A Saliency-boosted Corrosion Detector for the Visual Inspection of Vessels

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

Abstract. Corrosion is one of the major causes of the structural defects that affect vessel hulls. For its early detection, intensive inspections of the inner and outer structures of the vessel hull are carried out at a great cost, where the visual assessment plays an important role. In order to reduce the cost of the visual inspections, we present a corrosion detector to identify defective areas in digital images taken from vessel hulls. Two main contributions stand out: on the one hand, a specific detector which combines color and texture features to describe corrosion; on the other hand, a prior stage which implements a generic-defect search based on the concept of saliency, and it is used to boost the specific corrosion detector. Both the original and the saliency-boosted methods provide successful detection rates, but the guidance by means of saliency allows for precision improvements.

Keywords. Corrosion detection, Vessel inspection, Saliency

1. Introduction

The different steel surfaces that are part of a vessel hull can be affected by different kinds of defective situations, such as coating breakdown, corrosion, and, ultimately, cracks. These defects are indicators of the state of the metallic surface and, as such, an early detection prevents the structure from buckling or fracturing, and the subsequent ultimate consequences this can give rise to — at several levels, personal, environmental and financial. To avoid reaching such undesirable situations, inspections on-board sea-going vessels are regular activities being initiated partly due to applicable classification and statutory regulations, and partly because of the obvious interest of ship operators and ship owners in anticipating the defective situations, given the costs associated to unexpected disruptions of vessel service availability. In line with the aforementioned, and in order to enhance ships safety and avoid unexpected service disruptions, the INCASS project pursues the integration in a Decision Support System of inspection data collection approaches using robots and specific sensors, together with a risk evaluation, analysis and management framework comprising ship structures, machinery and equipment.

Among the different software and hardware components developed within the INCASS project, this paper presents a vision-based corrosion detector to identify defective areas in digital images taken from vessel hulls. This method is intended to be used as

[1] Corresponding Author: Ed. Anselm Turmeda - Campus UIB, Ctra. Valldemossa km. 7.5, 07122, Palma de Mallorca, Spain; E-mail: xisco.bonnin@uib.es.

an assistant during the visual inspection of vessel structures, in order to reduce cost and duration. It is based on colour and texture descriptors which are combined to describe the appearance of corrosion. Furthermore, this paper proposes an additional improvement to increase the detection performance of the presented corrosion detector. It consists in the use of the generic defect detector presented in [1] as a prior stage, in order to filter out those areas which seem not to be affected by any defective situation. This generic defect detector is based on the idea of saliency, and has proved to provide successful detection results with images including defective areas.

The computer vision literature contains a number of approaches for corrosion detection. Some of them make use of simple methods, such as computing the histogram of the red channel [4], or using a threshold in the HSV colour space [6], while other approaches apply more advanced techniques, such as Support Vector Machines [9,8] or Convolutional Neural Networks [5]. All of them are based on the use of colour and/or texture descriptors, as we also propose for our approach.

The rest of the paper is organized as follows: Section 2 presents our approach for corrosion detection, introducing the colour and texture features that are employed (2.1 and 2.2), detailing how these are combined to build up the detector (2.3), and providing the experimental evaluation to check its performance (2.4); Section 3 introduces the idea of using a generic defect detector to boost the performance of the corrosion detector, and evaluates the improvement; to finish, conclusions are summarized in Section 4.

2. Detection of Corrosion in Vessel Structures

The corrosion detector has been built around a cascade classification scheme, where its different stages can be considered as *weak classifiers*. The idea is to chain different fast classifiers with poor performance in order to obtain a global classifier attaining a much better global performance. 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 reduced false negative rate.

As mentioned before, two features are considered to describe corrosion: texture and colour. Therefore, the corrosion detector comprises two stages, one for each feature. One stage is based on the premise that corroded areas present a rough texture, while the other stage checks whether the inspected area presents a colour typically observed in corroded surfaces. This stage, thus, is based on machine learning and entails a previous training process to learn the colour distribution of corrosion.

2.1. Model of Corrosion Colour: Local Stacked Histograms

The corrosion colour model that we propose consists in learning the colour distribution through the neighbourhoods of corroded pixels. Given a corroded pixel, the histogram for each colour channel is computed for its $N \times N$ pixels neighbourhood. The resulting histograms are stacked together to build what we call a *codeword*.

Two different colour spaces have been considered. On the one hand, RGB is the most common colour space as for sensing, representation and display of images in electronic systems. On the other hand, the HSV colour model separates the colour and intensity

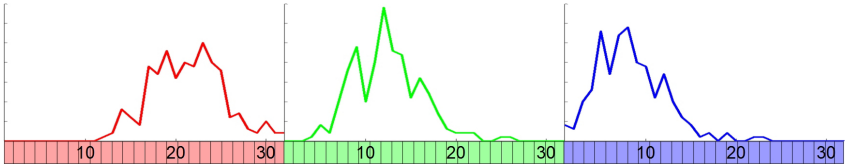

Figure 1. Codeword including colour information used to describe corrosion.

Algorithm 1 Procedure for RGB codewords dictionary generation.

```
1:  procedure CODEWORDS_DICTIONARY_GENERATOR(I, GT, N, K)
2:      I: input image training set
3:      GT: image set comprising a ground-truth image for each image in I
4:      N: patch size in pixels
5:      K: Number of codewords to generate
6:      for all image img in I do
7:          for all pixel p of img do
8:              if p is labelled as corrosion in the ground truth image then
9:                  Get the N×N patch Π centred at p
10:                 Compute the 32-bin histogram for the red channel of Π
11:                 Compute the 32-bin histogram for the green channel of Π
12:                 Compute the 32-bin histogram for the blue channel of Π
13:                 Stack the three histograms together
14:                 Save the resulting codeword
15:             end if
16:         end for
17:     end for
18:     Cluster codewords to K models using K-means
19:     Save the models
20: end procedure
```

information. Indeed, the intensity information is not used in our approach, since we want to learn the colour of corrosion disregarding the amount of light that illuminates the surface.

Therefore, each codeword results from stacking three (R-G-B) or two (H-S) histograms. These histograms are downsampled from 256 to 32 levels in order to reduce its dimensionality and sensitivity to noise. Accordingly, each codeword comprises 96 values, if the RGB colour space is used, and 64, if HS is employed.

By way of example, Fig. 1 shows the codeword corresponding to the neighbourhood of a corroded pixel for the RGB colour space. 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.

The learning stage consists in computing the codewords corresponding to the corroded pixels in the image dataset and clustering them by means of the well-known K-means algorithm [7]. The clustering process is performed in order to make the dictionary more compact and general. The resulting K codewords represent the information learned about the colour distribution around corroded pixels. Algorithm 1 describes the computation of the codeword dictionary for the RGB colour space.

2.2. Model of Corrosion Texture: GLCM Energy

To describe the texture of corrosion we propose using the symmetric Gray-Level Co-occurrence Matrix (GLCM). This matrix is defined over an image to be the distribution of co-occurring pixel values (gray-scale values, or colours) at a given offset. That is, every (i, j) value of the GLCM indicates the number of times in the input image that i and j pixel values occur at the given offset.

To evaluate the roughness of the image, we consider the *energy*, also known as Angular Second Moment (ASM), of the GLCM [7]. The energy of a texture is related to its roughness, so that the higher is the roughness the lower is the energy. The latter is computed by means of

$$E = \sum_i \sum_j p(i, j)^2, \tag{1}$$

where $p(i, j)$ is the probability of the occurrence of values i and j at the chosen offset.

In our approach, the GLCM is calculated for downsampled intensity values between 0 and 31, for a given distance d pixels and direction $\alpha \in k\pi/4$ rad, where $k \in [0, 7]$, so that eight directions are considered in order to compute the isotropic energy.

2.3. The Corrosion Detection Approach

To combine the colour and texture-based stages, we have considered their capability for discarding non-defective areas, together with their computational cost. Taking these criteria into account, the texture stage is executed in first place in order to reduce the number of queries in the codewords dictionary.

Regarding the texture stage, this classifies every pixel as corroded or not depending on whether the energy of the GLCM computed for the surrounding $N \times N$ patch is below a given threshold τ_E.

The colour stage consists in building the codeword for the $N \times N$ image patch centred in the current pixel and comparing with the models of the dictionary by means of the Bhattacharyya distance

$$D_B = -\log\left(\sum_{x \in \mathscr{X}} \sqrt{p_c(x) p_m(x)}\right), \tag{2}$$

where $\mathscr{X}$ refers to the histograms domain and p_c and p_m are histograms from, respectively, the patch codeword and the model from the dictionary.

A pixel is labelled as corroded as soon as a model is found in the dictionary such that $D_B < \tau_D$. Therefore, the approach does not intend to determine which is the closest model, but whether the patch is close enough to any model of corrosion, with the consequently important reduction in the computation time.

The flowchart for the complete algorithm using this colour model is shown in Fig. 2 and its pseudocode, for the case of using the RGB colour space, can be found as Alg. 2.

2.4. Performance Evaluation

In this study, we have used a dataset comprising images of vessel structures including different kinds of corrosion. The images have been collected at different distances

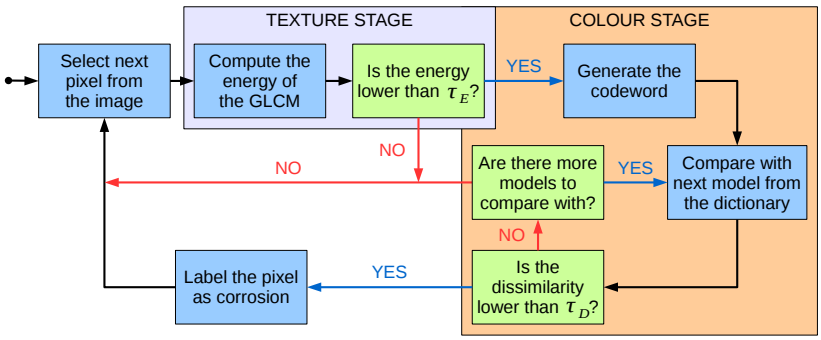

Figure 2. Flowchart of the corrosion detector.

Algorithm 2 Corrosion detector using RGB codewords.

procedure CORROSION_DETECTOR(img, N, τ_E, τ_D)
    img: input colour image
    N: patch size in pixels
    τ_E, τ_D: energy and dissimilarity thresholds
    Load codewords dictionary
    Initialize img_out to *no-defect* ▷ No pixel is classified as *corrosion*
    for all pixel p in img **do**
        Get the $N \times N$ patch Π centred at p
        Compute the energy e of Π
        if $e < \tau_E$ **then** ▷ Proceed with the colour stage
            Compute the 32-bin histogram for the red channel of Π
            Compute the 32-bin histogram for the green channel of Π
            Compute the 32-bin histogram for the blue channel of Π
            Stack the three histograms together into a codeword
            while there are more models *mod* in the dictionary **and**
                    p has not been labelled in img_out **do**
                Calculate the Bhattacharyya distance D to the model *mod*
                if $D < \tau_D$ **then** ▷ The codeword is similar to the model
                    Label p as *defect* in img_out ▷ p is classified as *corrosion*
                end if
            end while
        end if
    end for
    return img_out
end procedure

and under different lighting conditions. This dataset also includes the ground truth consisting in binary images where defects are labelled in white and it is available online (http://dmi.uib.es/ xbonnin/resources).

To evaluate the detection performance of the corrosion detector, different values of the parameters have been tested trying to find the best configuration. Regarding the texture stage, the GLCM has been computed using a distance $d = 5$ pixels and an image

Table 1. AUC values for the corrosion detector. The best results are indicated in blue.

			Threshold τ_E		
			0.2	**0.4**	**0.6**
Codewords dictionary	**100 models**	**RGB**	0.877	0.892	0.891
		HS	0.875	0.879	0.875
	300 models	**RGB**	0.867	0.880	0.883
		HS	0.860	0.858	0.850
	500 models	**RGB**	0.863	0.871	0.870
		HS	0.859	0.852	0.842

patch of 15×15 pixels, while the energy threshold τ_E has been set to different values covering the full [0, 1] range.

Concerning the colour stage, its parameters can be configured regardless the selected colour space. Local histograms are computed for 15×15 pixels patches, and, to create the codewords dictionary, three different sizes have been evaluated: 100, 300 and 500 codeword models. Notice that, when using a larger dictionary, more comparisons are performed prior to discarding non-defective pixels, what means a longer processing time. The dictionaries are created following the Leave-One-Out Cross-Validation (LOOCV) methodology [2], so that the image which is being inspected has not been used to create the corresponding dictionary during the training stage. Finally, the dissimilarity threshold τ_D has been set to different values in the [0, 1] range.

Several metrics are used to perform a quantitative evaluation of the performance of the detector. In this regard, Fig. 3[top] shows the ROC curves (True Positive Rate vs. False Positive Rate) obtained for all the images of the dataset, while Fig. 3[bottom] provides the Precision-Recall (PR) curves [3]. Each curve corresponds to the use of a specific dictionary size (100, 300 and 500) and energy threshold τ_E (0.2, 0.4 and 0.6), and has been generated by variation of the codeword dissimilarity threshold τ_D. To facilitate the comparison of the ROC curves, Table 1 provides the corresponding values for the Area Under the Curve (AUC) [3].

As can be observed, all the combinations show a similar performance, with ROC curves close to the (0, 1) corner (which corresponds to the perfect classifier), and AUC values between 0.84 and 0.89 (close to 1, which is the maximum value). The best results are obtained when using the smaller dictionary, which contains just 100 codewords. That means that the use of larger dictionaries, with 300, 500 or even more codewords, leads to overfitting and loss of generalization.

Regarding the colour space, the use of RGB codewords allows achieving a higher precision at low recall values, while, for higher recall values, HS and RGB codewords lead to a similar precision.

Table 2 provides the execution times required by the corrosion detector. These values correspond to the mean execution times per pixel using the parameter configuration that provides the best performance, based on the ROC curves presented in Fig. 3 [top]. The table also provides the expected execution times required if a 1024×768 image was processed. Notice that these are just estimations computed averaging the observed values. Actually, the processing time depends on the percentage of corroded area in the specific image considered. As can be observed, the use of RGB codewords entails an increment of 40% in the execution time, with regard to the use of HS codewords.

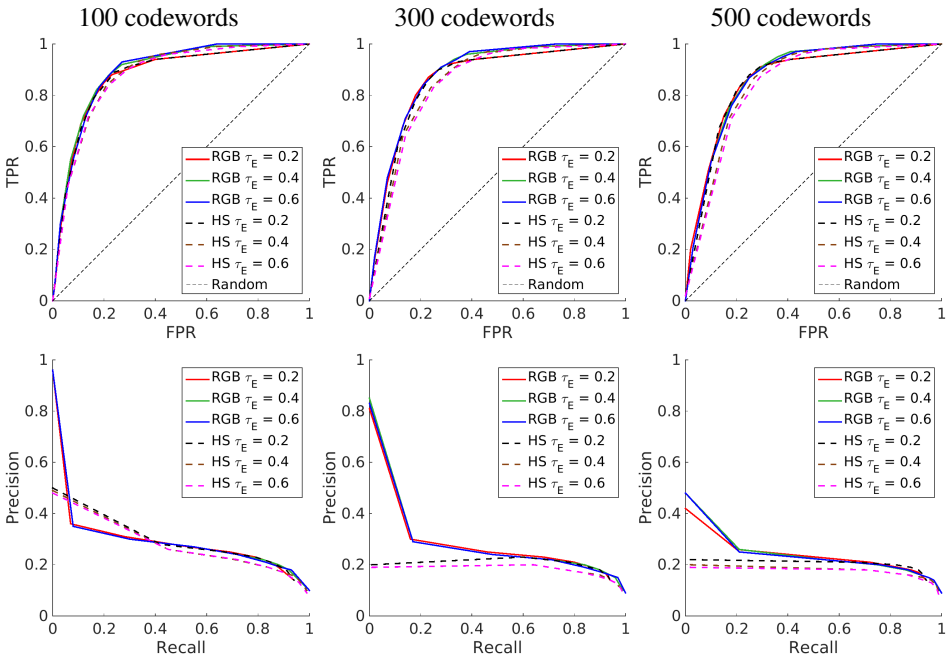

Figure 3. Performance of the corrosion detector: (top) ROC curves and (bottom) PR curves. Each curve corresponds to the use of a different combination colour space/energy threshold, and has been generated by variation of the threshold τ_D. Each column corresponds to a different dictionary size.

Table 2. Execution times of the corrosion detector using the different colour models. Mean values computed using the parameter configurations that provide the best performance.

	RGB	**HS**
μs / pixel	31.12	22.26
seconds / 1024×768 image	24.47	17.50

Figure 4 shows some results of the corrosion detector using the different colour spaces. As can be observed, both configurations are able to successfully detect the corroded areas in the input images, and their results approximately match the ground truth.

3. Saliency-boosted Corrosion Inspection

In this section we combine the image saliency-based generic defect detector proposed in [1] to boost the corrosion detector described in Section 2. That is to say, the idea is to take advantage of the high performance of the generic defect detector to discard non-defective image areas, and then use the specific detector to select, within the salient areas, those pixels affected by corrosion.

Among the different configurations of the saliency-based defect detector described in [1], we have selected the *OR* configuration. This version combines image contrast-based saliency and image symmetry-based saliency into a single defect map. As shown in [1], this setup outperforms the configurations which are based only on one of these two saliency-related features.

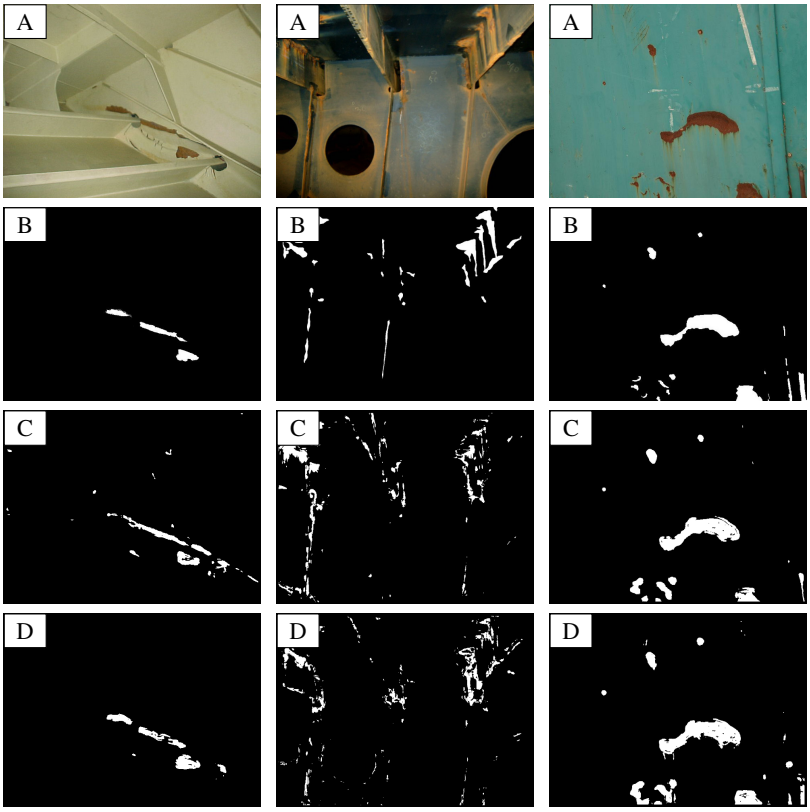

Figure 4. Corrosion detection results for some images of the dataset: (A) input image, (B) ground truth, (C-D) corrosion detection outputs using, respectively, RGB and HS codewords.

The saliency-based defect detector has been configured to improve the performance of the corrosion detector. To be precise, the saliency threshold τ_S, which is used to filter the potentially defective areas, has been set to provide a very high True Positive Rate (TPR), despite this means to move a bit further from the (0, 1) corner in the ROC space, increasing the False Positive Rate (FPR). The idea is to ensure that all the corroded pixels are not discarded by the first stage, but are input of the corrosion detector, i.e. to keep the false negatives close to zero.

Figure 5 [left] shows the ROC curve corresponding to the corrosion detector (using the configuration which provides the best results) together with the point corresponding to the saliency-based generic defect detector, once the threshold τ_S has been configured as indicated before. As can be observed, this point is situated in the coordinates (0.31, 0.97) of the ROC space, which is above the curve provided by the corrosion detector.

Notice that, if the output of the generic defect detector is provided as input to the corrosion detector, the new ROC curve resulting from varying its parameters will end up at point (0.31, 0.97), which corresponds to the corrosion detector that labels as positive all the pixels that have passed the first classifier. Therefore, we can expect that the new ROC curve will pass above the original curve.

Figures 5 [middle] and 5 [right] compare the performance of the original corrosion detector with the new version boosted by the saliency-based defect detector. As expected,

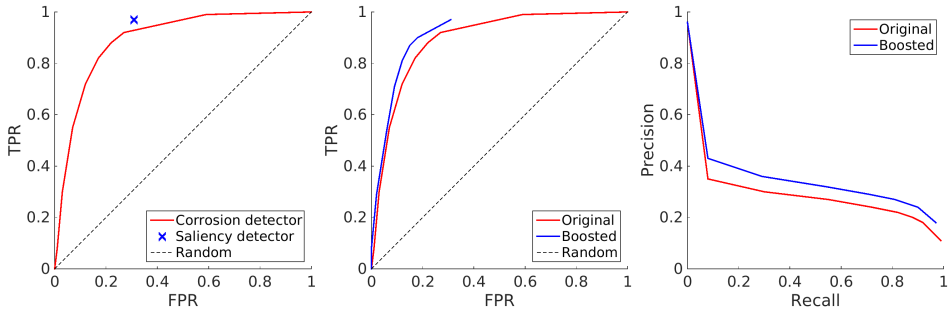

Figure 5. (Left) Performance of the corrosion and the saliency-based generic defect detectors. (Middle and right) Performance of the saliency-boosted corrosion detector against the original method: (middle) ROC curves, (right) PR curves. Performance curves obtained for energy threshold $\tau_E = 0.4$, and by variation of threshold τ_D.

the results show that the boosted version outperforms the original detector: on the one hand, the new ROC curve is closer to the (0, 1) corner; on the other hand, the boosted detector provides higher values of precision, as shown in Fig. 5 [right].

By way of example, Fig. 6 compares the output provided by the original and the boosted versions of the corrosion detector using the same configuration of their parameters ($\tau_E = 0.4$ and $\tau_D = 0.175$). As can be observed, the boosted version leads to less false positives than the original version.

4. Conclusions

This paper has presented a novel corrosion detector to assist during the visual inspection of vessels. This consists in the combination of texture and color features which are useful to describe the appearance of corroded surfaces. Several metrics have been used to evaluate the performance of the method with a dataset including real vessel hull images. The results indicate that the method is able to successfully label the corroded areas. Furthermore, a generic defect detector based on the idea of saliency has been used as a previous stage in order to filter out non-defective areas. In the light of the results obtained using this guidance, we can state that the saliency-based classifier allows boosting the specific defect detector, reducing the false positive detections and, thus, increasing precision.

5. Acknowledgements

This work has been partially supported by EU-FP7 project INCASS (MOVE/FP7/605200 /INCASS), by project number AAEE50/2015 (Govern de les Illes Balears, DGIR) and by FEDER funding.

References

[1] F. Bonnin-Pascual and A. Ortiz. A Generic Framework for Defect Detection on Vessel Structures based on Image Saliency. In *IEEE International Conference on Emerging Technologies and Factory Automation*, 2016.

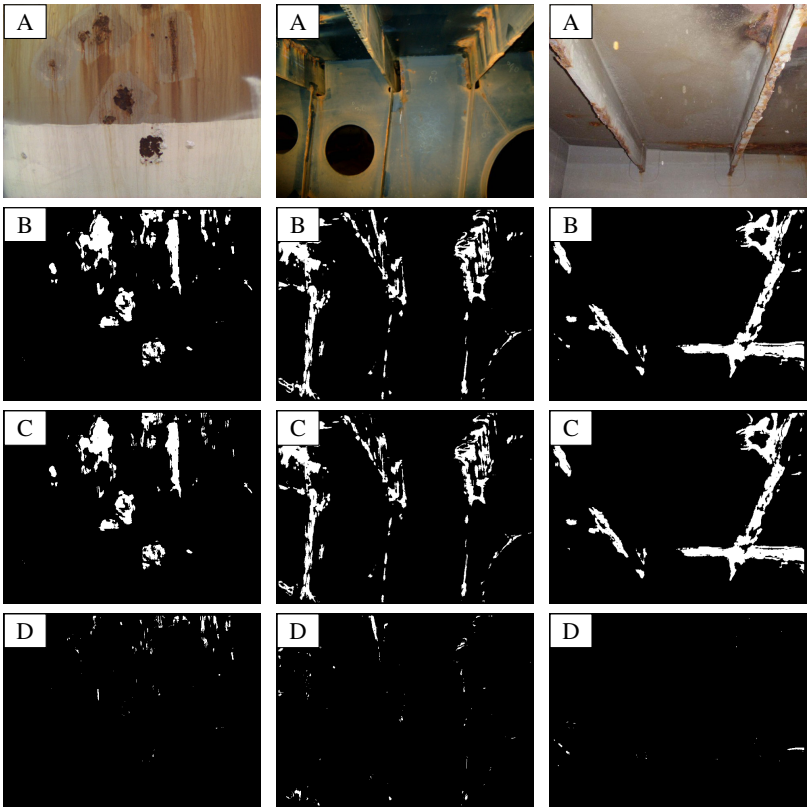

Figure 6. Corrosion detection results for some images: (A) input image, (B) output provided by the original version of the method, (C) output provided by the saliency-boosted version, and (D) difference image showing those areas that are labelled as corrosion by the original version but discarded by the saliency-boosted version.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2nd edition, 2000.

[3] T. Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[4] S. A. Idris, F. A. Jafar, and S. Saffar. Improving Visual Corrosion Inspection Accuracy with Image Enhancement Filters. In *International Conference on Ubiquitous Robots and Ambient Intelligence*, pages 129–132, 2015.

[5] L. Petricca, T. Moss, G. Figueroa, and S. Broen. Corrosion Detection using A.I.: A Comparison of Standard Computer Vision Techniques and Deep Learning Model. In *International Conference on Computer Science, Engineering and Information Technology*, pages 91–99, 2016.

[6] N. S. Roberts. Corrosion Detection in Enclosed Environments using Remote Systems. Master's thesis, Alfred University, 2016.

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

[8] F. Tsutsumi, H. Murata, T. Onoda, O. Oguri, and H. Tanaka. Automatic Corrosion Estimation using Galvanized Steel Images on Power Transmission Towers. In *Transmission and Distribution Conference and Exposition: Asia and Pacific*, 2009.

[9] M. Yamana, H. Murata, T. Onoda, T. Ohashi, and S. Kato. Development of System for Crossarm Reuse Judgment on the Basis of Classification of Rust Images using Support Vector Machine. In *IEEE International Conference on Tools with Artificial Intelligence*, 2005.