Detection of Cracks and Corrosion for Automated Vessels Visual Inspection

Francisco BONNÍN-PASCUAL [1] and Alberto ORTIZ
Dep. of Mathematics and Computer Science, University of Balearic Islands, Spain

Abstract. Vessel maintenance entails periodic visual inspections of internal and external parts of the vessel hull in order to detect cracks and corroded areas. Typically, this is done by trained surveyors at great cost. Clearly, assisting them during the inspection process by means of a fleet of robots capable of defect detection would decrease the inspection cost. In this paper, two algorithms are presented for visual detection of the aforementioned two kinds of defects. On the one hand, the crack detector is based on a percolation process that exploits the morphological properties of cracks in steel surfaces. On the other hand, the corrosion detector follows a supervised classification approach taking profit from the spatial distribution of color in rusty areas. Both algorithms have shown successful rates of detection with close to real-time performance.

Keywords. Crack detection, Percolation, Corrosion detection, Classification, Vessel inspection.

Introduction

Nowadays, vessels constitute the most cost effective form of transporting people and 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.

This paper presents two different algorithms for the visual detection of these kind of defects during vessel inspection. On the one hand, the cracks detector takes into account the particular geometry of cracks as they appear in digital images, i.e. elongated, narrow and connected sets of dark pixels, and, to this end, makes use of a region-growing

[1]Corresponding Author: Ed. Anselm Turmeda - Campus UIB, Ctra. Valldemossa, km. 7.5, Palma, Spain; E-mail: xisco.bonnin@uib.es.

technique based on a percolation model, a physical model for liquid permeation through porous materials (see [10]).

On the other hand, the corrosion detector 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. A nearest-neighbour rule is used during the classification stage to determine which parts of the image correspond to corroded areas.

The rest of the paper is organized as follows: Section 1 raises the problem and revises previous work, Sections 2 and 3 describe the algorithms developed for, respectively, crack and corrosion detection, Section 4 presents and discusses the results obtained in a set of experiments and, finally, Section 5 concludes the paper.

1. Description of the Problem and Related Work

Problem description. To perform a complete hull inspection, the vessel has to be emptied and situated in a dockyard, where typically temporary and permanent 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 more likely 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[right]).

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)

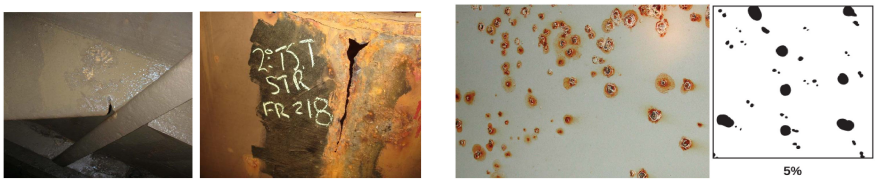

Figure 1. Defective situations to be detected.

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.

Previous work. So far, the problem of visual defect detection in materials has been addressed from different perspectives. Roughly speaking, two main categories can be distinguished: learning-based approaches and image processing approaches. While the former makes use of typical pattern recognition tools, such as neural networks [1,5,17], self-organizing maps [4] or support vector machines [2,5,6], the latter mostly comprises methods based on image filtering (e.g. Gabor [13] or wavelets-based approaches [8,11]) and methods taking advantage of the particular geometry of the defect [15,16].

To the best of authors' knowledge, only the work described in [14] refers specifically to detecting corrosion in metals. However, a number of works can be found for the detection of cracks, mostly for concrete [6,15], asphalt [9] or pavement [11]. In particular, the geometric method described in [15], based on a percolation model, is the point of departure for the method presented in this paper for the detection of cracks. Regarding the detector of corrosion, a learning-based approach is followed.

2. Crack Detection

This section presents a crack detector based on a percolation model, as well as the algorithm by Yamaguchi and Hashimoto described in [15]. 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 has been speeded up so that the algorithm can run as close as possible to real-time onboard a robotic platform. A flowchart of the algorithm can be found in Figure 2.

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

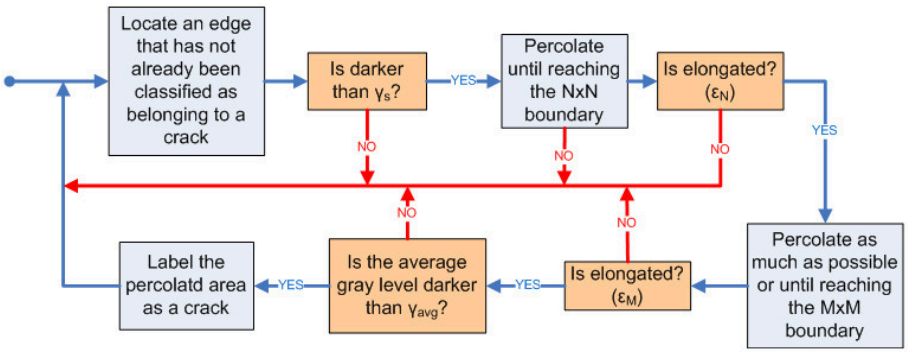

Figure 2. Crack detection algorithm flowchart

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:

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

where μ_{xx}, μ_{yy} and μ_{xy} are the normalized second central moments of the region [3].

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 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.

Regarding the differences with [15], 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.

3. Corrosion Detection

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. To reduce the dimensionality, intensity values are downsampled from 256 to 32 levels, and, thus, a codeword consists of 96 components. As can be observed, 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. 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 [12]. The size of the dictionary, 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 all 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 patch domain and p_c and p_m are histograms from, respectively, the codeword and the model. Using a nearest-neighbour rule, the closest model is selected and the patch is classified as corrosion only if $D < \tau_D$.

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. The symmetric *gray-level co-occurrence matrix* (GLCM) for each patch is calculated for gray values scaled between 0 and 31, for a given direction α and distance d [12]. The co-occurrence matrix is next used to calculate the patch energy 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 value stored in row i and column j of the co-occurrence matrix. Patches of low energy, i.e. have a rough texture, are finally candidates to be more deeply inspected. The flowchart for corrosion detection is shown in Figure 3.

4. Experimental results

4.1. Detecting cracks

The performance of the crack detector 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

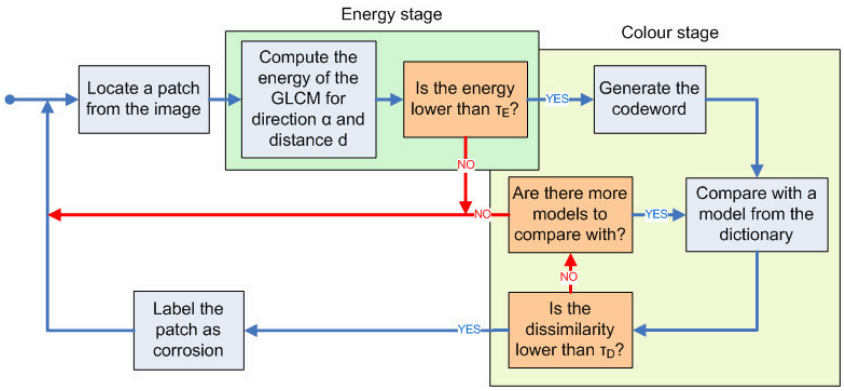

Figure 3. Corrosion detection algorithm

Table 1. (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

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 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 [7] 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 percentages, respectively *FP / #pixels* and *FN / #pixels*. After analyzing 2244960 pixels from 15 different images, 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 1. As can be observed in Table 1(a), false positive percentages are not null. This is because of some dark narrow areas, e.g. shadows, that sometimes are classified as cracks. False negative percentages, as can be seen in Table 1(b), are neither null 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. Nevertheless, 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 1(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 2. Crack inspection elapsed time for different size images

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

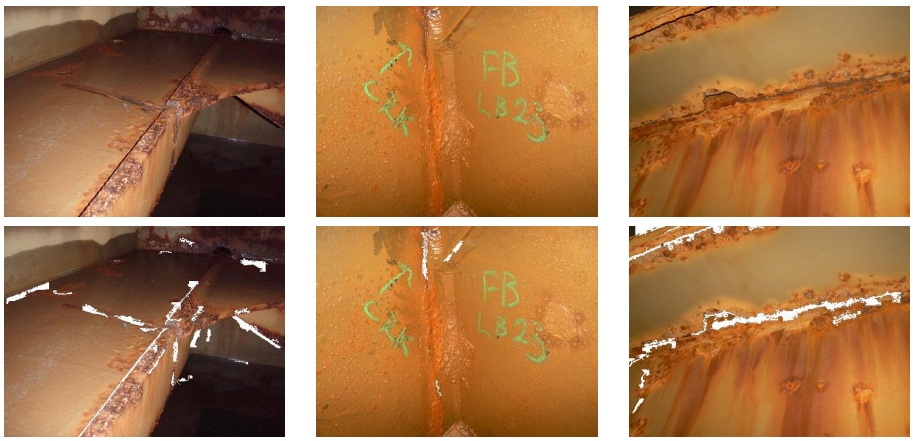

Figure 4. (1st row) Test images and (2nd row) detected cracks

Table 2 shows the elapsed time during cracks detection 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 the elapsed time depends more on the number and size of cracks, and edges in general, that are detected.

Some images and the resulting crack detection are shown in Figure 4. It can be observed that all the cracks that can be detected by means of visual inspection are detected by the crack detection algorithm as well.

4.2. Detecting corrosion

Regarding the corrosion detector, the patch size was 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.

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

0.5	0.2	0.05
175	99	46

Figure 5. (1st row) Corroded areas detected, (2nd row) τ_E and (3rd row) processing time (ms)

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 5 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.

As well as for the crack detector, quantitative performance results have been obtained by defining first a 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 / #patches*) and false negative percentages (*FN / #patches*) for the different strategies are shown in Figure 6. These misclassification percentages come from the analysis of 7384 patches from 11 different images. As can be observed, 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, it decreases dramatically, so that patches really affected by corrosion are always detected and only uncertain cases are left unidentified.

Similarly to what happened with the cracks detector, the processing time for the corrosion classifier does not depend on the image size but 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 3.

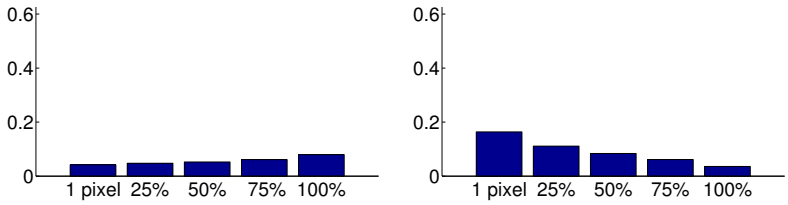

Figure 6. False positive (left) and false negative (right) percentages for different ground truth patch labelling strategies

Table 3. Corrosion inspection elapsed time for different size images

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

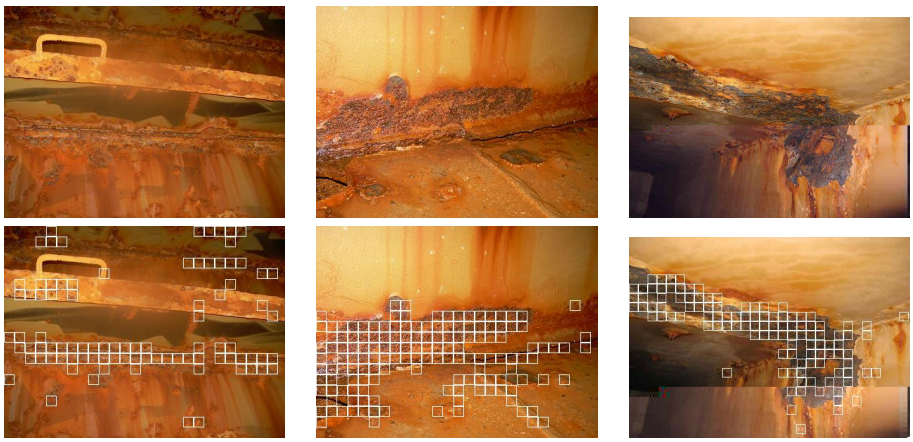

Figure 7. (1st row) Test images and (2nd row) detected corroded areas

Figure 7 shows detection results for some images. In those cases, the algorithm was configured to detect only the image areas most significantly affected by corrosion.

5. Conclusions and future work

Two algorithms for support in vessel hull inspection have been presented in this paper. One the one hand, a crack detection method based on a percolation idea has been introduced and its performance has been proved in the sense of being able to detect all the cracks in the test images. Nevertheless, better results are expected by controlling the imaging process. If the pose and distance of the camera respect to the inspected surface could be controlled, the algorithm parameters could be tuned for cracks of a specific size.

On the other hand, a corrosion detection algorithm based on a supervised classification method has also been presented. The results obtained with colour classification are significantly improved with the addition of a previous decision stage based on a patch energy criterion. Better results are also expected if the information about the correlation

between colour channels was preserved. Some kind of simplified colour histogram could be a good approach.

As future work, it is proposed to use the results of the corrosion detection algorithm to improve the performance of cracks detection, since most of the of cracks observed in the images appear in corroded areas. Then, a previous detection of those areas could be useful to indicate where cracks could be situated and thus, reduce the area to inspect.

Acknowledgements

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.

References

[1] 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-II of *Lecture Notes in Computer Science*, pages 790–801, 2006.

[2] 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*, volume III, pages 239–242, 2004.

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

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

[5] 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*, volume III, pages 353–356, 2002.

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

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

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

[9] 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.

[10] D. Stauffer. *Introduction to Percolation Theory*. CRC press, 2nd edition edition, 1994.

[11] 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.

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

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

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

[15] 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. (available online 11 February 2009).

[16] 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.

[17] 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.