for false negatives. Percentages for some of the images are additionally shown in Table 3. As can be observed in Table 3(a), false positive percentages are not null and its value is quite similar to the false positive rate since most of pixels in the images are not from cracks. False negative percentages, as can be seen in Table 3(b), are neither null due to the non-percolation of the lightest pixels of cracks, but its value does not increase so much as the false negative rate since the percentage is calculated taking into account all the pixels of the image, not only crack pixels.

Furthermore, if entire cracks are considered as entities and it is assumed that the labelling of a single pixel within a crack is useful, then the corresponding percentage *FN cracks / #cracks* turns out to be zero since all the cracks are always detected (see Table 3(c)). In this regard, it is important to remember that this algorithm is intended to be used to facilitate the visual inspection of vessel images carried out by a supervisor and, thus, informing about the existence of a crack in an image area is considered worth enough even if the crack is not completely marked.

Table 4 shows crack detection running times for some of the test images, together with their sizes. As can be seen, the elapsed time does not increase with the number of pixels since it depends more on the number and size of cracks, and edges in general, that are detected.

Some images and the corresponding output are shown in Figure 3. It can be observed that all the cracks that can be detected by means of visual inspection are detected by PCD as well. The values assigned to the algorithm parameters are provided through Table 2.

Table 2: Values taken by the PCD parameters

N	41
M	41
ϵ_M	0.3
ϵ_N	0.3
γ_s	0.5
γ_{avg}	0.4
Δ	5

4. Colour-based Corrosion Detector

4.1. Description of the algorithm

The corrosion detection approach has been built around a supervised classification scheme. The classifier makes use of a codeword dictionary computed during a previous learning stage. Each codeword consists of stacked histograms for the red, green and blue colour channels of image patches containing different kinds of rust. This structure gives the name to the algorithm which is called *Color-based corrosion detector*, CCD from now on.

To reduce the dimensionality and the sensitivity to noise, intensity values are downsampled from 256 to 32 levels, and, thus, a codeword consists of 96 components. As can be observed in Figure 4, this codeword does not preserve the spatial arrangement of intensity levels nor the relationship between colour channels for the same pixel.

Samples from different kinds of corrosion have been gathered for training, and 30 different images containing corroded metallic surfaces have been used for this purpose. In order to make the dictionary more compact, codewords have been clustered, independently for every kind of rust considered, by means of the well-known K-means algorithm [38]. The size of the dictionary for every type of corrosion, i.e. the number of models, is therefore given by the number of clusters selected during the clustering process.

Once the dictionary has been built, the corrosion detector proceeds scanning the image and classifying every image patch as affected by corrosion or not. To this end, the current patch codeword is built and compared with the models of the dictionary by means of the Bhattacharyya distance $D = -\log(1 - B)$, with B given by Equation 2:

$$B = \sum_{x \epsilon X} \sqrt{p_c(x) p_m(x)}\,, \tag{2}$$

where X refers to the histograms domain and p_c and p_m are histograms from, respectively, the codeword and the model.

A patch is labelled as corroded as soon as a model is found in the dictionary such that $D < \tau_D$. Therefore, the approach does not intend to determine which is the closest

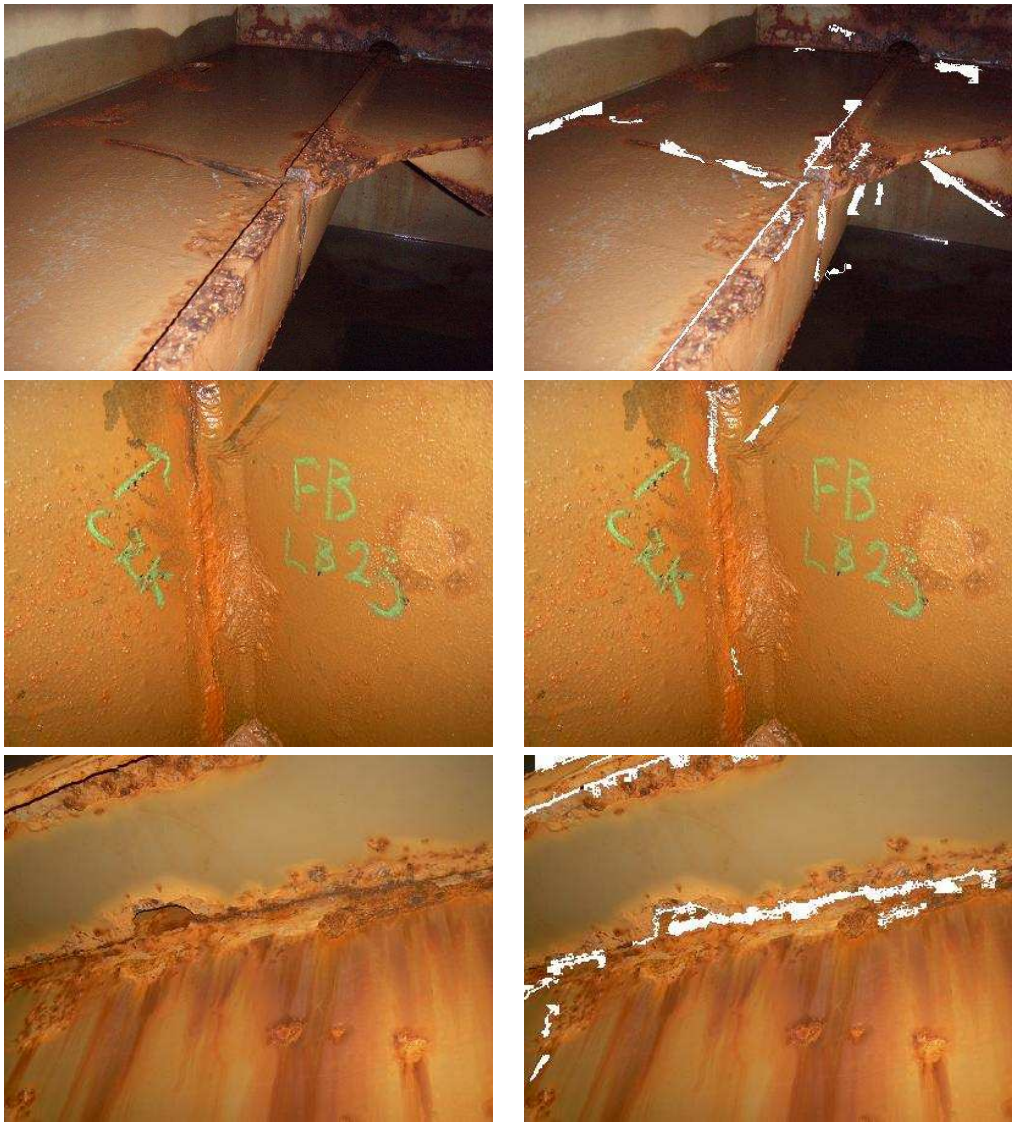


Figure 3: (1st column) Test images and (2nd column) detected cracks

Table 3: (a) Crack pixel FP percentages. (b) Crack pixel FN percentages. (c) Crack FN percentages.

(a)	0.15	2.01	2.58	0.65	1.50	0.76	2.76	0.20	1.81	5.50
(b)	0.02	0.20	0.97	0.42	0.91	0.08	0.20	0.21	1.02	1.57
(c)	0	0	0	0	0	0	0	0	0	0

Table 4: Crack inspection elapsed time for different image sizes

Pixels	120000	120000	144000	144000	158880	162720	172800	172800	177120
Time (ms)	139	438	189	67	816	661	353	117	130

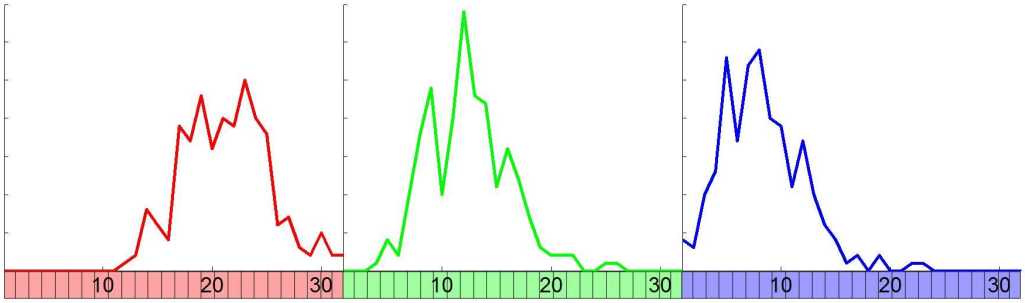


Figure 4: Codeword consisting of red, green and blue stacked histograms

model, but whether the patch is close enough to any model of corrosion. As a consequence, an important reduction in the computation time is obtained.

An additional stage has been introduced before this colour-based classification process in order to improve the classification and reduce the number of codewords to be compared with the dictionary.

This stage is based on the 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 [38]. The energy is obtained by means of Equation 3:

$$E = \sum_{i=0}^{31} \sum_{j=0}^{31} p(i,j)^2 \,, \tag{3}$$

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 flowchart for the complete algorithm is shown in Figure 5 and its pseudocode can be found as Algorithm 3. The pseudocode for codewords dictionary generation can be found as Algorithm 2.

Algorithm 2 Codewords dictionary generation (CCD)

Require: Ground truth images have been generated and different kinds of corrosion have been distinguished

```
for all different kinds of corrosion do
    for all images I with this kind of corrosion do
        for all patches Π of I do
            if Π is labelled as corrosion in the ground truth image then {generate codeword}
                Compute red histogram
                Compute green histogram
                Compute blue histogram
                Stack the three histograms together
                Save codeword
            end if
        end for
    end for
    Cluster codewords to K models using K-means
end for
Save models dictionary
```

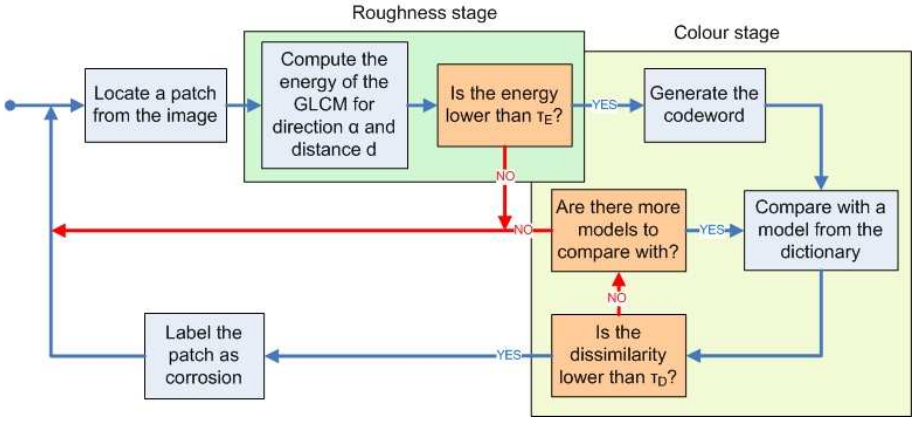


Figure 5: CCD algorithm flowchart

Algorithm 3 Colour-based Corrosion Detector

Require: RGB image and gray-scale image are available. Codewords dictionary has been already generated. No patch is labelled as *corrosion*

Load corrosion dictionary
for all patches Π **do**
  $e = Energy\ (\Pi)$
  if $e < \tau_E$ **then** {Generate the codeword}
    Compute red histogram
    Compute green histogram
    Compute blue histogram
    Stack the three histograms together
    while there are more models in the dictionary **and** the patch has not been labeled as *corrosion* **do**
      Calculate the Bhattacharyya distance D
      if $D < \tau_D$ **then**
        Label Π as *corrosion*
      end if
    end while
  end if
end for

4.2. Performance of CCD

To analyze the performance of CCD algorithm, the performance of the colour stage has been assessed and then compared with the results obtained after adding the roughness stage, in order to measure the improvement achieved by this filter.

Regarding the configuration of colour stage, the patch size has been set to 15 pixels. This value determines the amount of information contained in a patch as well as the necessary time to obtain the codeword from an image patch and to compare it with models from the dictionary. Thus, this parameter also plays a role in the running time. Furthermore, it has to be noticed that the dictionary size, i.e. the number of models in the dictionary, is also a determinant factor in terms of time: the more models are in the dictionary, the more comparisons have to be performed for the patches that actually are not affected by corrosion.

As for the dissimilarity threshold τ_D, a large value gives rise to a reduction in the execution time because image patches affected by corrosion are labelled as such in a few comparisons with the models stored in the dictionary. Nevertheless, this also decreases the performance of the algorithm because the number of false positives grows. The final value for τ_D has been selected so as to reduce as much as possible the misclassifications.

As well as for the PCD crack detector, quantitative performance results have been obtained by defining first the ground truth for every test image by visual inspection, and comparing next the reference with the algorithm output.

Since the corrosion detector output is in terms of image patches instead of image pixels, the ground truth has also been transformed to the patch domain. Since the ground truth patches labelling admits a number of possibilities, the following strategies have been considered:

> *label a ground truth patch as corroded if the number of pixels labelled as corrosion in the ground truth are*
>
> 1. *at least 1,*
> 2. *at least 25% the patch size,*
> 3. *at least 50% the patch size,*
> 4. *at least 75% the patch size, or*
> 5. *all the pixels of the patch.*

False positive (FP / (FP+TN)) and false negative rates (FN / (FN + TP)) for different strategies have been calculated and are shown in Figure 6. These misclassification rates come from the analysis of 7384 patches from 11 different images. As happened with the crack detector, the values obtained for the false negative rate are very high due to the miss detection of some patches labelled as corrosion in the ground truth images. For this reason, misclassification percentages have been used instead of the aforementioned rates, since these indicators are not affected by the percentage of the image that is corroded. False positive (FP / *#patches*) and false negative percentages (FN / *#patches*) are also shown in Figure 6.

After observing the obtained misclassification percentages, the roughness filter was incorporated, its parameters were tuned and its effect on the global algorithm performance was evaluated.

To configure the parameters of the roughness stage some considerations have been taken into account. The value for the energy threshold τ_E affects the algorithm performance in terms of computation time as well as reducing the number of false positives, since all patches with a high energy level are discarded and only those with a low value become input for the color checking step. Several experiments have been performed considering different values for d and α when calculating the GLCM and, consequently, its energy level. However, no significant differences have been observed among the output values, and so the parameter values have been set to $d = 5$ (pixels) and $\alpha = 0$ (horizontal direction).

Figure 7 shows rust detection outputs for the same input image using 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, while decreasing the computation time.

Table 5: Used values for CCD parameters

τ_E	0.05
τ_D	0.07

Once the roughness stage was configured, the performance of the CCD algorithm was assessed. Figure 8 shows the false positive and false negative percentages obtained after executing the algorithm using the same input images used for colour stage. As can be observed, the incorporation of the roughness filter has dramatically reduced the false positive percentage, while the effect on the false negative percentage has not been so important. Comparing the different patch labelling strategies, the number of incorrect classifications in any case is reduced: below 10% for the false positives and around 18% for the false negatives. Besides, the number of false positives grows very slightly with regard to the percentage of pixels that are required to be corroded for the patch to be considered as corroded, what indicates that the number of false alarms is low and independent of how the patches have been labelled. Regarding the number of false negatives, patches really affected by corrosion are always detected and only uncertain cases are left unidentified.

Similarly to what happened with the PCD, the processing time for the corrosion classifier depends on the percentage of image area that is corroded or that presents a low energy level. Some results can be found in this regard in Table 6.

Figure 10 shows detection results for some images. In those cases, the algorithm was configured to detect only the image areas most significantly affected by corrosion, assigning to the parameters the values shown in Table 5.

5. Weak-classifier Colour-based Corrosion Detector

5.1. Description of the algorithm

This section presents a second corrosion detection algorithm. As well as CCD, the new classifier has been

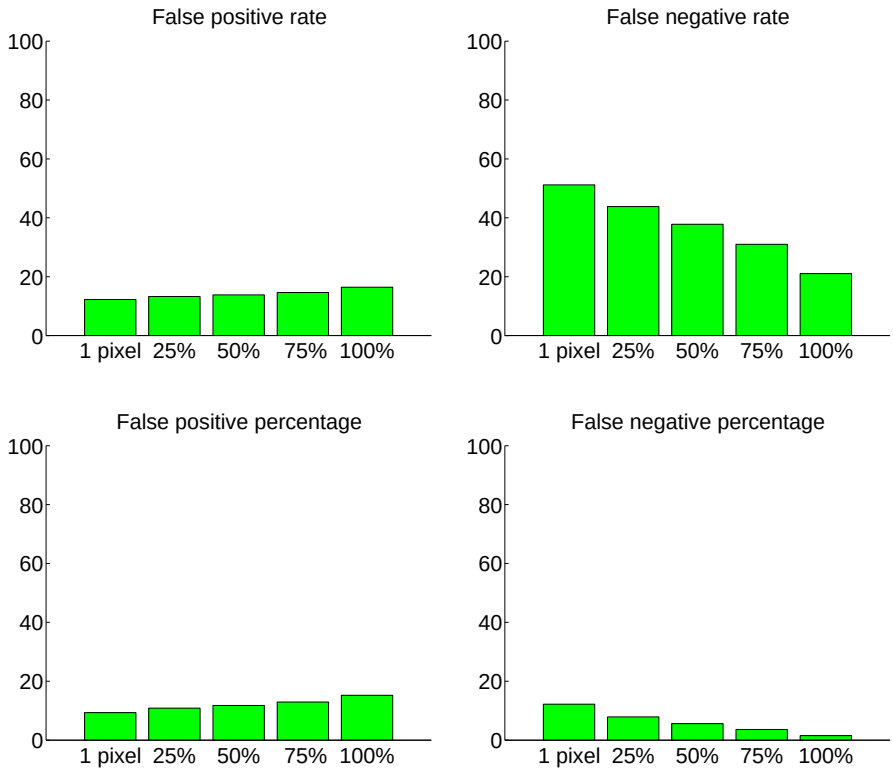


Figure 6: Misclassification rates and percentages for different ground truth patch labelling strategies to asses the color stage performance of CCD

Table 6: Corrosion inspection elapsed time for different image sizes (CCD)

Pixels	120000	120000	144000	154560	162720	172320	172800	172800	177120
Time (ms)	49	47	151	74	61	117	41	277	33

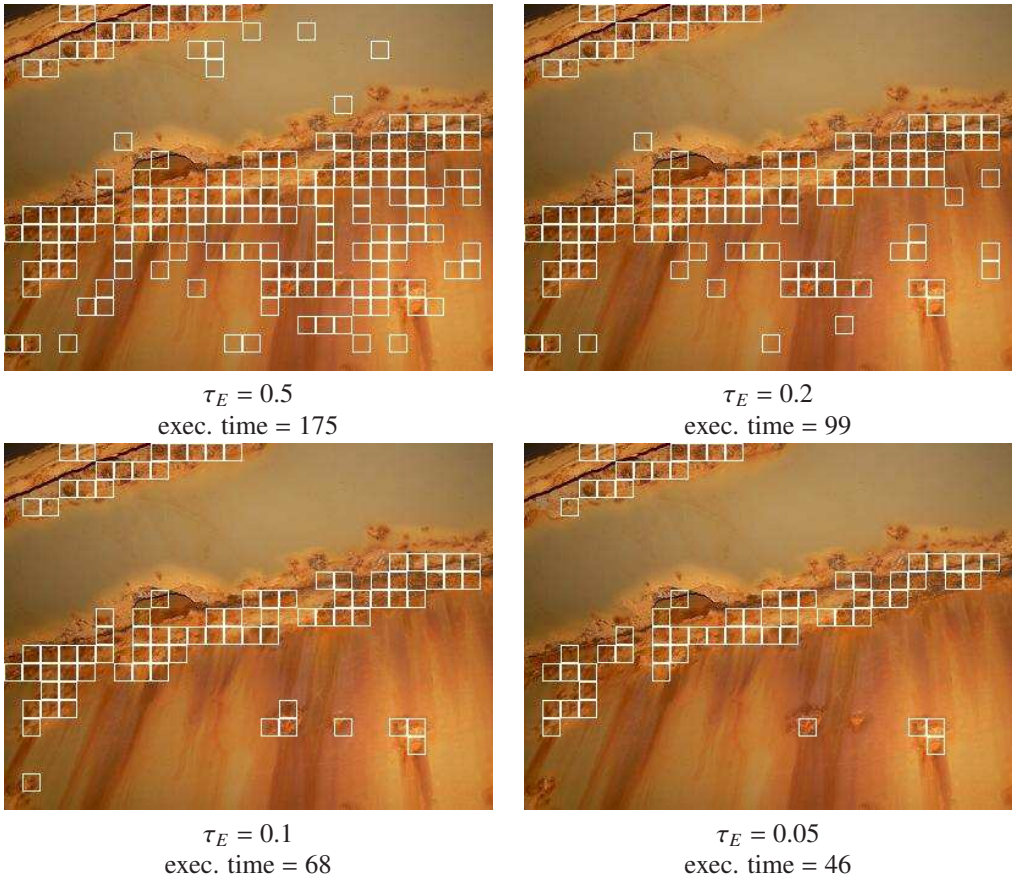


Figure 7: Corroded areas detected by CCD for different τ_E values and corresponding processing times (ms)

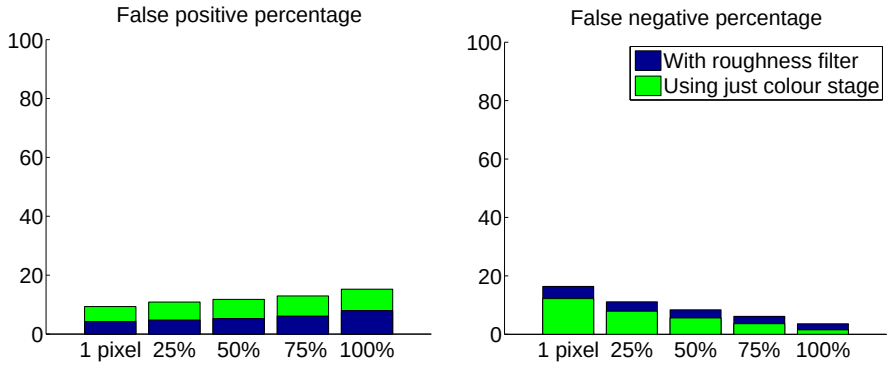


Figure 8: Misclassification percentages with and without the roughness filter (CCD)

built around a cascade scheme, although the two stages of the new corrosion detector can be considered as two weak classifiers. Thus, it will be called *Weak-classifier Colour-based Corrosion Detector*, WCCD from now on. The idea is to chain different fast classifiers with poor performance in order to obtain a global classifier attaining a much better 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 percentage close to zero.

The first stage of the cascade is the roughness stage explained in section 4 for CCD. Remember that this stage returns the image patches which present a rough texture, i.e. have an energy lower than a given threshold τ_E.

The second stage 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. In this case, training consists of building a bi-dimensional histogram of HS values 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 11 shows an HS histogram downsampled between 0 and 31, calculated using 9 different images with corrosion.

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 $HS(h, s) > 0$,

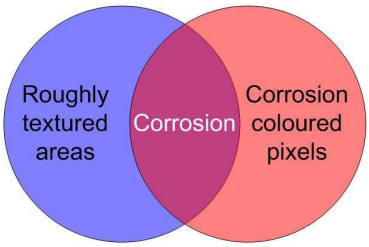


Figure 9: Venn diagram for corrosion definition

for given thresholds mV, MV and mS.

Figure 9 shows a Venn diagram that depicts the corrosion definition used by WCCD. Notice that the stages of the cascade 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. Figure 12 shows the flow diagram of WCCD. The pseudocodes for the dictionary corrosion color generation and WCCD corrosion detection algorithm can be found as Algorithms 4 and 5.

Algorithm 4 Corrosion color dictionary generation (WCCD)

Require: Ground truth images have been generated

```
for all images C do
    for all pixels p do
        if p is labelled as corrosion in the ground truth image then
            Insert that pixel in HS histogram
        end if
    end for
end for
for all bins b of the histogram do
    if b is lower than 0.1 × max(HS) then
        b = 0
    end if
end for
Save the histogram
```

5.2. Performance of WCCD

The performance of WCCD depends on the performance of its different stages. Regarding the roughness

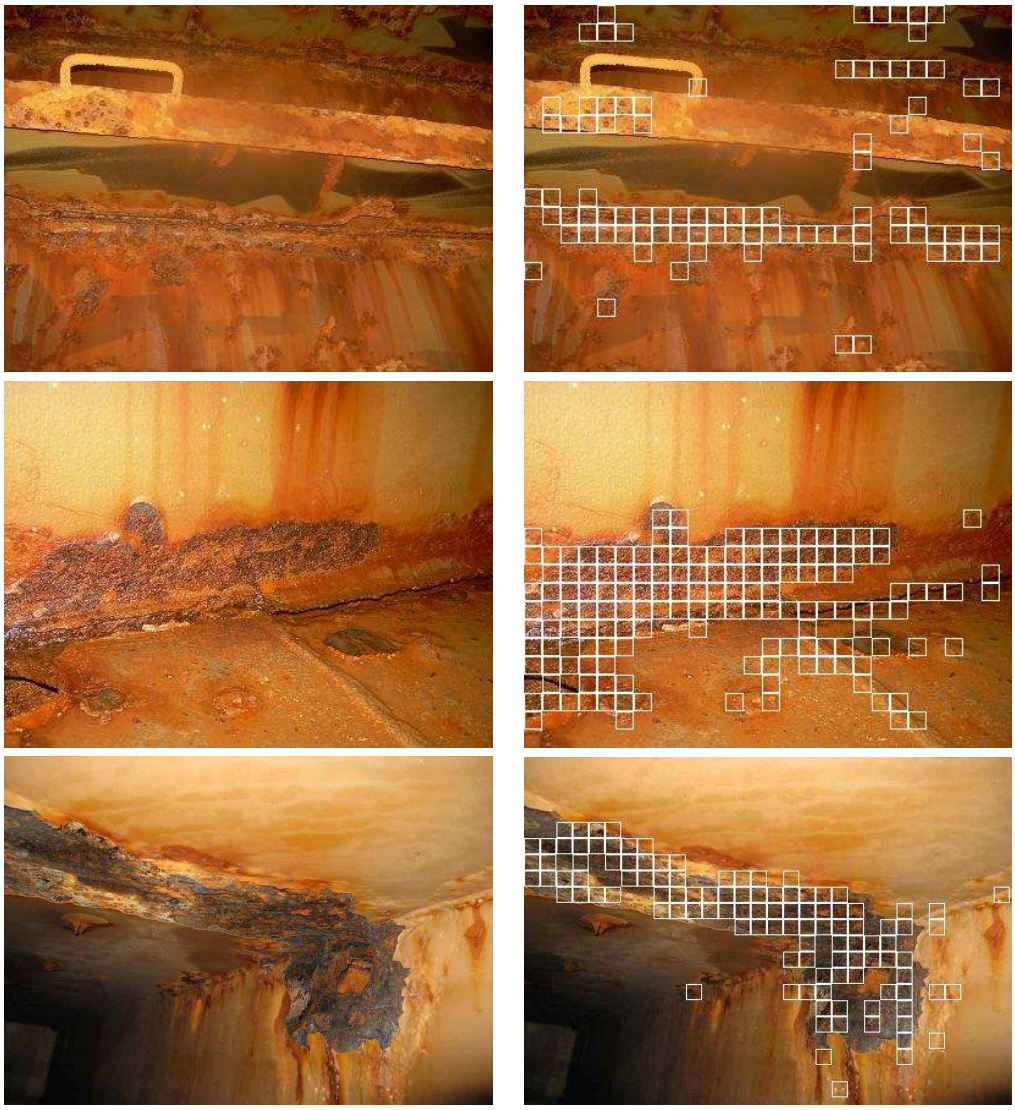


Figure 10: (1st column) Test images and (2nd column) corroded areas detected by CCD

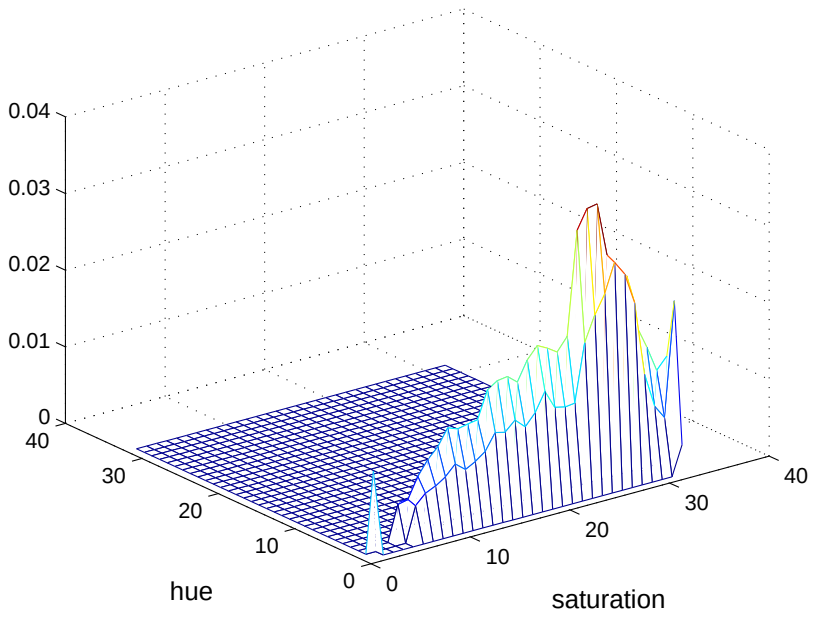


Figure 11: Hue-Saturation histogram for the colour stage of WCCD

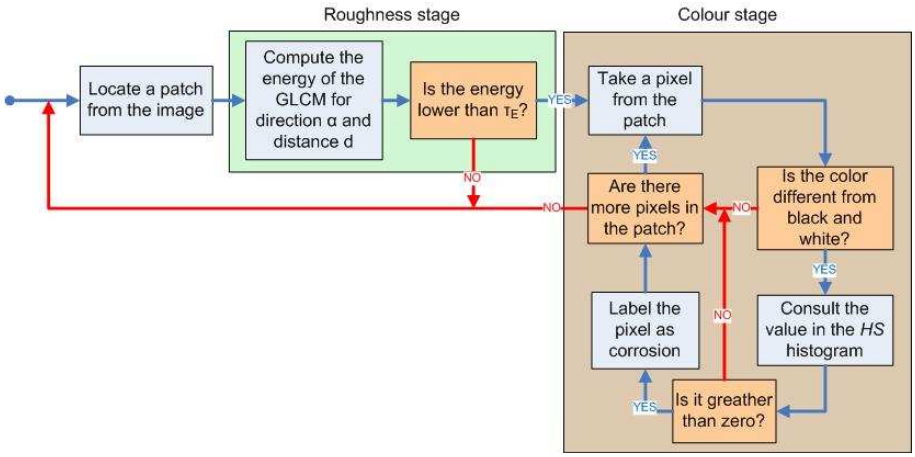


Figure 12: WCCD algorithm flowchart

Algorithm 5 Weak-classifier Colour-based Corrosion Detector (WCCD)

Require: HSV image and gray-scale image have been already obtained. HS histogram has been already generated. No pixel is labelled as *corrosion*

for all patches Π **do**
  $e = Energy\ (\Pi)$
  if $e < \tau_E$ **then**
    for all pixels $p = (h, s, v)$ from Π **do**
      if $v > mV$ **and** ($v < MV$ **or** $s > mS$) **then** {p is neither black nor white}
        if $HS(h, s) \geq 0.75\mathbb{HS}$ **then** {where $\mathbb{HS}$ is the highest peak of the HS histogram}
          Label pixel as *corrosion* in red
        else if $HS(h, s) \geq 0.5\mathbb{HS}$ **then**
          Label pixel as *corrosion* in orange
        else if $HS(h, s) \geq 0.25\mathbb{HS}$ **then**
          Label pixel as *corrosion* in green
        else if $HS(h, s) \geq 0$ **then**
          Label pixel as *corrosion* in blue
        end if
      end if
    end for
  end if
end for

stage, the energy threshold τ_E and the parameters d and α play exactly the same role as for CCD, thus they have been configured to the same values.

Figure 13 provides classification outputs for the same input image using different energy thresholds. As happened with CCD, τ_E can be tuned to decrease false positives and just allow the detection of the most significant corroded areas. In the images, a pixel labeled as corroded is 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 $HS(h, s) \in [0.75\mathbb{HS}, 1.00\mathbb{HS}]$,
- 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 the instabilities of h and s values from affecting the pixel classification. 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.

In order to evaluate the performance of roughness stage, misclassification percentages have been obtained for stand-alone configuration, i.e. without chaining any other stage behind. Figure 14 compares the false positive percentage (*FP / #patches*) and false negative percentage (*FN / #patches*) obtained after the execution of the roughness stage with the percentages obtained with CCD, that have been already presented in Figure 8. As can be seen, the false positive percentage is always higher for roughness stage, while false negative percentage is always lower. These results are easy to explain since the roughness stage is also the first stage of CCD.

The performance of the complete corrosion detector has been determined after combining the roughness stage with the colour stage and analyzing the same 7384 patches used to asses the performance of CCD. The process carried out to perform this assessment has entailed the implementation of different techniques to filter the HS histogram and a posterior comparison among the alternatives with regard to the generalization capability of the resulting classifier. The indicators used for this comparison

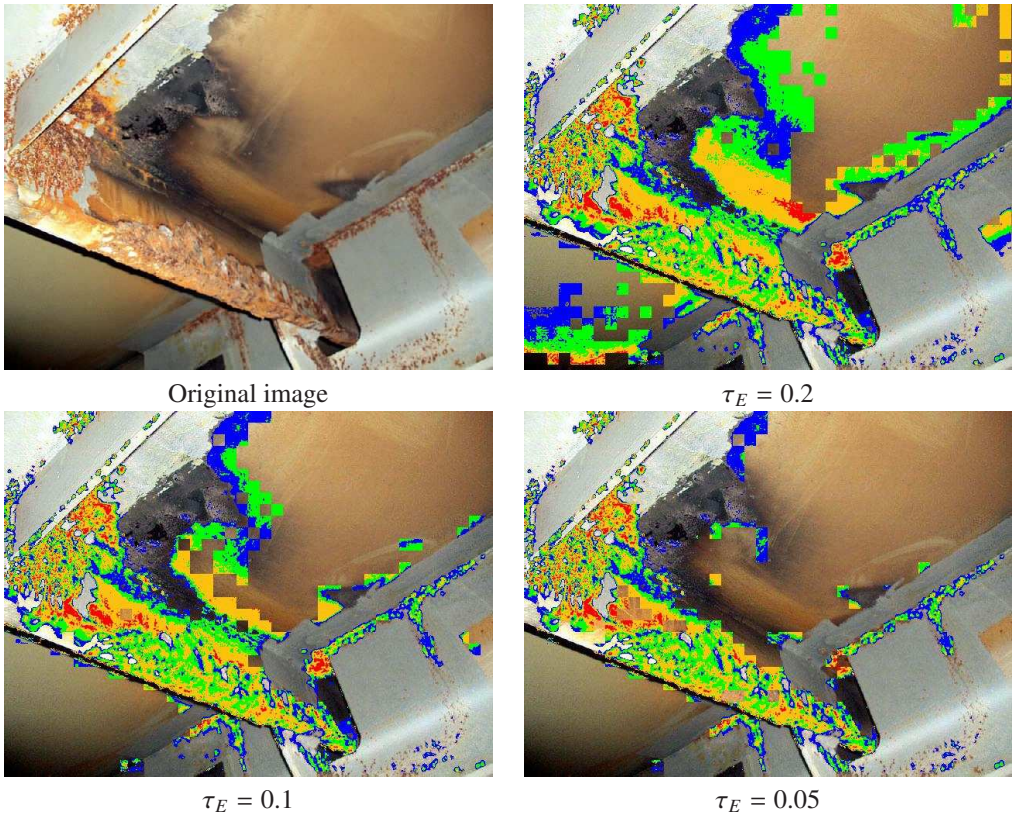


Figure 13: Corroded areas detected by WCCD for different energy threshold values

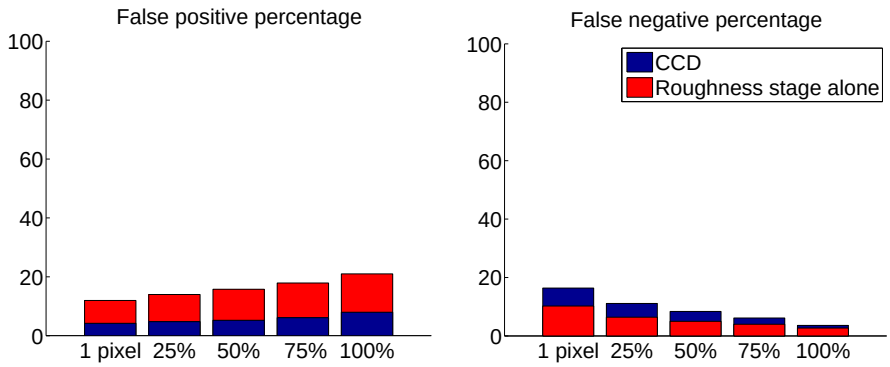


Figure 14: Misclassification percentages for roughness stage in comparison with the ones obtained with CCD

have been once more the false positive rate (FP / (FP + TN)), the false negative rate (FN / (FN + TP)), the false positive percentage (*FP / #pixels*) and the false negative percentage (*FP / #pixels*). Nevertheless, as happened before with CCD, the uncertainty during ground truth definition has derived in high false negative rates, thus misclassification percentages have been considered better indicators of the algorithm performance.

Downsampling the histogram to 32 levels for hue and saturation has been the first filter considered, which merely groups bins with similar hue-saturation values. As can be seen in the first and second rows of Table 7, this filter has resulted a considerable improvement in comparison with the original 256×256 HS histogram, thus it has been considered as the reference for comparing with the next filtering strategies.

More specifically, two more attempts have been performed in order to reduce the false negative percentage while preserving the false positive percentage. On the one hand, the Parzen windows method [38] has been applied to the original 256×256 histogram using the two-dimensional Gaussian kernel shown in Equation 4:

$$G(x, y, \sigma) = \frac{1}{2\pi\,\sigma^2}\, e^{-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma^2} + \frac{(y-\mu_y)^2}{\sigma^2}\right)}, \quad (4)$$

where $\mu_x and \mu_y$ are the hue and saturation values for the neighbourhood center, x and y are the values for a nearby sample, and σ is the standard deviation. The algorithm performance has been assessed filtering the histogram using different values for σ. Best misclassification rates and percentages are shown in the third row of Table 7, which correspond to $\sigma = 12$.

On the other hand, a Bilateral filter [39] has been applied to the original 256×256 histogram, considering the bins height as the intensity values of an image. This approach filters the histogram using a kernel consisting of two Gaussians, one for the spatial domain and another for the range domain (see the Appendix for more details). After the different experiments carried out, the best performance has been obtained for $\sigma_{spatial} = 15$, $\sigma_{range} = 1$ and a kernel size of 30 pixels. The fourth row of Table 7 provides the resulting performance values. The resulting histograms for the different filters can be found in Figure 15. By way of conclusion, it seems the approach based on the bilateral filter is the one providing best results, although it is true the different strategies lead to a final similar performance. The bilateral filter has thus been selected for being part of WCCD.

Examples of final classification outputs for WCCD are provided in Figure 16. Among the different images shown, special mention is done for the image in the first row, where a specific kind of corrosion called *pitting*, affecting a very reduced fraction of the image, is successfully detected.

Regarding execution times, WCCD took between 7 and 15 ms for images ranging from 120.000 to 172.800 pixels. A comparison with the execution times of CCD can be found in Table 8. A 7.76x speed up has been observed after testing with 11 different images.

To sum up, comparing the performance of both corrosion detectors, the method explained in this section has presented false positive and false negative percentages of the same order as the percentages obtained with the first method. Nevertheless, the execution time has been considerably reduced. For these reasons, one can conclude stating that WCCD is preferable to CCD.

6. Guided Percolation-based Crack Detector

6.1. Description of the algorithm

In order to improve the performance of PCD, described in Section 3, a new detector is proposed that combines WCCD with the percolation-based strategy of PCD. 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. This combination is called *Guided-PCD* or GPCD from now on.

WCCD has been selected for taking part in the combination since it presents an acceptable performance and a reduced execution time. Furthermore, WCCD works at the pixel level, instead of at the patch level, what facilitates the combination with the crack detection algorithm.

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. The pseudocode for this improved version can be found as Algorithm 6.

Table 7: Misclassification measures for different HS histograms (WCCD)

	FP rate	FN rate	FP percentage	FN percentage
Original (256 bins)	0.94	91.22	0.80	13.56
Downsampled to 32 bins	11.50	41.02	9.78	6.10
Using Parzen-window	10.85	40.31	9.23	5.99
Using Bilateral filter	11.51	39.45	9.80	5.86

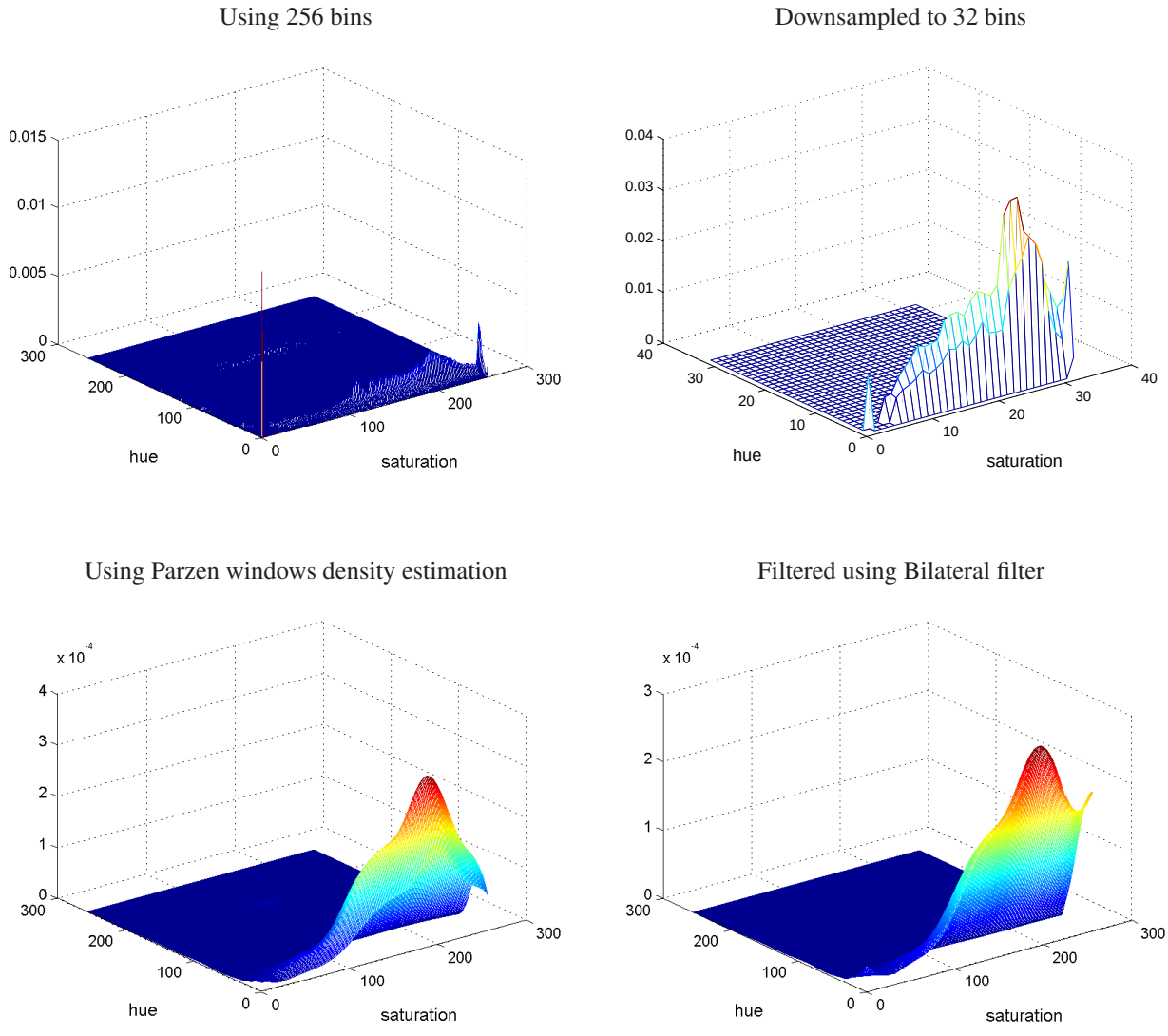


Figure 15: HS histograms resulting from the different filtering strategies (WCCD)

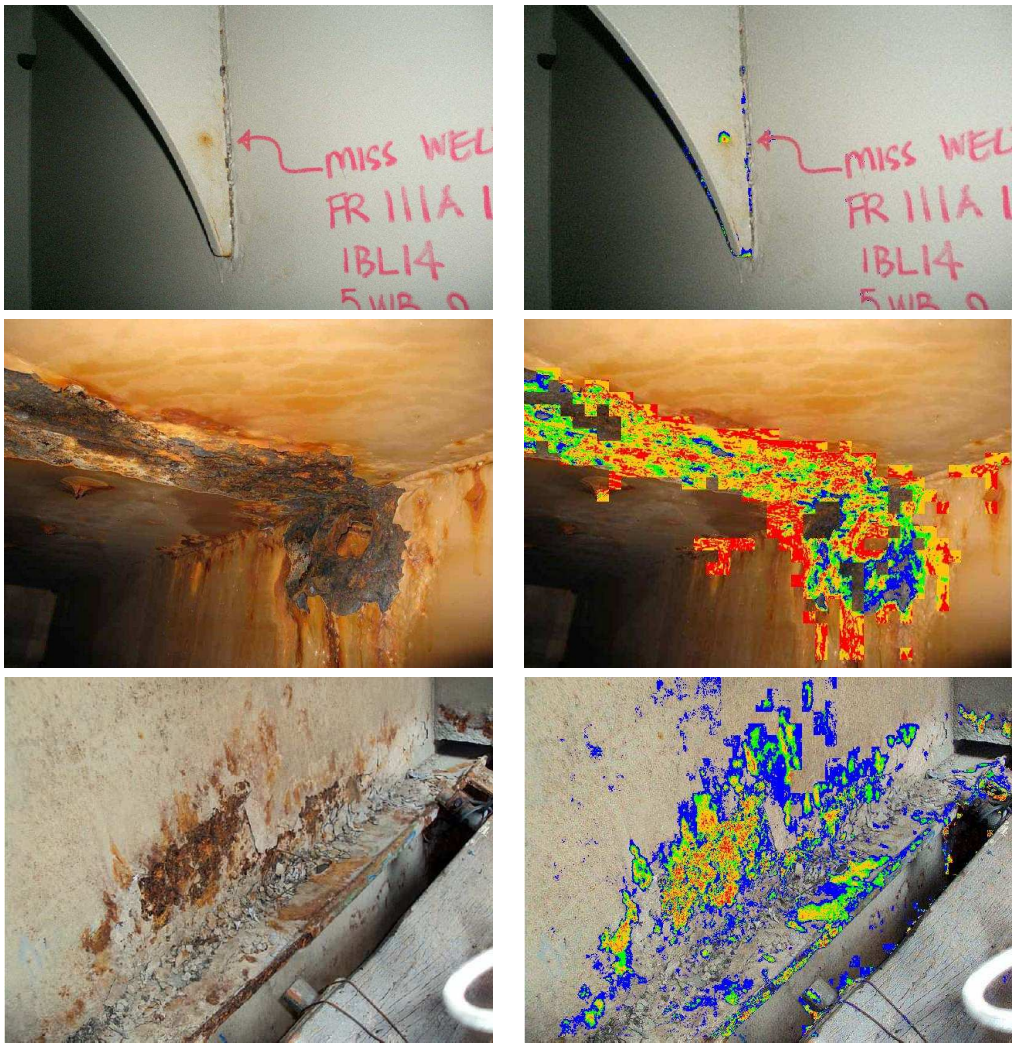


Figure 16: (1st column) Test images and (2nd column) corroded areas detected by WCCD

Table 8: Comparison between elapsed times using CCD and WCCD

Pixels	120000	120000	144000	154560	162720	172320	172800	172800	177120
CCD (ms)	52	42	153	66	56	122	41	196	33
WCCD (ms)	8	7	10	11	10	12	10	15	10

Algorithm 6 Guided Percolation-based Crack Detector (GPCD)

Require: Bilateral-filtered gray-scale image I is available. No pixel is labelled as a *crack*

1: Call to *WCCD* (Algorithm 5)
2: Compute the edge map from I using Sobel operator
3: **for** all pixels $p = I(u, v)$ such $u = \lfloor u/\Delta \rfloor \Delta$ **and** $v = \lfloor v/\Delta \rfloor \Delta$ **do**
4: **if** p is darker than γ_s **and** p is an edge **and** p is not labelled as a *crack* **and** p is labelled as *corroded* **then** {new seed definition}
5: Proceed as in Algorithm 1, lines 4-29
6: **end if**
7: **end for**

6.2. Performance of GPCD

Figure 17 provides some classification outputs for the unguided and guided versions of the crack detector. As can be observed, the computation time falls below 50% for GPCD with regard to PCD. In fact, after testing over 15 different images, a 3.13x speed up has been observed when using the guided version. Some values in this regard are shown in Table 9. Moreover, it is also worth observing that the time reported for guided detection does include the execution of the WCCD 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. The values for the false positive and false negative percentages are 0.72% and 0.57%, respectively. Comparing with the percentages obtained without guidance, 2.29% and 0.47%, the improvement is obvious.

As happened with all other detectors presented in this report, the uncertainty defining ground truth images causes a high false negative rate (67.20%), while false positive rate is similar to the false positive percentage (0.73%).

As can be seen in Figure 17, both PCD and GPCD detect all the cracks from the images but the unguided version also marks other elongated, narrow and dark zones, e.g. shadows. GPCD, however, prevents the percolation of some of these false positives since corrosion has not been detected there. Tables 2 and 5 indicates the parameter values used to obtain the results shown in Figure 17 for, respectively, PCD/GPCD and WCCD.

7. Conclusions

Several defect detection algorithms for support in vessel hull inspection have been presented in this report. On the one hand, a crack detection method based on a percolation idea, PCD, has been introduced and its performance has been observed in the sense of being able to detect all the cracks in the test images. Nevertheless, shadows and other crack-like collections of pixels were misclassified, increasing the false positive percentage.

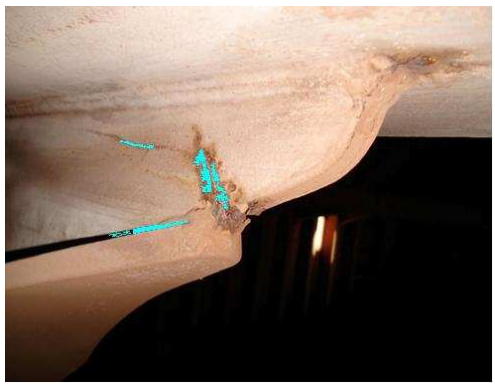
On the other hand, two different corrosion detection algorithms have been introduced. The first one, CCD, is based on a supervised classification method that takes profit from the distribution of color in corroded areas, and has been significantly improved with the addition of a previous decision stage based on a texture roughness criterion. The second corrosion detection algorithm, WCCD, has been built around a weak classifiers approach, separating the spatial and colour analysis in two different stages. Both algorithms (CCD and WCCD) have presented good performance, with similar misclassification percentages, but WCCD performs the detection more than seven times before than CCD.

As an additional contribution, WCCD has been used to improve the performance of PCD. The results obtained prove that the guided approach, GPCD, reduces more than three times the computation time with regard to the unguided version, while the number of false positive detections is also reduced, and the classifier reliability increased.

Better results are expected for all the algorithms proposed by controlling the imaging process. If the pose and distance of the camera respect to the inspected surface could be controlled, the crack detection algorithms parameters could be tuned for cracks of a specific size, as well as the patch size could be adjusted for improving the corrosion detection algorithms performance.

As for future work, WCCD is planned to be enhanced by means of an *Adaptative Boosting* (AdaBoost) [38] 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

exec. time = 161 ms



exec. time = 102 ms

exec. time = 149 ms

exec. time = 73 ms

exec. time = 224 ms

exec. time = 97 ms

Figure 17: Cracks detected by PCD (1st column) and GPCD (2nd column)

Table 9: Comparison between elapsed times using PCD and GPCD

Pixels	120000	120000	144000	144000	157440	158880	162720	172800	177120
PCD (ms)	140	412	190	68	182	795	656	342	111
GPCD (ms)	63	182	47	33	50	37	104	159	58

classified samples of previous stages. This strategy is successfully used by the Viola-Jones classifier [24, 42, 43] which is able to robustly detect complex structures, e.g. faces, in real-time.

Another research line to be investigated is around the concept of *saliency* [20]. Considering corrosion and cracks as anomalies over metallic surfaces, saliency maps turn out to be relevant tools to improve their detection. Furthermore, some other works, e.g. see [27, 46], gives other ways to work with corrosion that could be useful for that research.

Acknowledgements

This work is partially supported by FP7 project SCP8-GA-2009-233715 (MINOAS).

The author is 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, and specially grateful to his tutor Alberto Ortiz, for his help and guidance.

Appendix: The Bilateral filter

The Bilateral filter is a simple filter that is very used in image processing for noise reduction while preserving the edges that appear in the image. It consists in a combination of two Gaussian filters. One works in the spatial domain, thus takes into account the distance between a pixel and its neighbours; the other one works in the range domain since it takes into account the difference between the value (e.g. intensity) of a pixel and the values of its neighbours.

$$I_p^{bf} = \frac{1}{W_p^{bf}} \sum_{q \epsilon S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(| I_p - I_q |) I_q$$

where

$$W_p^{bf} = \sum_{q \epsilon S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(| I_p - I_q |)$$

$$G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}}$$

and S is set of pixels involved in the convolution.

References

[1] T. S. Akinfiev, M. A. Armada, and R. Fernandez. Nondestructive testing of the state of a ships hull with an underwater robot. *Russian Journal of Nondestructive Testing*, 44(9):626–633, 2008.

[2] T. Amano. Correlation based image defect detection. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages 163–166, 2006.

[3] S. Avril, A. Vautrin, and Y. Surrel. Grid method: Application to the characterization of cracks. *Experimental Mechanics*, 44(1):37–43, 2004.

[4] A. Baykuta, A. Atalay, A. Ercil, and M. Guler. Real-time defect inspection of textured surfaces. *Real-Time Imaging*, 6(1):17–27, 2000.

[5] C. Boukouvalas, J. Kittler, R. Marik, M. Mirmehdi, and M. Petrou. Ceramic tile inspection for colour and structural defects. *Technical Report CS-EXT-1995-052*, 1995. Also: Proceedings of AMPT95, pp. 390-399, 1995.

[6] A. Branca, F. P. Lovergine, G. Attolico, and A. Distante. Defect detection on leather by oriented singularities. *Computer Analysis of Images and Patterns*, pages 223–230, 1997.

[7] N. Bryson, R. Dixon, J. Hunter, and C. Taylor. Contextual classification of cracks. *Image and Vision Computing*, 12(3):149–154, 1994.

[8] C.-P. Hsu C.-L. Chang, H.-H. Chang. An intelligent defect inspection technique for color filter. In *Proceedings of the IEEE International Conference on Mechatronics*, pages 933–936, 2005.

[9] A. Carvalho, L. Sagrilo, I. Silva, J. Rebello, and R. Carneval. On the reliability of an automated ultrasonic system for hull inspection in ship-based oil production units. *Applied Ocean Research*, 25(5):235–241, 2003.

[10] H. Castilho, J. Pinto, and A. Limas. An automated defect detection based on optimized thresholding. In *Proceedings of the International Conference on Image Analysis and Recognition*, volume 4142 of *Lecture Notes in Computer Science*, pages II:790–801, 2006.

[11] C.-Y. Chang, J.-W. Chang, and M. D. Jeng. An unsupervised self-organizing neural network for automatic semiconductor wafer defect inspection. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3000– 3005, 2005.

[12] S. Cho, K. Hisatomi, and S. Hashimoto. Cracks and displacement feature extraction of the concrete block surface. In *Proceedings of the IAPR Workshop on Machine Vision Applications*, pages 246–249, 1998.

[13] A. Cormack. Ship hull inspections using AquaMap. *Seventh International Symposium on Technology and the Mine Problem*, 2006.

[14] R. Damus, S. Desset, J. Morash, V. Polidoro, F. Hover, and C. Chryssostomidis. A new paradigm for ship hull inspection using a holonomic hover-capable auv. *Informatics in Control, Automation and Robotics I*, pages 195–200, 2006.

[15] Y. Fucheng, Z. Lifan, and Z. Yongbin. The research of printing's image defect inspection based on machine vision. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 2404–2408, 2009.

[16] Y. Fujita, Y. Mitani, and Y. Hamamoto. A method for crack detection on a concrete structure. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages III: 901–904, 2006.

[17] J. Hongbin, Y. Murphey, S. Jinajun, and C. Tzyy-Shuh. An intelligent real-time vision system for surface defect detection. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages III:239–242, 2004.

[18] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[19] J. Iivarinen and A. Visa. An adaptive texture and shape based defect classification. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages I:117–122, 1998.

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

[21] B. C. Jiang, C. C. Wang, and P. L. Chen. Logistic regression tree applied to classify pcb golden finger defects. *The International Journal of Advanced Manufacturing Technology*, 24(7-8):496–502, 2004.

[22] A. Kumar and G. Pang. Defect detection in textured materials using optimized filters. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 32(5):553–570, 2002.

[23] A. Kumar and H. Shen. Texture inspection for defects using neural networks and support vector machines. In *Proceedings of the IEEE International Conference on Image Processing*, pages III:353–356, 2002.

[24] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of IEEE International Conference on Image Processing*, pages I: 900–903, 2002.

[25] L. Liu and G. Meng. Crack detection in supported beams based on neural network and support vector machine. In *Advances in Neural Networks*, volume 3498 of *Lecture Notes in Computer Science*, pages 597–602, 2005.

[26] C.-J. Lu and D.-M. Tsai. Automated defects inspection for lcd using singular value decomposition. *Internal Journal of Advanced Manufacturing Technology*, 25(1-2):53–61, 2005.

[27] D. Martin, D. M. Guinea, M. C. Garca-Alegre, E.Villanueva, and D. Guinea. Multi-modal defect detection of residual oxide scale on a cold stainless steel strip. *Machine Vision and Applications*, 21(5):653 – 666, 2010.

[28] R. Oullette, M. Browne, and K. Hirasawa. Genetic algorithm optimization of a convolutional neural network for autonomous crack detection. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages I:516 – 521, 2004.

[29] P. Perner. A knowledge-based image-inspection system for automatic defect recognition, classification, and process diagnosis. *Machine Vision and Applications*, 7(3):135–147, 1994.

[30] W. Pratt. *Digital Image Processing*. John Wiley and Sons, 2nd edition, 1991.

[31] M. A. Rodrigues and Y. Liu. A novel 3d-2d computer vision algorithm for automatic inspection of filter components. In *Proceedings of the international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 560–569, 1999.

[32] S.Negahdaripour and P. Firoozfam. An rov stereovision system for ship hull inspection. *International Journal of Oceanic Engineering*, 31(3):551–564, 2006.

[33] J. Sobral. Optimised filters for texture defect detection. In *Proceedings of the IEEE International Conference on Image Processing*, pages III:565–568, 2005.

[34] K. Song, M. Petrou, and J. Kittler. Texture crack detection. *Machine Vision and Applications*, 8(1):63–76, 1995.

[35] S. Sorncharean and S. Phiphobmongkol. Crack detection on asphalt surface image using enhanced grid cell analysis. In *Proceedings of IEEE International Symposium on Electronic Design, Test & Applications*, pages 49–54, 2008.

[36] P. Subirats, J. Dumoulin, V. Legeay, and D. Barba. Automation of pavement surface crack detection using the continuous wavelet transform. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3037–3040, 2006.

[37] N. Tanaka and K. Uematsu. A crack detection method in road surface images using morphology. In *Proceedings of the IAPR Workshop on Machine Vision Applications*, pages 154–157, 1998.

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

[39] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 839 – 846, 1998.

[40] D.-M. Tsai and S.-K. Wu. Automated surface inspection using gabor filters. *Internal Journal of Advanced Manufacturing Technology*, 16(7):474–482, 2000.

[41] J. Vaganay, M. Elkins, S. Willcox, F. Hover, R. Damus, S. Desset, J. Morash, and V.Pollidoro. Ship hull inspection by hull-relative navigation and control. In *Proceedings of MTS/IEEE Oceans*, pages I: 761–766, 2005.

[42] P. Viola and M. Jones. Robust real-time object detection. In *Proceedings of International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing and Sampling*, 2001.

[43] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[44] P. Woods and P. Allen. A cue generator for crack detection. *Image and Vision Computing*, 7(4):268–273, 1989.

[45] X. Xie and M. Mirmehdi. Localising surface defects in random colour textures using multiscale texem

analysis in image eigenchannels. In *Proceedings of the IEEE International Conference on Image Processing*, pages III: 1124–1127, 2005.

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

[47] T. Yamaguchi and S. Hashimoto. Fast crack detection method for large-size concrete surface images using percolation-based image processing. *Machine Vision and Applications*, 21(5):797–809, 2010.

[48] M. Yoshioka and S. Omatu. Defect detection method using rotational morphology. *Artificial Life and Robotics*, 14(1):20–23, 2009.

[49] T. Zhang and G. Nagy. Surface tortuosity and its application to analyzing cracks in concrete. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages 851–854, 2004.

[50] X. Zhang, R. Liang, Y. Ding, J. Chen, D. Duan, and G. Zong. The system of copper strips surface defects inspection based on intelligent fusion. In *Proceedings of the IEEE International Conference on Automation and Logistics*, pages 476–480, 2008.

[51] H. Zheng, L. Kongb, and S. Nahavandia. Automatic inspection of metallic surface defects using genetic algorithms. *Journal of Materials Processing Technology*, 125-126:427–433, 2002.