Towards automated ship inspection: A visual data-oriented toolbox

A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo & J.P. Company-Corcoles
Dep. of Mathematics and Computer Science, University of Balearic Islands, Palma de Mallorca, Spain

ABSTRACT: The obvious interest on the monitoring and identification of defects affecting the metallic structures of ships, as well as on reducing the costs of the always expensive onboard inspections, suggests the introduction of technological tools to assist the involved agents (ship owners/operators, surveyors). This paper describes a toolbox, developed within the frameworks of the INCASS and MINOAS projects, whose goal is to provide decision support at the defect detection level. These tools must be considered in a broader sense, covering both robotic solutions and software utilities. In brief, the toolbox comprises: (1) a multi-rotor based aerial platform which is used to collect images from the surfaces under inspection, either focusing on providing access to remote areas or for broader and intensified data acquisition; (2) a control architecture running onboard the aerial platform, specifically oriented to simplifying visual inspections, but generic enough to be able to run over any aerial platform fitted with one of several compatible sensor suites; (3) an image stitching software aiming at enhancing the presentation of the visual data collected, as well as improving defect detection since defects no longer appear broken across consecutive images, but as single units in the image composite; (4) a lightweight image-saliency detector tuned for detecting generic defects, what can be useful to either guide the image capture onto relevant areas from the inspection point of view or to guide specific defect detectors; (5) a collection of crack and corrosion defect detectors spanning a range of image processing and machine learning techniques.

KEYWORDS: aerial vehicle; supervised autonomy; image stitching; defect detection; machine learning

1 INTRODUCTION

The different steel surfaces that are part of a vessel's 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 many levels, personal, environmental and financial.

To avoid reaching such undesirable situations, inspections onboard 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.

Unfortunately, continuous monitoring of the structure at a required level of detail, with proper identification of defects and/or assessment of corrosion, is not trivial and quite expensive (taking into account the vessel's preparation, use of yard facilities, cleaning, ventilation and provision of access arrangements). In this regard, since visual inspections are and will be an important source of information for structure condition assessment, it seems necessary to try to reduce the effort and cost related to these activities with the introduction of new technological tools. This paper describes a set of such tools, developed within the frameworks of the INCASS and the MINOAS projects, whose goal as a whole is to provide decision support at the defect detection level. These tools must be considered in a broader sense, covering both robotic solutions and software utilities. In brief, the toolbox comprises: (1) a multi-rotor based aerial platform which is used to collect images from the surfaces under inspection, either focusing on providing access to remote areas or for broader and intensified visual data acquisition; (2) a control architecture running onboard the aerial platform, specifically oriented to simplifying visual inspection operations, but generic enough to be able to run over any aerial platform fitted with one of several compatible navigation sensor suites; (3) an image stitching software aiming at

enhancing the presentation of the visual data collected, as well as improving defect detection since defects no longer appear broken across consecutive images, but as a single unit in the image composite; (4) a lightweight image-saliency detector tuned for detecting generic defects, what can be useful to either guide the image capture onto relevant areas from the inspection point of view or to guide specific defect detectors; (5) a collection of defect detectors covering a range of image processing and machine learning techniques devised for crack and coating breakdown/corrosion detection.

The rest of the paper is organized as follows: Section 2 describes the aerial platform, with particular emphasis on the features it is fitted with thanks to a control architecture specifically developed for inspection applications; Section 3 details the sensor data collection capabilities of the platform; post-processing tools, including the image stitching software and the defect detectors, are addressed in, respectively, Sections 4 and 5; to finish, conclusions are summarized in Section 6.

2 THE AERIAL PLATFORM

2.1 *Platform overview*

As previously said, the aerial vehicle is based on a multi-rotor design. These robotics platforms have become increasingly popular in recent years, and, as a consequence, a number of control and navigation solutions can be found in the related literature. They differ mainly in the sensors used to solve these tasks, the amount of processing that is performed onboard/offboard, and the assumptions made about the environment. Apart from other devices, such as infrared and ultrasound sensors, laser scanners (Bachrach et al. 2011, Dryanovski et al. 2013, Grzonka et al. 2012) and, lately, vision cameras (Achtelik et al. 2012, Chowdhary et al. 2013, Engel et al. 2014, Fraundorfer et al. 2012, Shen et al. 2013, Troiani et al. 2015) have become the preferred sensor modalities to undertake these tasks, mostly within *Simultaneous Localization and Mapping* (SLAM) frameworks and combined with *Inertial Measuring Units* (IMU).

The control software has been configured to be hosted by any of the research platforms developed by *Ascending Technologies* (the quadcopters *Hummingbird* and *Pelican*, and the hexacopter *Firefly*), although it could be adapted to other systems. The *AscTec* vehicles are equipped with one IMU, which comprises a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer, and two ARM7 microcontrollers. Attitude stabilization control loops linked to the onboard IMU and thrust control run over the main ARM7 microcontroller as part of the platform firmware. The manufacturer leaves almost free an additional secondary ARM7 microcontroller which can execute onboard higher-level control loops.

All platforms are fitted with a navigation sensor suite that allows them to estimate the vehicle *state*, which comprises 3-axis speed (v_x, v_y, v_z), the flying height z and the distances to the closest obstacles in different orientations, e.g. left (d_l), right (d_r) and forward (d_f). These estimations can be performed by means of different sensor combinations leading to different weight, volume occupied and power consumption. This permits preparing for the inspection application either vehicles of low payload capacity (lower-cost platform) or vehicles able to lift a heavier sensor suite (higher-cost platform). By way of example, Figure 1 shows a Hummingbird platform, fitted with two lightweight optical-flow sensors for speed estimation, ultrasound sensors for obstacle detection and an infrared height-meter, and a Pelican platform fitted with a laser scanner for speed estimation and obstacle detection, and a laser-based height-meter. Additional details for the latter can be found in Table 1.

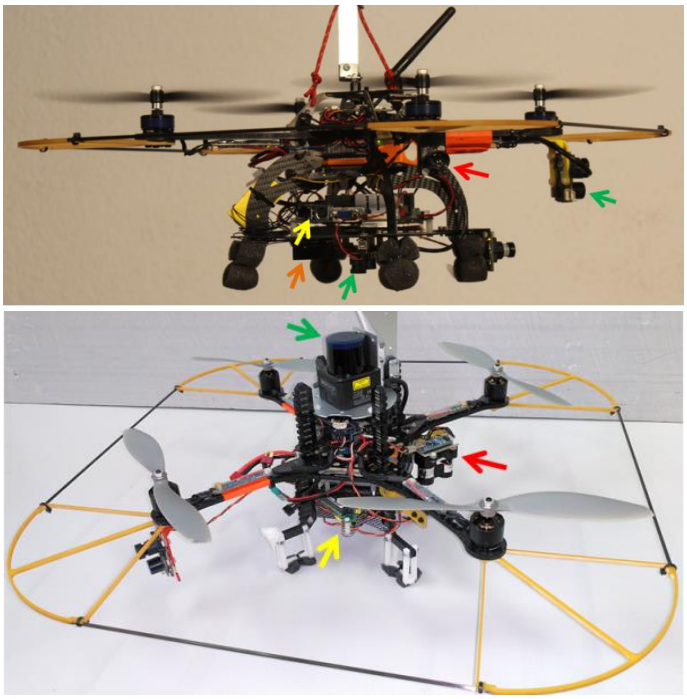

Figure 1. [top] A Hummingbird platform featuring optical flow sensors (green), an infrared height-meter (orange), and ultrasound sensors (red). [bottom] A Pelican platform featuring a laser scanner (green) and a laser-based height-meter (red). The embedded PC is indicated by a yellow arrow in each case.

Besides the navigation sensor suite, all platforms carry, in accordance to their payload capacity, one or several cameras for collecting the expected visual inspection data.

To finish, apart from the two ARM7 microcontrollers integrated in the *flight control unit* of the *AscTec* platforms, all vehicles carry an embedded PC, which avoids sending sensor data to a base station, but process them onboard and, thus, prevent communications latency inside critical control loops. Once again, the different platforms are endowed with boards compatible with their payload limits, e.g. the Hummingbird of Figure 1 features a Commell LP-172 Pico-ITX

board fitted with an Intel Atom 1.86 GHz processor and 4 GB RAM, while the Pelican carries an Intel NUC D54250WYB with an Intel Core i5-4250U 1.3 GHz processor and 4 GB RAM.

Table 1. Specifications of one of the aerial robots.

Aerial platform (Pelican)	
Size (L x W x H)	650 mm × 650 mm × 270 mm
Weight	1700 g
Propulsion	4 × 160 W brushless motors
	10” propellers
Power	11.1V, 4500mAh
	3-cell Lithium-Polymer
Speed	0.5 – 2 m/sec
Sensors	3-axis IMU
	Laser scanner / optical flow sensors
	Height meter
	2 Mpx still camera
	12 Mpx full HD video camera
Communication	Dual 2.4 - 5GHz Wi-Fi LAN
& interaction	Joystick / gamepad
Aux. components	Onboard 10W LED spotlight(s)

2.2 *Control software*

Following the advice received during the MINOAS field trials, the current aerial platforms integrate a control architecture that follows the *supervised autonomy* (SA) paradigm (Cheng & Zelinsky 2001). This is a human-robot framework where the operator is always allowed to be within the general platform control loop though assisted by the robot, which implements a number of autonomous functions, including self-preservation and other safety-related issues, which make simpler the intended operations for the operator and so permits he/she to focus in accomplishing the task at hand. Within this framework, the communication between the robot and the user is performed via qualitative instructions and explanations: the user prescribes high-level instructions to the platform while this provides instructive feedback. In our case, we use simple devices such as a joystick or a gamepad to introduce the qualitative commands and a *graphical user interface* (GUI) to receive the robot feedback. Joystick commands and the GUI are handled at a *base station* (BS) linked with the MAV via a Wi-Fi connection.

The control software is organized around a layered structure distributed among the available computational resources. On the one hand, the *low-level control* layer implementing attitude stabilization and direct motor control executes over the main microcontroller as the platform firmware provided by the manufacturer (Gurdan et al. 2007). On the other hand, *mid-level control*, running over the secondary microcontroller, comprises height and velocity controllers which map input speed commands into roll, pitch, yaw and thrust orders. Lastly, the *high-level control* layer, which executes over the embedded PC, implements a reactive control strategy coded as a series of ROS nodes (http://www.ros.org, the Robot Operating System) running over Linux Ubuntu, which combine the user desired speed command with the available sensor data $-v_x$, v_y, and v_z velocities, height z and distances to the closest obstacles d_l, d_r and d_f –, to obtain a final and safe speed set-point that is sent to the speed controllers.

Speed commands are generated through a set of robot behaviours organized into a hybrid competitive-cooperative framework (Arkin 1998). That is to say, on the one hand, higher priority behaviours can overwrite the output of lower priority behaviours by means of a suppression mechanism taken from the *subsumption* architectural model. On the other hand, the cooperation between behaviours with the same priority level is performed through a *motor schema*, where all the involved behaviours supply each a motion vector and the final output is their weighted summation. An additional flow control mechanism selects, according to a specific input, among the outputs provided by two or more behaviours.

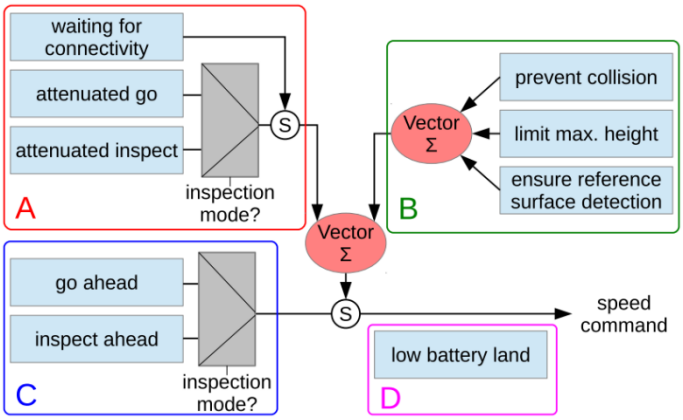

Figure 2. Behaviour-based upper control layer.

Figure 2 details the behaviour-based architecture, grouping the different behaviours depending on its purpose. A total of four general categories have been identified for the particular case of visual inspection: (a) *behaviours to accomplish the user intention*, which propagate the user desired speed command, attenuating it towards zero in the presence of close obstacles, or keeps hovering until the Wi-Fi link is restored after an interruption; (b) *behaviours to ensure the platform safety within the environment*, which prevent the robot from colliding or getting off the safe area of operation; (c) *behaviours to increase the autonomy level*, which provide the platform with higher levels of autonomy to both simplify the operation and to introduce further assistance during inspections; and (d) *behaviours to check flight viability*, which checks whether the flight can start or progress at a certain moment in time. Some of the behaviours in groups (a) and (c) can operate in the so-called *inspection mode*. While in this mode, the vehicle moves at a constant

and reduced speed (if it is not hovering) and user commands for longitudinal displacements or turning around the vertical axis are ignored. In this way, during an inspection, the platform keeps at constant distance/orientation with regard to the front wall, for improved image capture.

3 DATA COLLECTION CAPABILITIES

During flight, any of the aerial platforms can collect pictures on demand or at a fixed rate, e.g. 10 fps, as well as log flight data. The latter includes the vehicle pose, i.e. 3D position and 3D attitude, the vehicle speeds and the distances to the closest obstacles. Of particular relevance is the vehicle pose, which permits associating a 3D position to the defects found. For this purpose, two *simultaneous and localization methods* (SLAM) have been integrated onboard the aerial platforms given their different payload capacities. One adopts a laser-based SLAM strategy while the other is a visual single-camera SLAM solution: while the first one aligns consecutive laser scans to estimate the vehicle motion from one time instant to the next, the second solution matches image features across consecutive images, projects them in 3D space and determines the corresponding 3D transformation. Depending on the robot onboard computational capabilities, the latter process can run on-line or off-line, after flight. By way of example, Figure 3 (top) shows the paths estimated by the laser-based approach for two flights, as well as illustrates the defect localization process after visual inspection through the projection, as different coloured rectangles, of the bounding boxes of the defects found during a flight (bottom). Figure 4 shows the aerial robot during visual inspections onboard an oil tanker (top) and onboard a bulk carrier (middle); examples of the images captured for both cases can be found, respectively, at the bottom left and right.

4 IMAGE MOSAICING

Typically, a large number of images are available after an inspection operation. Instead of just archiving them, and perhaps processing them later, this section describes a tool to improve their presentation and avoid at the same time the redundancy that naturally results when the images are taken at a constant rate, so that it is usually the case that a number of consecutive images contain essentially the same information. The idea is to determine the overlap between the different images collected and use them and the detected overlap to build one single, seamless composite image from the area under inspection.

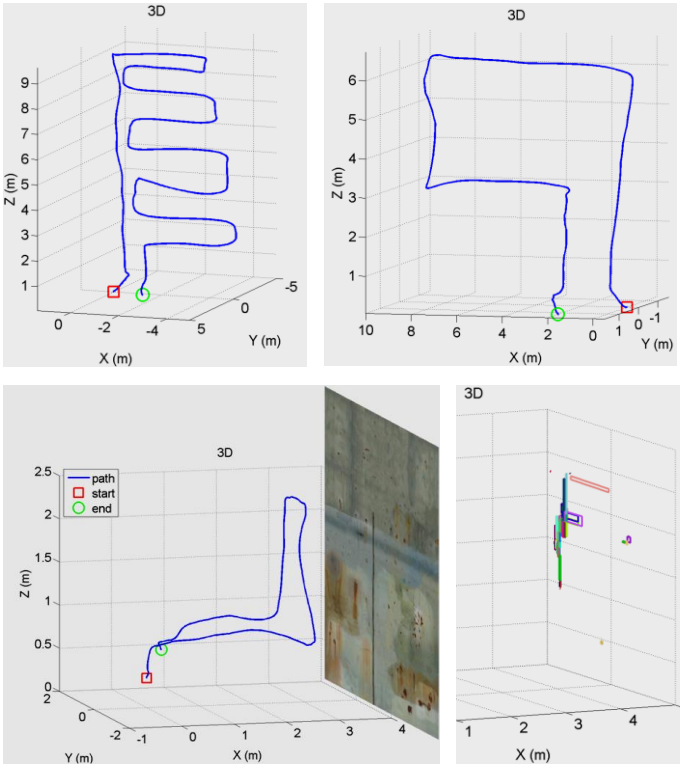

Figure 3. Paths estimated after a flight [top] and illustration of the defect localization process after visual inspection [bottom].

Figure 4. Visual inspections onboard an oil tanker [top and bottom left] and onboard a bulk carrier [middle and bottom right].

This process, generically known as *image mosaicing*, can provide a number of additional benefits to the inspection process: (1) defects, which, depending on the camera optics, distance and viewpoint, can appear

broken in all images they are contained in, in the mosaic they will likelier appear in its full extension, so that their severity will be better appreciated; (2) since the different images are transformed to a common frame, the mosaic compensates for differences in camera viewpoint and distance among the individual images; and (3) this common frame makes it possible to take measurements from the images containing the defects if needed. Besides, it is not necessary to build a mosaic from all the images collected (which can take some time), but from a selected set of images, e.g. to get an image containing a single, large defect.

In this section, we briefly describe BIMOS (*Binary descriptor-based Image MOSaicing*), an image stitching approach which can produce seamless mosaics on different scenarios and camera configurations in a reasonable amount of time thanks to the multi-threaded architecture. In more detail, after decoupling the strategic steps involved in the mosaicing process, and as outlined in Figure 5, BIMOS consists of four threads that run in parallel and contribute all to the data structure that finally gives rise to the image composite, called the *mosaic graph*.

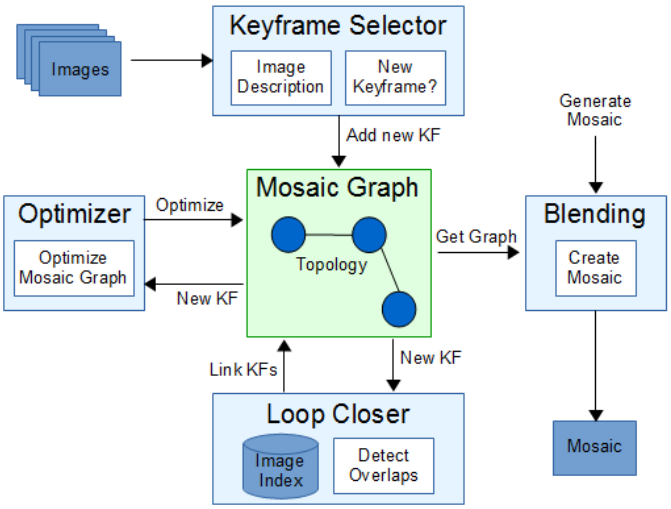

Figure 5. BIMOS architecture: the four threads (in blue) interact with a shared structure called *mosaic graph* (in green).

For a start, BIMOS makes use of ORB features (Rublee et al. 2011) to describe images, what accelerates the image description process thanks to the binary nature of ORB. Besides, instead of using all the input images to build the mosaic, we adopt a *keyframe* selection policy, by which only images providing a significant contribution to the image composite (keyframes) are employed, and the rest are discarded, what in turn avoids unnecessary drift during the image alignment process. This contribution is measured as the amount of overlap between the current image and the last keyframe inserted in the mosaic graph.

To find overlapping images (i.e. close a visual loop), we employ a binary visual dictionary (Garcia-Fidalgo & Ortiz 2014), which is based on a *Bag of Words* (BoW) scheme that is built in an online manner. This avoids a training stage prior to building mosaics as well as the dependence on a specific dictionary.

The optimizing thread finds the camera motion between keyframes by optimizing the alignment error. This motion is modelled through a similarity transformation H:

$$H = \begin{pmatrix} \lambda\cos\theta & -\lambda\sin\theta & t_x \\ \lambda\sin\theta & \lambda\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}, \quad (1)$$

where λ is a scale factor, (t_x, t_y) is a translation vector and θ is a rotation angle.

Finally, the blending thread produces the final mosaic under demand, making use of the camera motion estimated by the optimizer up to the moment in which the mosaic generation command is issued. The multi-band blending algorithm by Burt & Adelson (1983) is employed to diminish the visual artifacts that result from the combination of the images contributing to the mosaic.

To finish with this section, Figure 6 shows two mosaics produced by BIMOS.

Figure 6. Mosaics from sequences resulting from an inspection operation (left) and a flight at high altitude (right).

5 DEFECT DETECTION

Along projects MINOAS and INCASS, we have addressed automatic defect detection in images from a number of different technologies, ranging from image processing to machine learning. Due to lack of space, in this section we will be able to cover only a selection

of the different defect detectors which we have developed. In more detail, we will describe first an innovative approach based on image saliency, which results into an effective and generic detector of defects in vessel surfaces. In the next section, we will describe specific detectors for coating breakdown/corrosion and cracks, which actually could be guided by the saliency-based detector.

5.1 *Image saliency-based defect detection*

This approach considers defects as rare phenomena that may appear on a regular surface or structure. Since they are rare, the probability that an area is affected by a defect should be rather low. As described below, this low probability can be used as an indicator of image saliency, and thus to highlight defective areas in digital images.

5.1.1 *Bayesian approach for saliency computation*

Similarly to Zhang et al. (2008), we make use of a Bayesian approach to compute the saliency map Σ_{ij}:

$$\Sigma_{ij} = \frac{1}{p(F=f_{ij})} p\left(F = f_{ij} \middle| T = \delta\right), \quad (2)$$

where f_{ij} is the value of the feature *F* found at an image location *(i, j)*, and *T* stands for the target class, i.e. the defect class δ in our case. Hence, equation (2) combines top-down information with bottom-up saliency to find the pointwise mutual information between the feature and the target. Using this formulation, the saliency at a given image point decreases as the probability of feature value f_{ij} gets higher, and increases as the probability of feature value f_{ij} for the defect class δ increases.

5.1.2 *Image contrast-based saliency*

As said before, we consider defects as rare phenomena that catch the visual attention of the observer during visual inspection. Following this idea, we describe defects by means of features typically used in cognitive models to predict human eye fixations. To this end, we make use of one of the most influential saliency computational models based on contrast, described in Itti et al. (1998). In this model, the contrast levels in intensity, colour and orientation are computed as centre-surround differences between fine and coarse scales over image pyramids of up to 7 levels; that is to say, the difference between each pixel on a fine (or centre) scale *c* and its corresponding pixel in a coarse (or surrounding) scale *s* is calculated as $M(c,s) = |M(c) \otimes M(s)|$, where $\otimes$ is the centre-surround operator, $c \in \{1, 2, 3\}$ and $s = c + \lambda$, with $\lambda \in \{3, 4\}$. Given an RGB colour image, this process is performed over: ($\oplus$ denotes across-scale addition)

- the intensity channel $I = (r + g + b)/3$, with *r*, *g* and *b* as the original red, green and blue channels, to build the *intensity conspicuity map* $IM = \oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} N(I(c,s))$;
- the colour channels RG and BY defined as $RG = R - G$ and $BY = B - Y$, with $R = r - (g + b)/2$ for red, $G = g - (r + b)/2$ for green, $B = b - (r + g)/2$ for blue and $Y = (r + g)/2 - |r - g|/2 - b$ for yellow (negative values are set to zero for all channels), to build the *colour conspicuity map* $CM = \oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} N(RG(c,s)) + N(BY(c,s))$;
- the orientation channels $O(\theta)$, calculated by convolution between channel *I* and Gabor filters at orientations 0º, 45º, 90º and 135º, to build the *orientation conspicuity map* $OM = \sum_{\theta \in \{0º, 45º, 90º, 135º\}} N\left(\oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} N(O(c,s,\theta)) \right)$.

The map normalization operator $N(*)$ highlights saliency peaks in maps where a small number of strong peaks of activity (conspicuous locations) are present, while globally suppressing peaks when numerous comparable peak responses are present. To this end: (1) the map is normalized to a fixed range, (2) the global maximum *M* is found, (3) the local maxima average *m* is determined, and (4) the map is multiplied by $(M - m)^2$.

Finally, the three conspicuity maps are normalized and summed into the final contrast-based defect map:

$$con_{ij} = \frac{1}{3}\left(N(IM_{ij}) + N(CM_{ij}) + N(OM_{ij})\right) \quad (3)$$

5.1.3 *Illustrative results*

Figure 7 shows defects maps for a number of images containing defects. These results have been obtained using *probability density functions* (PDFs) for contrast, i.e. $p(F = con_{ij})$, and contrast conditioned on the presence of defects, i.e. $p(F = con_{ij}|T = \delta)$, both estimated by means of the *Parzen windows* method (Theodoridis & Koutroumbas 2008), and an image set comprising cracks, coating breakdown and corrosion.

5.2 *Specific defect detection*

5.2.1 *Coating breakdown/Corrosion detection*

The coating breakdown/corrosion (CBC) detector described in this section is based on a supervised classification scheme that comprises two steps which can be considered as two weak classifiers, reason why it is named WCCD (*Weak-classifier Colour-based Corrosion Detector*). The idea is to chain different poor-performance fast classifiers to obtain a global classifier attaining a higher global performance (Duda et al. 2000).

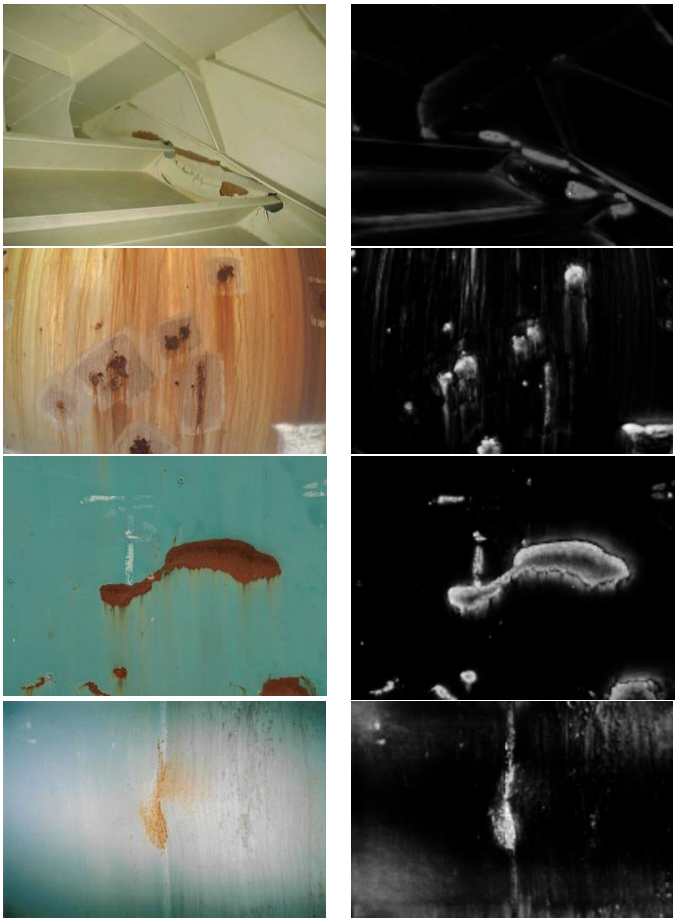

Figure 7. Saliency-based defect maps for real images containing defects. (*Whiter* means *more salient*.).

The first stage of this classifier is based on the premise that a corroded area exhibits a rough texture, where roughness is measured as the energy of the symmetric *gray-level co-occurrence matrix* (GLCM), calculated for down-sampled intensity values between 0 and 31 (Theodoridis & Koutroumbas 2008). The energy of an image patch is then obtained by means of

$$E = \sum_{i=0}^{31} \sum_{j=0}^{31} p(i,j)^2 \,, \qquad (4)$$

where $p(i, j)$ is the probability of the co-occurrence of gray levels i and j. Low-energy patches, i.e. exhibit a rough texture, are candidates to be more deeply inspected.

The second classifier operates over the pixels of the patches that have survived during the roughness step. This classifier makes use of the colour information that can be observed from the corroded areas, unlike the first classifier. It works over the Hue-Saturation-Value (HSV) colour space after the realization that HSV values that can be found in corroded areas are confined to a bounded subspace of the HS plane. This second step requires a prior training step to learn the colour of the corroded areas, by building a bi-dimensional histogram of HS values for image pixels labelled as corroded.

5.2.2 *Corrosion-guided crack detection*

After the observation that most cracks in metallic surfaces coincide, at least partly, with corroded areas, in this section we describe a crack detector guided by the output of WCCD. This crack detector, named GPCD (*Guided Percolation-based Crack Detector*), proceeds in accordance to a percolation model, similarly to the detector described in Yamaguchi & Hashimoto (2010). It actually consists in a region-growing scheme that starts from a seed pixel and propagates in accordance to a set of rules that take into account the geometry of the crack. In this case, the rules are defined to identify dark, narrow, and elongated sets of connected pixels.

Seed pixels are defined over a regular grid with a step of Δ pixels, and are required to coincide with an edge not belonging to an already detected crack and whose gray level is dark enough. To ensure that the relevant edges are always considered, a dilation step follows the edge detection, where the dilation thickness is in accordance to Δ. Furthermore, since, as mentioned above, the crack detector operates under the guidance of the corrosion detector, seed pixels are required to have been labelled as corrosion by WCCD. The propagation proceeds over the dark neighbouring pixels until reaching an $N \times N$ boundary. Then, the elongation of the percolated area is checked to be large enough. If that is the case, the percolation process continues until reaching an $M \times M$ ($M > N$) boundary. The final percolated area is classified as a crack if: (1) its average gray level is dark enough and (2) its elongation is large enough. Otherwise, the region is discarded, and another percolation starts at a different seed pixel.

5.2.3 *Illustrative results*

Figure 8 shows some examples of corrosion detection. The output is colour-coded in accordance to how frequent is the colour of the underlying pixel in the CBC class: the more reddish, the more frequent.

Next, Figure 9 shows some results of crack detection followed and guided by corrosion detection.

6 CONCLUSIONS

This paper has described a collection of tools, developed within the frameworks of the INCASS and the MINOAS projects, whose goal is to provide decision support at the defect detection level during inspection operations. This toolbox comprises an aerial platform for inspection data collection, together with the underlying control architecture, specifically developed for visual inspection operations, and a number of software tools aiming at improving the presentation of inspection data (through image stitching) and the automatic detection of defects.

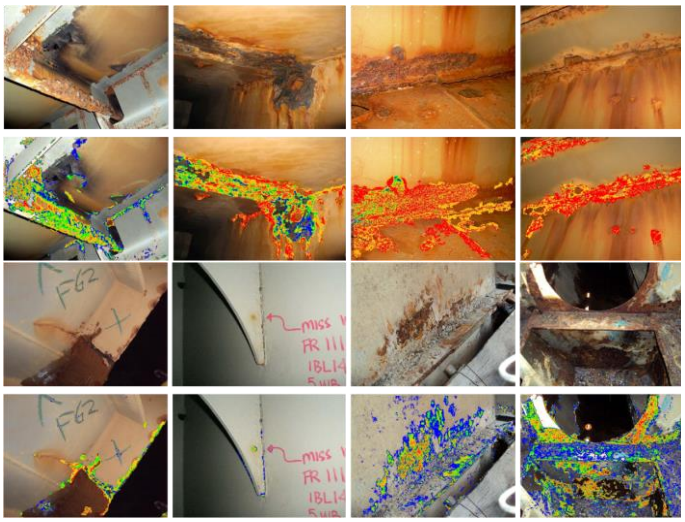

Figure 8. Examples of CBC detection: [rows 1 & 3] original images, [rows 2 & 4] processed images (see text for the colour code).

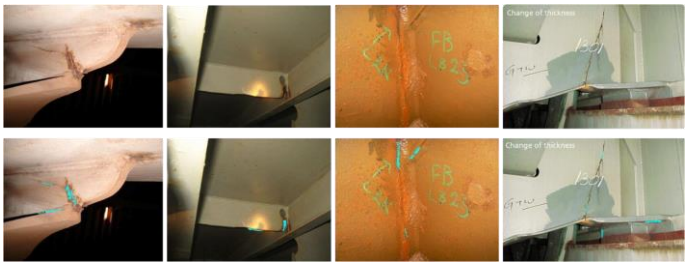

Figure 9. Examples of corrosion-guided crack detection: [top] original images, [bottom] processed images (cracks indicated in light blue).

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, Direccio General d'Innovacio i Recerca), by FEDER funding and by scholarship BES-2015-071804 (MINECO DPI2014-57746-C3-2-R). This publication reflects only the authors' views and the European Union is not liable for any use that may be made of the information contained therein.

REFERENCES

Achtelik, M., Lynen, S., Weiss, S., Kneip, L., Chli, M. & Siegwart, R. 2012. Visual-inertial SLAM for a small helicopter in large outdoor environments, in Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 2651–2652.

Arkin, R. 1998. Behavior-based Robotics. MIT press.

Bachrach, A., Prentice, S., He, R. & Roy, N. 2011. RANGE-Robust Autonomous Navigation in GPS-denied Enviroments, Journal of Field Robotics., vol. 28, no. 5, pp. 644–666.

Burt, P. & Adelson, E. 1983. A Multiresolution Spline with Application to Image Mosaics, ACM Transactions on Graphics, vol. 2, no. 4, pp. 217–236.

Cheng, G. & Zelinsky, A. 2001. Supervised Autonomy: A Framework for Human-Robot Systems Development, Autonomous Robots, vol. 10, pp. 251–266.

Chowdhary, G., Johnson, E., Magree, D., Wu, A. & Shein, A. 2013. GPS denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft, Journal of Field Robotics, vol. 30, no. 3, pp. 415–438.

Dryanovski, I., Valenti, R. & Xiao, J. 2013. An Open-source Navigation System for Micro Aerial Vehicles, Autonomous Robots, vol. 34, no. 3, pp. 177–188.

Duda, R., Hart, P. & Stork, D. 2000. Pattern Classification. Wiley Interscience.

Engel, J., Sturm, J. & Cremers, D. 2014. Scale-aware navigation of a low-cost quadrocopter with a monocular camera, Robotics and Autonomous Systems, vol. 62, no. 11, pp. 1646–1656.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G., Meier, L., Tanskanen, P. & Pollefeys, M. 2012. Vision-based Autonomous Mapping and Exploration Using a Quadrotor MAV, in Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 4557–4564.

Garcia-Fidalgo, E. & Ortiz, A. 2014. On the Use of Binary Feature Descriptors for Loop Closure Detection, in Proc. IEEE Intl. Conf. on Emerging Technologies and Factory Automation, pp. 1–8.

Grzonka, S., Grisetti, G. & Burgard, W. 2012. A Fully Autonomous Indoor Quadrotor, IEEE Transactions on Robotics, vol. 28, no. 1, pp. 90–100.

Gurdan, D., Stumpf, J., Achtelik, M., Doth, K., Hirzinger, G. & Rus, D. 2007. Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz, in Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 361–366.

Itti, L., Koch, C. & Niebur, E. 1998. A Model of Saliency-based Visual Attention for Rapid Scene Analysis, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254-1259.

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. 2011. ORB: An Efficient Alternative to SIFT or SURF, in Proc. IEEE Intl. Conf. on Computer Vision, pp. 2564–2571.

Shen, S., Mulgaonkar, Y., Michael, N. & Kumar, V. 2013. Vision-based state estimation for autonomous rotorcraft MAVs in complex environments, in Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 1758–1764.

Theodoridis, S. & Koutroumbas, K. 2008. Pattern Recognition. Academic Press.

Troiani, C., Martinelli, A., Laugier, C. & Scaramuzza, D. 2015. Low computational-complexity algorithms for vision-aided inertial navigation of micro aerial vehicles, Robotics and Autonomous Systems, vol. 69, pp. 80–97.

Yamaguchi, T. & Hashimoto, S. 2010. Fast crack detection method for large-size concrete surface images using percolation-based image processing, Machine Vision and Applications, vol. 21, no. 5, pp. 797-809.

Zhang, L., Tong, M., Marks, T., Shan, H. & Cottrell, G. 2008. SUN: A Bayesian Framework for Saliency Using Natural Statistics, Journal of Vision, vol. 8, no. 7, pp. 1–20.