2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Semi-Autonomous Visual Inspection of Vessels Assisted by an Unmanned Micro Aerial Vehicle

Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Alberto Ortiz

***Abstract*— Vessel maintenance entails periodic visual inspections of internal and external parts of the hull in order to detect the typical defective situations affecting metallic structures, such as cracks, coating breakdown, corrosion, etc. The main goal of the EU-FP7 project MINOAS is the automation of the inspection process, currently undertaken by human surveyors, by means of a fleet of robotic agents. This paper overviews a semi-autonomous approach to the inspection problem consisting of an autonomous *Micro Aerial Vehicle* (MAV) to be used as part of this fleet and which is in charge of regularly supplying images that can effectively teleport the surveyor from a base station to the areas of the hull to be inspected. Specific image processing software to analyze those images and assist the surveyor during the repair/no repair decision making process is also contributed. The control software approach adopted for the MAV, including self-localization and obstacle avoidance, is described and discussed, and experimental results in this regard are as well reported.**

I. INTRODUCTION

Vessels constitute one of the most cost effective ways of transporting 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 shipwrecks and, as such, *Classification Societies* 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.

The goal of the EU-funded FP7 MINOAS project is to develop a fleet of robots for automating the aforementioned inspection (and maintenance) operations as much as possible. Within this general context, this work presents an autonomous *Micro Aerial Vehicle* (MAV) to be adopted as part of the MINOAS re-engineered inspection process. As such, it is in charge of providing the surveyor with a first overview of the state of the metallic structures of the vessel, which requires a self-localized platform capable of attaining a set of waypoints while avoids obstacles and takes pictures of the specified views. After being conveniently tagged with positioning information, those images are to permit the same platform, or other platforms of the fleet, to re-visit, under demand of the surveyor, the same places for taking closer pictures that enable a more accurate assessment. Due to its inherent properties for flying indoors and close to other structures, quadrotors have resulted in the platform of election for this application.

With regard to the vehicle, the main requirement is the integration of a full navigation solution that covers the different functional and safety aspects needed to implement a typical inspection mission. Some recent navigational solutions can be found in this regard in the related literature, which particularly differ in the sensors used to solve the involved navigation tasks —namely, platform stabilization, self-localization, mapping and obstacle detection, using either infrared/ultrasounds [1], [2], [3], [4], laser scanners [5], [6], [7], or vision cameras [8], [9], [10], [11]—, and in the amount of processing that is performed onboard/off-board as well as in the assumptions made about the environment. In our case, a laser-based solution has been adopted for self-localization and obstacle avoidance, and all flight safety-related processing is performed onboard in order to reduce the use of wireless datalinks, which are not favoured within vessel holds due to the surrounding metallic structures.

A second contribution of this paper is an integrated algorithm for corrosion and crack detection. It runs on a separated (base) station, processing images from the platform as they arrive in order to assist the surveyor in almost real-time. The corrosion detector adopts a weak-classifier-based approach which learns from the texture and colour of image areas corresponding to corroded metallic surfaces. Besides, the corrosion detector guides the crack detection process by triggering a geometry-based crack detection scheme at those image areas where cracks can be expected, which reduces considerably false positives and running time. To the best of the authors' knowledge, no similar solution can be found in the related literature, apart from general detectors of anomalies in non-metallic materials (see [12]).

The rest of the paper is organized as follows: Section II describes in a more detailed manner the inspection problem and derives requirements for the platform, the MAV hardware and control software are presented in, respectively, Sections III and IV, experimental results are provided in Section V, Section VI describes the corrosion/crack detection solution adopted, as well as reports on its performance, and, finally, Section VII reviews the paper contributions.

This work is partially supported by FP7 project SCP8-GA-2009-233715 and the European Social Found through grants FPI10-43175042V and FPI11-43123621R (Conselleria d'Educacio, Cultura i Universitats, Govern de les Illes Balears)

F. Bonnin-Pascual, E. Garcia-Fidalgo and A. Ortiz are with the Department of Mathematics and Computer Science, University of Balearic Islands, 07122 Palma de Mallorca, Spain. {xisco.bonnin, emilio.garcia, alberto.ortiz} at uib.es

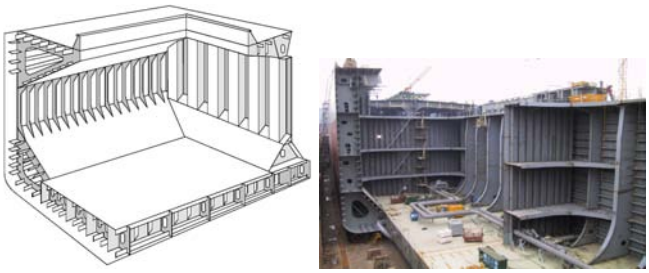

Fig. 1. (left) Typical structure of a bulk carrier. (right) Oil tanker in shipyard during construction.

II. INSPECTION PROBLEM AND PLATFORM REQUIREMENTS

To illustrate the enormity of the inspection task, the surveying of a central cargo tank on a *Very Large Crude Carrier* (VLCC) involves checking over 860m of web frames (primary stiffening members) and approximately 3.2km of longitudinal stiffeners, while the total inspection of a medium size cargo vessel can mean visually assessing more than 600,000m^2 of steel. Moreover, this often has to be carried out by a single surveyor within a short amount of time (generally a few days) in order to return the ship to service. Furthermore, this surveying is performed in a potentially hazardous environment with both flammable and toxic gases and significant heights involved. As a result, although accidents are extremely rare, when they do arise they can have serious consequences.

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 of the different metallic surfaces and structures (and for their repair if needed). Due to these complications, the total cost of a single surveying can exceed $1M once you factor in the vessel's preparation, use of yard's facilities, cleaning, ventilation, and provision of access arrangements. In addition, the owners experience significant lost opportunity costs while the ship is inoperable.

For those ships where there is a real cost saving [13], the main goal of the MINOAS project is to allow the surveyor to be teleported to those parts of the hull that require inspection as indicated above. To this end, the surveyor must be provided with imagery detailed enough so as to remotely enable the visual assessment of the state of the hull, so as to allow him to make proper *repair/no repair decisions*. As part of this general framework, the aerial platform is intended to provide detailed surveys of vertical structures such as the ones that are shown in Fig. 1. The main requirements stem, thus, from the very nature of the inspection process: the vehicle must be able to perform vertical, stationary and low speed flight, as well as permit indoor navigation. These requirements rapidly discard fixed-wing aircrafts and focus the search on helicopter-type MAVs, naturally capable of manoeuvres such as hovering and *Vertical Take-Off and Landing* (VTOL).

More precisely, the MAV is expected to implement phase 1 of the inspection missions. In this phase, the platform sweeps the relevant metallic surfaces and grabs pictures at a rate compatible with its speed, so that the surveyor can have an overall view of the vessel's condition. Those images must as well be tagged with pose information, so that, on demand of the surveyor, the areas suspected of being defective can be re-visited for acquiring close-up images, taking thickness measurements (by means of other platforms of the robot fleet), or even be compared in a posterior inspection.

Hence, the motion capabilities of the MINOAS flying robot comprise vertical take-off and landing, flying through waypoints and obstacle avoidance. Since the vertical structures that are found in vessel holds are quite similar along their full extent, 3D mapping does not result to be a critical requirement. Finally, the platform should not rely on GPS since the signal reception can not be ensured inside the vessel.

III. THE MICRO AERIAL VEHICLE

Among the different kinds of helicopter designs that have been proposed so far, multi-rotor configurations present several advantages over comparably scale helicopters (see, for instance, [14]): (1) they do not require mechanical linkages to vary rotor angle of attack as they spin, which simplifies the design of the vehicle and reduces maintenance time and cost; (2) the use of several rotors allows each individual rotor to have a smaller diameter than the equivalent helicopter rotor, for a given vehicle size; and (3) flight is safer than for other helicopters because the small rotor size make them store less kinetic energy during flight, which reduces the damage in case the rotors hit any object.

Among other multi-rotor UAVs, the four-rotor, or *quadrotor*, is emerging as the most popular multi-rotor configuration. Our MAV prototype is based on the well-known Pelican quadrotor from Ascending Technologies (see Fig. 2). This is a 50 cm-diameter platform with 25.4 cm propellers, able to carry a payload of 500 g, and equipped with a barometric pressure sensor for height estimation, a GPS receiver and an inertial measuring unit (IMU), comprising a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer. Attitude stabilization control loops making use of those sensors run over an ARM7 microcontroller as part of the platform firmware; the manufacturer leaves almost free an additional secondary ARM7 microcontroller, so that higher-level control loops (e.g. a position controller) can also be run onboard.

Furthermore, the MAV has been furnished with a lightweight laser scanning range finder. Platform motion is estimated by computing the roto-translation that makes consecutive laser scans match, and obstacle detection and avoidance is also implemented by means of this sensor. (Fig. 2 shows the MAV carrying a Hokuyo URG-04LX-UG01, 5.60 m range device, although, for increased operation range, it is replaced by a Hokuyo URG-UTM-30LX —up to 30 m range.)

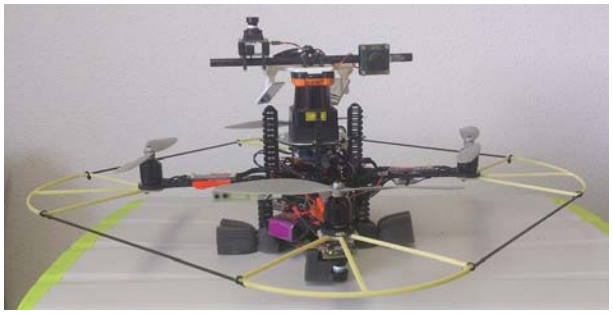

Fig. 2. The Micro Aerial Vehicle.

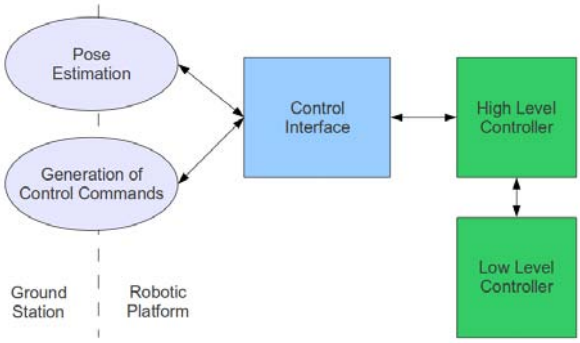

Fig. 3. Vehicle control architecture.

The laser device is also used, by deflection of lateral beams using mirrors, to estimate distance to the floor as well as to the ceiling (see Fig. 2). This method has been found more adequate for the application at hand (the accuracy is around 1-3% of the distance travelled by the beam), instead of using the barometric pressure sensor or the GPS, which tend to show large variations around the true height, making height stabilization difficult when the platform navigates indoors or relatively close to other objects.

Visual information is collected by means of a flexible vision system devised around an appropriate structure for supporting one bottom-looking camera and two additional units, which can be tailored for the particular inspection mission to be performed as: two forward-facing cameras forming a stereo vision system, one camera facing forward and the other facing up, or, to save weight, a single camera facing forward (Fig. 2 depicts a configuration consisting of two forward-facing uEye 1226-LE-C cameras organized as a stereo vision system, and a third uEye 1226-LE-C oriented to the bottom). All three cameras are intended to provide visual information about the state of the surfaces under inspection (either being at the front –e.g. web frames and walls in general–, at the bottom –the floor– or above the platform –e.g. cross-decks).

Finally, the vehicle carries an additional processing board which avoids sending sensor data to a base station, but process it onboard and, thus, avoid communications latency inside critical control loops. This processor will be referred to as the high-level processor from now on. (The current configuration includes a Kontron pITX-SP board equipped with an Intel Atom 1.6GHz processor and 2GB RAM.)

IV. CONTROL SOFTWARE ARCHITECTURE AND ORGANIZATION

As on similar platforms, the control software architecture comprises at least two physically separated agents: the MAV itself and a base/ground station. More specifically, the different computational resources of the MAV run the control algorithms as detailed next (either as firmware or as software): (1) as it is well known in the Pelican, the main ARM7 controller essentially runs the low-level software taking care of attitude stabilization and direct motor control [15] (in Fig. 3, it appears as the low-level controller); (2) the secondary ARM7 controller runs the position controller described in [11] (in Fig. 3, it appears as the high-level controller); and (3) the high-level processor executes, on top of the Robot Operating System (ROS [16]) running over Linux Ubuntu, ROS *nodes* providing platform motion estimation as well as platform safety, interaction with the onboard platform controllers and WiFi communication with the base station. Apart from this, the base station supporting the MAV also runs ROS over Linux Ubuntu. For this configuration, ROS becomes particularly relevant as it supplies the middleware functionality for transparent messages exchange between processes. Those that can tolerate latency in the communications are executed on the base station, while critical control loops run onboard the vehicle in the form of ROS *nodelets* in order to ensure minimum delay, a requirement also reported by other authors [4] to permit autonomous flying.

As well as the self-localization and mapping solution described in [17], our MAV control software has been designed around open-source components and following modularity and software reutilization principles. In this way, adapting the platform for different missions involving different payloads, or the selective activation of software modules, can be performed in a fast and reliable way. In this regard, ROS has also proved to be specially useful and, in fact, has guided the control software modularization.

Next sections comment on the details of the control architecture, whose top-level logical components are enumerated in Fig. 3.

A. Pose Estimation

Fig. 4 depicts the pose estimation system. It receives scan data and attitude angles (ϕ, θ, ψ) from, respectively, laser and IMU sensors and estimates the 3D pose of the vehicle.

Within this system, the *Laser Pre-processing* module prepares raw laser scans for the rest of the system. More precisely, it is in charge of: (1) filtering the laser beams that are of no interest; (2) projecting the scans comprising the beams of interest onto the ground, using the attitude information provided by the IMU; and (3) splitting the laser scans into several categories, so that beams reflected by the lateral mirrors are separated from beams providing information on detected obstacles ahead.

The *Vertical Position* module estimates the distance of the robot to the ground and to the ceiling. It uses, respectively,

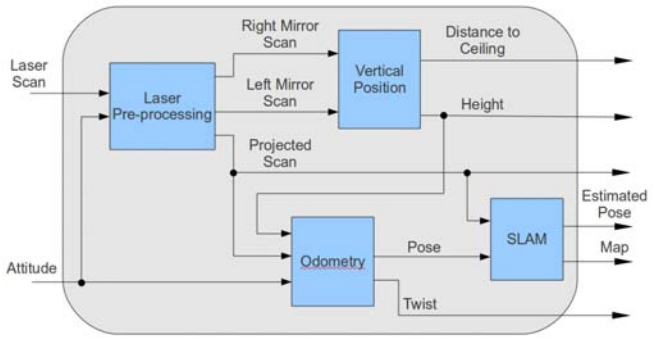

Fig. 4. Pose estimation.

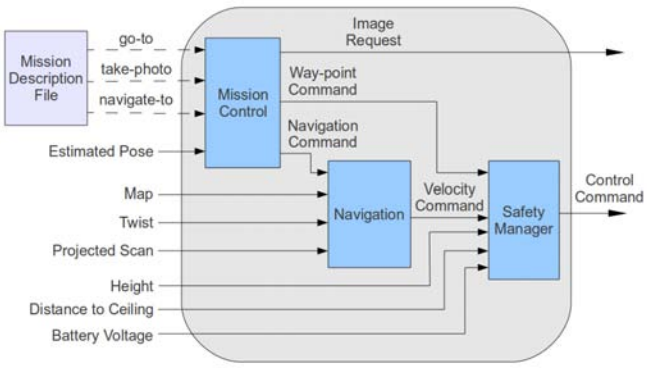

Fig. 5. Generation of control commands.

the laser beams which are deflected by the down-looking and up-looking mirrors. Apart from being useful for obstacle detection above and below the platform, depending on the mission and the environment, one or the other measurement feeds the vehicle height controller while flying, keeping constant the desired altitude or distance to the ceiling.

The *Odometry* component estimates the MAV 3D pose. First, the projected laser data is passed onto a scan matcher, which computes the platform roto-translation between consecutive scans and estimates a new 2D pose (x, y, ψ), using the yaw angle provided by the IMU as initial estimate for ψ. The 2D pose so obtained is then combined with the height and the roll and pitch angles (ϕ , θ), provided by, respectively, the laser altimeter and the IMU, to obtain a 3D pose for the vehicle. Robot speed is also estimated by this module.

In order to compensate the drift in the estimations produced by the scan matcher, a *Simultaneous Localization and Mapping* process is executed as part of the *SLAM* module. It receives projected laser scans and provides 2D corrections of the robot position and the environment map. Due to its high computational needs, this process runs on the base station. A further component within the *SLAM* module, responsible for monitoring the corrections provided by the SLAM, is executed onboard. This component limits the SLAM corrections to prevent the robot from moving too aggressively in tight environments. Furthermore, if the connection with the base station is suddenly lost, it keeps the platform operating with the last correction received. The public ROS package *gmapping*, based on [18], provides the SLAM functionality.

B. Generation of Control Commands

This system generates the control commands to be executed by the robot in accordance with the actions described in a mission specification file. See Fig. 5 for an overview.

The *Mission Control* component is executed on the base station and it is in charge of the accomplishment of the mission. The mission is described in an XML file as a sequence of actions. These actions can be: (1) *go-to*, which specifies a 3D pose to be attained by the vehicle, together with maximum speeds; (2) *navigate-to*, similar to the *go-to* action, but avoiding obstacles; and (3) *take-photo*, which requests for a picture to be taken by the robot using one of the attached cameras.

A client program, which is inside the *Mission Control* module, parses the mission specification file and invokes the corresponding tasks. A new action is sent if the previous one succeeds before a specified timeout. Otherwise, the mission is aborted and the vehicle hovers at the attained position. This module is also responsible for handling *go-to* and *take-photo* actions. Actions of the first kind are directly processed by the *Safety Manager* module, which filters out motion commands towards the high-level controller, while actions of the second kind are sent to the camera driver for image grabbing.

Navigate-to actions are handled by the 2D *Navigation* module, which includes functionalities such as reactive obstacle avoidance and path planning. Given a two-dimensional map and a waypoint, it generates the velocity commands needed to achieve the waypoint preventing collisions. These commands are also sent to the *Safety Manager* module. The *Navigation* module is a public ROS stack [19] that has been configured for using a *Dynamic Window Approach* local planner [20].

Finally, the *Safety Manager* is in charge of filtering all control commands before sending them to the *Control Interface*. Currently, it implements three safety behaviors: (1) it prevents the robot from flying too high or too close to the ceiling, (2) it monitors the battery voltage and sends commands for landing when this is lower than a safety threshold, and (3) it sends commands to make the robot hover when the wireless connection with the base station is lost.

V. MAV NAVIGATION RESULTS

This section reports on the execution of a number of missions to demonstrate the navigation capabilities of the MAV.

The first experiment, whose results are shown in Fig. 6, assesses the ability of the vehicle for navigating within an environment involving obstacles. It consists of reaching four waypoints in a room with a column in the middle that has to be avoided. At each location, the robot gets an image and stores it in the base station. The experiment was performed several times in order to check the repeatability of the task regarding waypoint achievement. In this regard, the small

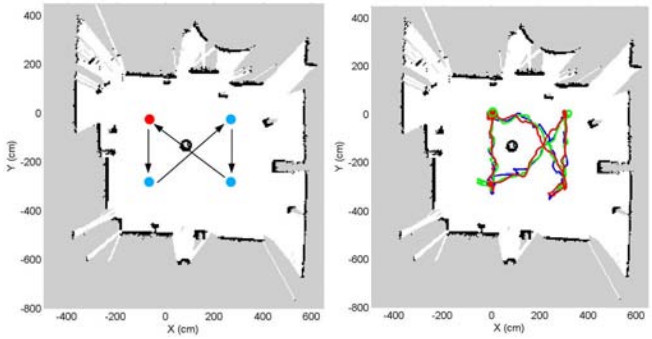

Fig. 6. (left) A map of the scenario for the first experiment. The red point indicates the beginning and the end of the experiment, the blue points show the locations to be reached by the vehicle, and arrows represent the direction of the robot. (right) Paths of three executions of the experiment in different colours. The MAV is able to avoid obstacles during mission execution.

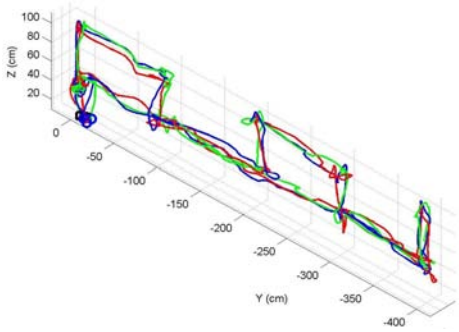

Fig. 7. Three executions of a sweeping mission, shown in different colours. The robot keeps the x coordinate almost constant during all the mission.

differences which can be observed are due to, among other factors, the accuracy parameter that determines when an action has succeeded or not (set to 0.1m in this experiment). The similarity between images taken at the same place can also be used as a measure of repeatability. Due to lack of space, this proof will only be given for the last experiment.

The second experiment corresponds to an inspection mission, which, at the scale of our laboratory, represents the kind of mission to be performed by the MAV onboard a real ship for providing the surveyor with an overall view of the state of a certain hull area. More precisely, the mission describes a sweeping task, consisting in achieving a total of thirteen waypoints along a zig-zag-like path. Again, the experiment was executed several times to compare results between consecutive executions. The resulting paths can be found in Fig. 7.

The last experiment combines the two previous missions in order to put together the different complexities the MAV will have to face in a real situation. To this end, a number of obstacles were spread throughout the room to make the global and local planners take part and produce safe paths towards the targets. The scenario is shown in Fig. 8, while a graphical description of the mission can be found in Fig. 9(left). As can be observed: first, the vehicle is instructed to reach a point at the other end of the room; then, a sweeping task is performed, and the corresponding images

Fig. 8. Images of the environment used to perform the last experiment. There are several obstacles which the vehicle has to avoid in order to, first, reach the initial point of the sweeping, and then reach the home location again, once the inspection has finished.

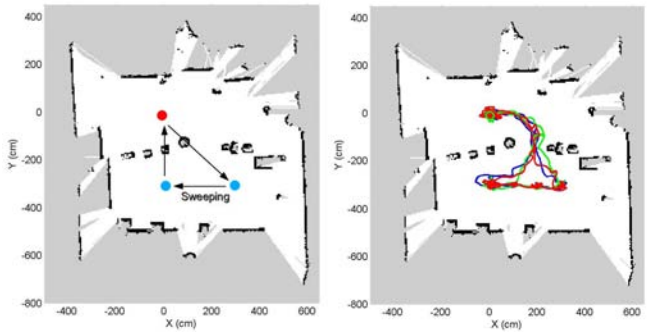

Fig. 9. (left) A map of the scenario for the third experiment. The red point indicates the beginning and the end of the experiment, the blue points show the locations to be reached by the vehicle, and arrows represent the direction of the robot. (right) Paths of three executions in different colours.

are taken; after the finalization of the inspection, the robot must return to the home position. Fig. 9(right) shows, for this experiment, the different paths followed by the vehicle during three executions of the experiment, while examples of the photos taken at the same location for two of the executions of the mission can be found in Fig. 10. The slight viewpoint changes that can be observed in the pictures are again due to the accuracy parameter, set to 6 degrees on this occasion.

A video corresponding to the third experiment is available at http://www.youtube.com/watch?v=L-m1Ey40OpQ&feature=plcp.

VI. INTEGRATED SOLUTION FOR CORROSION AND CRACK DETECTION

A *Defect Inspection Assistant* has been developed as another tool at the service of the surveyor while making *repair/no repair decisions*. This software can receive images taken by the platform and processes them looking for evidences of corrosion and cracks almost in real-time. The details can be found next.

A. Corrosion detection

The corrosion detector has been built around a supervised classification scheme implemented as two stages running a weak classifier each, following a cascading approach by which fast classifiers with poor performance alone lead to a global classifier of better performance [21]. This algorithm will be referred to as WCCD (*Weak-classifier Colour-based Corrosion Detector*) from now on.

Fig. 10. Pairs of images taken at the same location in two different executions of the last experiment. The slight deviations in viewpoint are due to the orientation accuracy parameter, set to 6 degrees in the mission specification file. The drawing at the bottom shows the positions where the photos are to be taken during the inspection mission. The pairs above correspond to, respectively, locations 2, 4 and 6.

The first stage of the 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 downsampled intensity values between 0 and 31, for a given direction α and distance d [21]. The energy of an image patch is then obtained by means of $E = \sum_{i=0}^{31} \sum_{j=0}^{31} p(i,j)^2$, where $p(i,j)$ is the probability of the co-occurrence of gray levels i and j at distance d and orientations α and $\alpha + \pi$. Patches with an energy lower than a given threshold τ_E, i.e. exhibit a rough texture, are candidates to be more deeply inspected.

Unlike the first stage, the second 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 pixels from 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 colour of corrosion, it is used to prevent the well-known instabilities when computing hue and saturation values for colours close to white or black. In those cases, the pixel is classified as non-corroded. In order to learn the HS values for image pixels known to correspond to corroded surfaces, a two-dimensional histogram is built in a previous training step.

In order to generalize the HS histogram [12] to cases out of the training set, different standard techniques have been considered, such as downsampling and/or Parzen windows for different two-dimensional kernels [21]. Although considerable improvements can be observed for those methods, best results have been obtained following a smoothing approach by means of a bilateral filter [22], which combines two Gaussians, one that operates at the spatial domain and the other at the intensity domain.

B. Corrosion-guided crack detection

A crack detector guided by the output of WCCD has been implemented after the observation that most cracks in metallic surfaces coincide, at least partly, with corroded areas. This algorithm will be referred to as GPCD (*Guided Percolation-based Crack Detector*) from now on.

The crack detection algorithm is based on a percolation model, similarly to the detector described in [23], which takes into account the crack geometry within a region-growing scheme. The region-growing procedure starts from an edge pixel suspected from being affected by corrosion. Additionally, it is required to be darker than a threshold γ_s and must not belong to an already detected crack. The propagation proceeds over the dark neighboring pixels till reaching an $N \times N$ boundary. Then, the elongation of the percolated area is checked to be larger than ϵ_N. If that is the case, the percolation process continues until reaching an $M \times M$ boundary. The final percolated area is classified as a crack if: (1) its average gray level is darker than a threshold γ_{avg}, and (2) its elongation is larger than ϵ_M. The elongation is computed by means of the normalized second central moments of the region μ_{xx}, μ_{yy} and μ_{xy} as $\epsilon = \sqrt{1 - \frac{\mu_{xx}+\mu_{yy}-\sqrt{4\mu_{xy}^2+(\mu_{xx}^2-\mu_{yy}^2)}}{\mu_{xx}+\mu_{yy}+\sqrt{4\mu_{xy}^2+(\mu_{xx}^2-\mu_{yy}^2)}}}$ [24].

C. Performance assessment

The performance of WCCD depends on the performance of its different stages. Regarding the roughness stage, several experiments have been performed considering different values for d and α when computing the GLCM and, consequently, its energy level.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 colour checking step.

Examples of classification outputs for WCCD are provided in Fig. 11. In these experiments the following values have been used: $\alpha = 0$ (horitzontal direction), $d = 5$ pixels and $\tau_E = 0.05$. The corrosion detected is colour-marked depending on the probability of being a successful detection: the warmer is the colour the higher is the probability. For a test set comprising a total of 7384 patches, global WCCD performance has been measured as 9.80% for the false positive percentage (*FP / #pixels*) and 5.86% for the false negative percentage (*FN / #pixels*).

Regarding GPCD, its performance was assessed after a proper configuration of its different parameters. The parameters related with the expected elongation of cracks, ϵ_N and ϵ_M, and the gray level thresholds γ_s and γ_{avg}, were all tuned so as to reduce as much as possible the number of false positives over the test set, while the values for *N* and *M*, related with the size of the percolation boundaries, were determined using the mean value of Pratt's FOM measure [25] calculated for all the test images. Fig. 12 shows some results of crack detections, where the following values have been used: $N = M = 41$, $\epsilon_N = \epsilon_M = 0.3$, $\gamma_s = 0.5$ and $\gamma_{avg} = 0.4$. The global performance was measured as

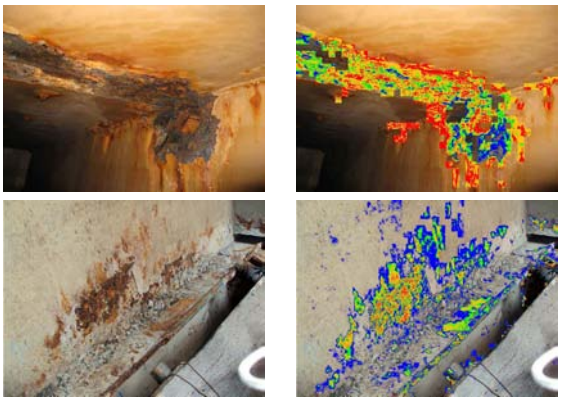

Fig. 11. Corroded areas detected by WCCD.

Fig. 12. Cracks detected by GPCD.

0.72% and 0.57% for, respectively, the false positive and false negative percentages.

To finish, execution times for WCCD ranged between 7 and 15 ms for images comprising from 120.000 to 172.800 pixels, while GPCD took between 30 and 150 ms for images of similar size. Tests were performed on an Intel Core2 Duo @2.2 GHz processor with 4 GB of RAM.

VII. CONCLUSIONS

A Micro Aerial Vehicle intended to assist human surveyors during visual inspections of vessels has been described. It is based on a commercial platform which integrates a control architecture intended to cover the requirements imposed by the inspection missions. The details and organization of the control software have been described and discussed. Results for a number of experiments have as well been reported, showing the suitability of the platform for the problem at hand. An integrated solution for corrosion and crack detection has been described, as well.

Videos showing the performance of the different parts of the solution described in this paper can be found in http://www.youtube.com/user/MINOASProject.

ACKNOWLEDGMENT

The authors of this paper would like to thank Markus Achtelik for his support on the use of the ROS *Asctec MAV framework*.

REFERENCES

[1] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Towards Autonomous Indoor Micro VTOL," *Autonomous Robots*, vol. 18, pp. 171–183, 2005.

[2] A. Matsue, W. Hirosue, H. Tokutake, S. Sunada, and A. Ohkura, "Navigation of Small and Lightweight Helicopter," *Transactions of the Japan Society for Aeronautical and Space Sciences*, vol. 48, no. 161, pp. 177–179, 2005.

[3] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor Using Minimal Sensing For Autonomous Indoor Flight," in *Proc. European Micro Air Vehicle Conf. and Flight Competition*, 2007.

[4] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments," *International Aerial Robotics Competition*, pp. 582–586, 2009.

[5] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice, and N. Roy, "RANGE - robust autonomous navigation in GPS-denied environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1096–1097, 2010.

[6] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 2878–2883, 2009.

[7] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1814–1820, 2008.

[8] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 21–28, 2010.

[9] G. Buskey, J. Roberts, P. Corke, and G. Wyeth, "Helicopter automation using a low-cost sensing system," *Computing Control Engineering Journal*, vol. 15, no. 2, pp. 8–9, 2004.

[10] S. Hrabar and G. Sukhatme, "Vision-based navigation through urban canyons," *Journal of Field Robotics*, vol. 26, no. 5, pp. 431–452, 2009.

[11] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 3056–3063, 2011.

[12] F. Bonnin-Pascual, "Detection of Cracks and Corrosion for Automated Vessels Visual Inspection," Master's thesis, University of Balearic Islands (Spain), 2010.

[13] A. Ortiz, F. Bonnin, A. Gibbins, P. Apostolopoulou, W. Bateman, M. Eich, F. Spadoni, M. Caccia, and L. Drikos, "First steps towards a roboticized visual inspection system for vessels," in *Proc. IEEE Intl. Conf. on Emerging Technologies and Factory Automation*, pp. 1–6, 2010.

[14] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[15] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 361–366, 2007.

[16] "ROS: an open-source Robot Operating System," in *Proc. of ICRA Workshop on Open Source Software*, 2009.

[17] I. Dryanovski, W. Morris, and J. Xiao, "An Open-Source Pose Estimation System for Micro-Air Vehicles," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 4449–4454, 2011.

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[19] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2010.

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

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

[22] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *Proc. IEEE Intl. Conf. on Computer Vision*, pp. 839 – 846, 1998.

[23] T. Yamaguchi and S. Hashimoto, "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, 2010.

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

[25] W. Pratt, *Digital Image Processing*. John Wiley and Sons, 1991.