Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

A Micro-Aerial Vehicle based on Supervised Autonomy for Vessel Visual Inspection

FRANCISCO BONNIN-PASCUAL, ALBERTO ORTIZ, EMILIO GARCIA-FIDALGO and JOAN P. COMPANY

March / 2015

A-02-2015

TECHNICAL REPORT

A Micro-Aerial Vehicle based on Supervised Autonomy for Vessel Visual Inspection

Francisco Bonnin-Pascual, Alberto Ortiz, Emilio Garcia-Fidalgo and Joan P. Company

Abstract

Seagoing vessels have to undergo regular visual inspections in order to detect the typical defective situations affecting metallic structures, such as cracks and corrosion. These inspections are currently performed by ship surveyors manually at a great cost. To make ship inspections safer and more cost-efficient, this paper presents a *Micro-Aerial Vehicle* (MAV) intended for visual inspection and based on supervised autonomy. On the one hand, the vehicle is equipped with a vision system that effectively teleports the surveyor from the base station to the areas of the hull that need inspection. On the other hand, the MAV is the result of a complete redesign of a visual inspection-oriented aerial platform that we proposed some years ago, with the aim of introducing the surveyor in the control loop and, in this way, enlarge the range of inspection operations that can robustly be carried out. Another goal is to make the platform as usable as possible for a non-expert. All this has been accomplished by means of the definition of different autonomous functions, including obstacle detection and collision prevention, and extensive use of behavior-based high-level control. The results of some experiments conducted to assess both the performance and usability of the platform are discussed at the end of the paper.

I. INTRODUCTION

The movement of goods by vessels is today one of the most time and cost effective ways of transportation. The safety of these ships is supervised by the so-called *Classification Societies*, which are continually seeking to improve standards and reduce the risk of maritime accidents. Structural failures are the major cause of such accidents, whose catastrophic consequences are unfortunately well-known. For these reasons, vessels are periodically inspected to ensure that the hull surfaces and structures are all in good condition.

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 vessels preparation, use of yards facilities, cleaning, ventilation, and provision of access arrangements. In addition, the owners experience significant lost opportunity costs while the ship is inoperable. For all these reasons, the introduction of any degree of automation into the surveying process by means of inspection-oriented robotic platforms is thus fully justified.

One of the main goals of the already concluded EU FP7 project MINOAS was to develop a fleet of robotic platforms with different locomotion capabilities with the aim of teleporting the human surveyor to the different vessel structures to be inspected. Given the enormity of these structures and the requirement for vertical motion as part of the inspection process, a multirotor platform was selected as one of the members of the fleet due to their small size, agility and fast deployment time. These platforms, also known as Micro-Aerial Vehicles (MAVs) by some authors, have become increasingly popular in recent years. Of particular relevance for this research are the several MAV navigation solutions that can be found in the related literature, comprising platform stabilization, self-localization, mapping, and obstacle avoidance. 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. The laser scanner has been used extensively due to its accuracy and speed. For instance, [1], [2] proposed full navigation systems using laser scan matching and IMU fusion for motion estimation embedded within SLAM frameworks that enable such MAVs to operate indoors. In [1], [3], a multilevel approach is described for 3D-mapping tasks. Infrared or ultrasound sensors are other possibilities for implementing navigation solutions. Although they typically have less accuracy and require higher noise tolerance, several researchers (see [4]–[6]) have used them to perform navigation tasks in indoor environments, since they are a cheaper option than laser scanners. To finish, vision-based navigation has become quite popular for MAVs lately, mostly because of the richness of the sensor data supplied and the low price of the cameras. Nevertheless, the associated higher computational cost has led researchers to find optimized solutions that can run over low-power processors. Among them, some researchers propose visual SLAM solutions based on feature tracking, either adopting a frontal mono or stereo camera configuration (e.g. [7]) or choosing a ground-looking orientation (e.g. [8]). Others focus on efficient implementations of optical-flow calculations, either dense or sparse, and mostly from ground-looking cameras (e.g. [9]) or develop methods for landing, tracking, and taking off using passive (e.g. [10]), or active markers (e.g. [11]), also using a ground-looking camera.

This work is partially supported by the European Social Fund through grant FPI11-43123621R (Conselleria d'Educacio, Cultura i Universitats, Govern de les Illes Balears) and by project INCASS. This project has received research funding from the EU FP7 under GA 605200. This publication reflects only the author's views and the European Union is not liable for any use that may be made of the information contained therein.

All authors are with the Department of Mathematics and Computer Science, University of the Balearic Islands, 07122 Palma de Mallorca, Spain, email: xisco.bonnin@uib.es

As a first attempt of aerial vehicle for visual inspection of vessels, the MINOAS aerial platform consisted in an autonomous MAV, similar to the ones surveyed above, and fitted with a flexible set of cameras to provide a first overview of the vessel structures inside cargo holds [12]. The platform was devised for indoor/semi-indoor flights (i.e. GPS-denied scenarios), and featured way-point navigation and obstacle avoidance. These tasks were implemented by means of a 30 m-range laser scanner that was used to feed a laser-scan-based odometer and a SLAM algorithm. The mission was specified as a list of way-points to reach and the actions to perform once reached, e.g. take a picture. This aerial vehicle, and the rest of robotic platforms, were successfully delivered at the end of the project, satisfying the initial set of requirements [13].

Nonetheless, a complete redesign of, particularly, the aerial platform was decided after some constructive advices received during field testing, in order to make the aerial robot more suitable for the vessel inspection problem and the involved users. Ship surveyors essentially referred to the usability and flexibility of the platform: instead of way-point navigation, what required the specification of a precise list of points for each mission, they suggested the implementation of a friendly, flexible and robust way to interact with the platform so that they could take the robot to any point of e.g. a cargo hold without the need to be an expert pilot.

In this regard, this paper describes a novel semi-autonomous aerial platform intended for the visual inspection of vessels. This robot, which has been under development within the FP7-EU INCASS project, has been devised to include the human surveyor in the position control loop so that he/she can friendly indicate the movements to be performed by the platform. Meanwhile, the control architecture autonomously takes care of attitude stabilization and XYZ speed control, as well as the activation of hovering flight (i.e. 0 speed) whenever there is no command pending to be executed. A higher control layer implements additional functionalities which are in charge of providing further assistance to the operator in the form of added autonomous functions. This includes, among others, collision prevention, avoidance of getting too far from the wall under inspection, avoidance of flying too high, etc. This shared control is implemented via a *Supervised Autonomy* (SA) approach [14] and making use of behavior-based control technology [15]. All the autonomous functionality is implemented on the basis of the information supplied by two optical-flow sensors and range finders at different orientations.

The rest of the paper is organized as follows: Section II describes the robotic platform and the hardware configuration; Section III details the control software architecture, including state estimation (Section III-.1), behavior-based control architecture for command generation (Section III-.2), and flight control organization (Section III-.3); Section IV discuss on the results of some experiments; and Section V concludes the paper.

II. THE ROBOTIC PLATFORM

In line with the robotic platform developed for the MINOAS project, the new micro-aerial vehicle is based on a multi-rotor configuration. The software architecture has been designed to be hosted by any of the research platforms developed by Ascending Technologies (the quadcopters Hummingbird and Pelican, and the hexacopter Firefly). These vehicles are equipped with an inertial measuring unit (IMU) that comprises a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer. Attitude stabilization control loops using the IMU data and thrust control 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 or velocity controller) can also run onboard.

Figure 1 shows a Hummingbird platform fitted with the sensor suite intended for the inspection application. This includes two optical-flow sensors, one looking to the ground and the other pointing forward, to estimate the vehicle speed with regard to, respectively the ground and the front wall (i.e. the inspected surface). These devices are PX4Flow sensors developed within the PX4 Autopilot project [16]. They comprise a CMOS high-sensitivity imaging sensor, an ARM Cortex M4 microcontroller to compute the optical flow at 250 Hz, a 3-axis gyroscope for angular rate compensation, and a MaxBotix ultrasonic (US) sensor HRLV-EZ4, with 1 mm resolution, used to scale the optical flow to metric velocity values. The distance measured by the US sensor is an additional output supplied by the PX4Flow device.

The distances provided by both PX4Flow sensors are used together with the ranges supplied by two additional Maxbotix HRLV-EZ4 oriented to the left and to the right for obstacle detection and collision prevention. This kind of sensor provides information at 10 Hz and its detection range is up to 5 m. An additional range sensor has been installed for height estimation when flying above that distance. This is a Teraranger One device [17] which consists in an infrared time-of-flight measurement sensor that weigts less than 20 g and can detect an obstacle at a maximum distance of 14 m.

The vehicle carries an additional processing board which avoids sending sensor data to a base station, but process them onboard and, thus, prevent communications latency inside critical control loops. This processor will be referred to as the high-level processor from now on. The configuration shown in Fig. 1 includes a Commell LP-172 Pico-ITX board featuring an Intel Atom 1.86 GHz processor and 4 GB RAM.

Finally, a flexible vision system is attached to the vehicle to collect the requested images from the vessel structures under inspection. The specific configuration of this vision system depends on the particular inspection to perform and the payload capacity of the chosen platform. For instance, the Hummingbird shown in Fig. 1 features a minimalistic configuration comprising a single forward-looking uEye UI-1221LE camera, while the Pelican-based MAV described in [12], fitted with

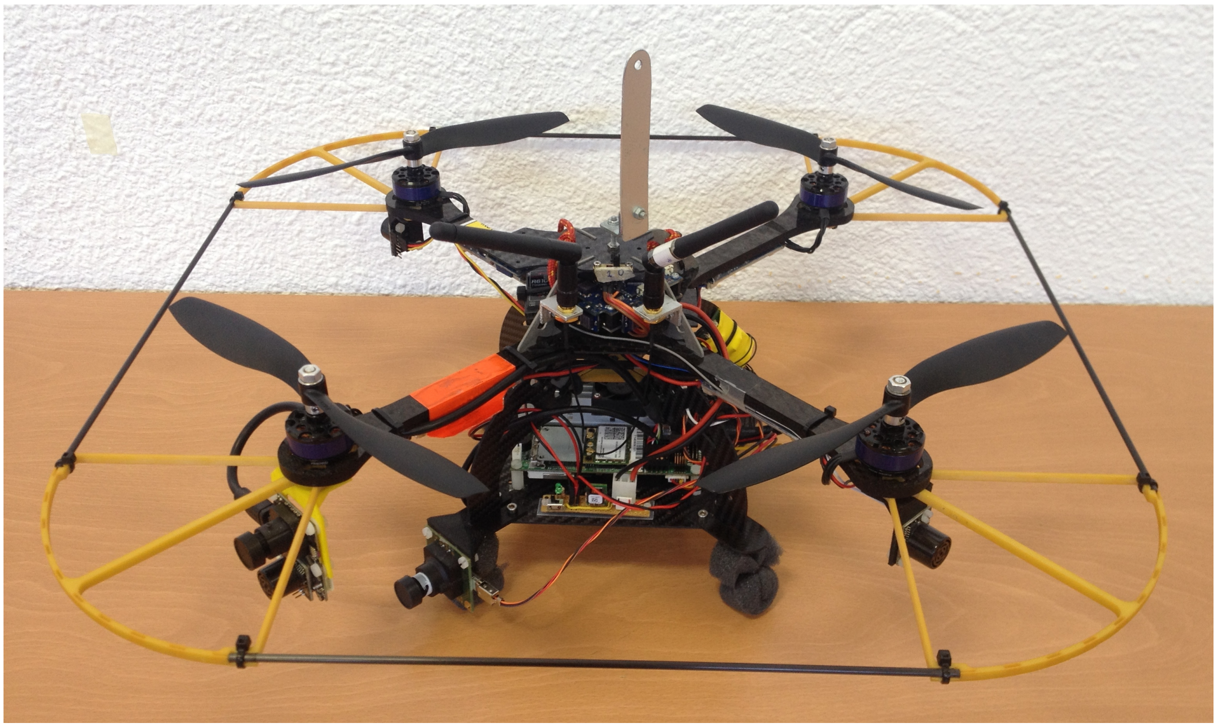

Fig. 1. The Hummingbird platform fitted with the sensor suite intended for the inspection application.

the same vision system, carried three cameras, oriented backward, forward and upward, to take pictures from all the relevant surfaces during the flight.

III. CONTROL ARCHITECTURE

The control software architecture is based on the SA paradigm [14]. This is a human-robot framework where the platform implements a number of autonomous functions, including self-preservation and other safety-related issues, that make simpler the intended operations for the user, so that he/she, which is allowed to be within the general platform control loop, can 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 a joystick to introduce the qualitative commands and a graphical user interface to receive the robot feedback. Regarding the robot capabilities, several behaviors have been developed to provide different autonomous functions that range from platform self-preservation to the functionality specifically devised for the defect inspection task.

The architecture comprises two separate agents: the MAV and the Base Station (BS). All the state estimation algorithms as well as the control algorithms run over the different computational resources of the MAV: the main ARM7 controller runs the attitude stabilization and direct motor control firmware provided by the manufacturer [18], while the secondary ARM7 runs the height and velocity controllers that have been developed for this work. The high-level processor executes, on top of the Robot Operating System (ROS) [19] running over Linux Ubuntu, ROS nodes to estimate the velocity and height of the vehicle (i.e. the vehicle state), as well as the distances to the closest obstacles situated in front and at both sides of the platform. This processor is also in charge of executing the different behaviors that implement the higher level autonomous functionalities of the platform. These behaviors combine the user desired speed command with the available sensor data, to obtain a final and safe speed set-point that is sent to the speed controllers.

The BS, which also runs ROS over Linux Ubuntu, is linked with the MAV via a WiFi connection, through which it sends the qualitative user commands introduced by means of the joystick. The BS also runs the GUI used to supply the operator with information about the state of the platform as well as about the task in execution, e.g. the images collected via the vision system attached to the platform.

The MAV control software has been designed around open-source components and following modularity and software reutilization principles. In this way, adapting the framework for different platforms 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.

Figure 2 overviews the control architecture. The different components are detailed along the next sections:

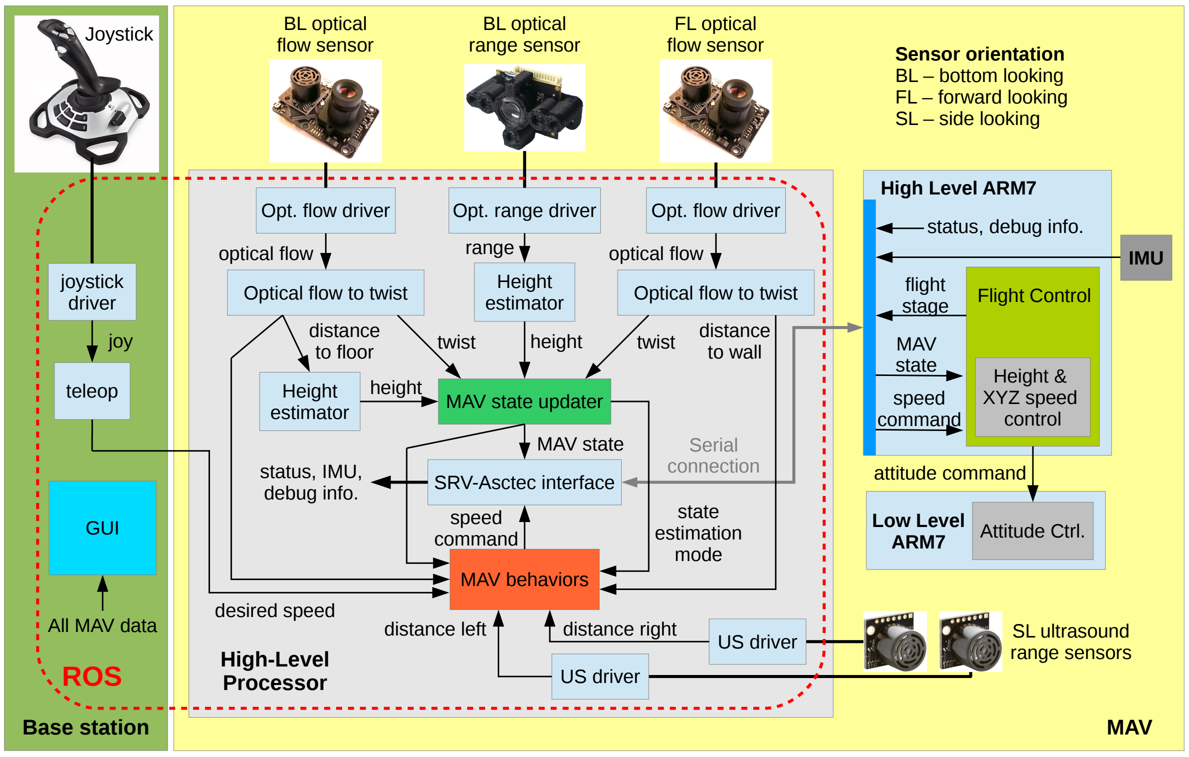

Fig. 2. Full system control architecture.

1) State estimation: The platform state includes the velocities along the three axes as well as the flight height. The estimation of these values is performed by the *MAV_state_updater* ROS node. It receives input from the different range and optical-flow sensors and combine them depending on their operational range. For example, when the platform is flying between 5 and 14 m height, the floor distance provided by the bottom-looking PX4Flow sensor cannot be used (since its range is limited to 5 m) so the vehicle height is estimated using the optical range sensor. A ROS node called *height estimator* is used to assess the change between two consecutive distance measurements and to decide whether this change is due to motion along the vertical axis or because of some discontinuity in the floor surface (e.g. the vehicle overflies a 1 m height box).

Regarding speed estimation, the *MAV_state_updater* module combines the information provided by the forward-looking and the ground-looking optical-flow sensors. The literature on aerial robots contains a number of navigation approaches based on optical flow. They mainly differ on the additional sensors that they combine to determine the vehicle speed. For instance, in [20] and [21], inertial sensors are used to estimate the scale of the optical flow and compute the metric measurements, while in [22] the authors are not interested in the optical-flow scale, but only in its direction to correct the inertial sensor drift. On the other hand, [23] and [16] combine the optical-flow sensor with an ultrasonic (US) range sensor to solve for the scale, resulting in an accurate speed estimator leading to low position drift. The drawback of this approach is that the flying height results limited to the maximum range of the US sensor (5 m in the case of the PX4Flow sensor).

In our architecture, the data provided by the two PX4Flow sensors is combined depending on the proximity to the inspected wall and to the ground. When only one of these surfaces is within the range of the corresponding sensor, the velocities over the plane that is parallel to this surface are estimated from the corresponding PX4Flow sensor, while the velocity along the perpendicular direction to that surface is estimated as the derivative of the distance provided by the range sensor included in the same PX4Flow. When both surfaces are within the scope of both PX4Flow sensors, the data available from the different sensors is combined in such a way that the velocities estimated by an optical-flow sensor are always preferred to the derivative of a range sensor. For instance, when flying in front of a vertical surface, within the scope of the forward-looking and the bottom-looking PX4Flow US sensors, the vehicle state is estimated as follows: the vertical velocity comes from the forward-looking PX4Flow sensor, while the lateral and longitudinal velocities, and the vehicle height, come all from the bottom-looking PX4Flow sensor. Table I shows the different modes for state estimation that the platform can activate depending on the operative sensors. As can be observed, there are two alarm modes (negative identifiers) that are activated when there is no reference surface available for estimating part of the state. These modes fill the missing MAV state components with values that make the vehicle descend.

TABLE I
STATE ESTIMATION MODES.

	Sensors availability			Sensors used to estimate...			
Mode	BL PX	BL TR	FL PX	**Height**	**Long. Speed**	**Lat. Speed**	**Vert. Speed**
0	OK	OK	N/A	BL PX range	BL PX flow	BL PX flow	BL PX range integ.
1	OK	OK	OK	BL PX range	BL PX flow	BL PX flow	FL PX flow
2	N/A	OK	OK	BL TR	FL PX range integ.	FL PX flow	FL PX flow
3	N/A	N/A	OK	FL PX flow integ.	FL PX range integ.	FL PX flow	FL PX flow
-1	N/A	OK	N/A	BL TR	0	0	Positive value
-2	N/A	N/A	N/A	Increasing value	0	0	Positive value

BL stands for bottom-looking and *FL* for forward-looking
PX refers to the PX4Flow sensor
TR refers to the TeraRanger One sensor

Speed is estimated for longitudinal, lateral and vertical axes
N/A means that the sensor data are not available
integ. means integration

2) Behavior-based command generation: The speed command is generated by the *MAV_behaviors* ROS node combining the user desired speed with the available sensor data through a reactive control strategy. This node implements different robot behaviors that are organized in a hybrid competitive-cooperative framework [15]. More precisely, on the one hand, higher priority behaviors can overwrite the output of lower priority behaviors by means of a suppression mechanism taken from the *subsumption* architectural model. On the other hand, the cooperation between behaviors with the same priority level is performed through a *motor schema*, where all the involved behaviors supply each a motion vector, so that the final output is the weighted summation of all motion vectors. An additional flow control mechanism has been incorporated to select, according to a specific input, between the output provided by two or more behaviours, i.e. a sort of multiplexer.

Figure 3 details our behavior-based architecture, showing how the different behaviors are organized and how they contribute to the final speed command. The different behaviors are grouped depending on its purpose. For the particular case of visual inspection, we have identified a total of four general categories:

- *Behaviors to accomplish the user intention.* This group comprises the *attenuated_go*, the *attenuated_inspect* and the *waiting_for_connectivity* behaviors. *attenuated_go* propagates the user desired speed vector command, attenuating it towards zero in the presence of close obstacles. *attenuated_inspect* proceeds in the same way, but it is only activated 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 a constant distance and orientation with regard to the front wall, for improved image capture. Finally, the *waiting_for_connectivity* behavior sets zero speed (i.e. hovering) when the connection with the BS is lost.
- *Behaviors to ensure the platform safety within the environment.* This category includes the *prevent_collision* behavior, which generates a repulsive vector to separate the platform from surrounding obstacles, whose magnitude increases with proximity. The joint actuation of this behavior and the *attenuated_go/inspect* behaviors implements the collision avoidance functionality onboard the platform. A second behavior called *limit_max_height* produces an attraction vector towards the ground when the vehicle is approaching its maximum flight height. A last behavior called *ensure_reference_surface_detection* generates suitable vectors that keep the platform close enough to at least one of the reference surfaces (the ground or the front wall), to ensure proper state estimations from the optical-flow sensors.
- *Behaviors to increase the autonomy level.* This category comprises the behaviors that provide higher levels of autonomy to both simplify the vehicle operation and to introduce further assistance during inspections. In the current implementation, the user can only activate one of the *go_ahead* and *inspect_ahead* behaviors, although this category is expected to be populated with additional autonomous functionalities if deemed effective during visual inspections. *go_ahead* is in charge of keeping the user speed command until some obstacle is detected or a new desired speed is introduced. An analogous behavior called *inspect_ahead* is in use when the platform is flying in inspection mode.
- *Behaviors to check flight viability.* This group does not contribute to the final speed command but it is in charge of ensuring the flight can start or progress at a certain moment in time. The current version includes only a behavior named *low_battery_land* that makes the vehicle descend and land when the battery is almost exhausted.

3) Flight control: The MAV flight control is implemented as a finite state machine that comprises five states: landed, taking off, flying, descending and landing. The transitions between the states take place when the particular conditions are met. For example, the vehicle changes from *landed* to *taking off* when the user pushes the take-off button from the joystick. Some other transitions do not depend on the user commands but on sensor data and on the vehicle state, e.g. the vehicle changes from *taking off* to *flying* when the estimated height is above a certain value (0.5 m) or after some time at a high level of motor thrust. The complete state machine is shown in Fig. 4.

Notice that the landing procedure is split into two stages: *descending*, which is in charge of reducing the flight height to ensure that the platform is close enough to the floor (about 0.5 m) before the platform enters the *landing* stage, which

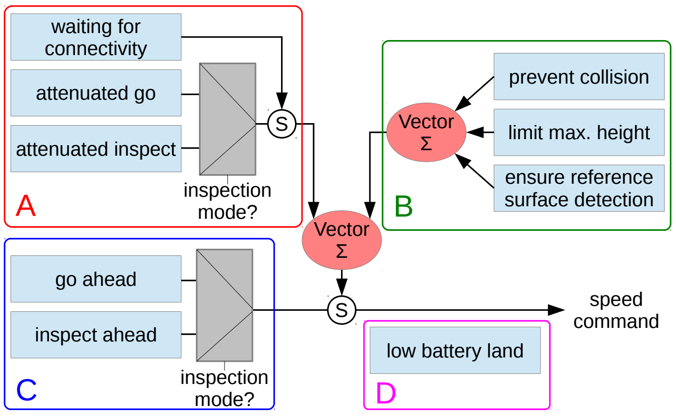

Fig. 3. MAV behaviors: A groups behaviors to accomplish the user intention, B groups behaviors that ensure the platform safety within the environment, C groups behaviors that increase the autonomy level, and D groups behaviors oriented to check flight viability.

performs a deceleration ramp of the motors thrust.

When the vehicle is in the *flying* stage, three PID controllers are in charge of keeping the speed command in the longitudinal, lateral and vertical axes (resp. X, Y and Z axes). When the speed command in the vertical axis is zero, the vertical speed PID controller is disabled while a height PID controller activates. While in the *descending* stage, the vertical speed controller is in charge of the platform although the different behaviours are still active to obey user-ordered movements along the X and Y axes, prevent collisions, etc.

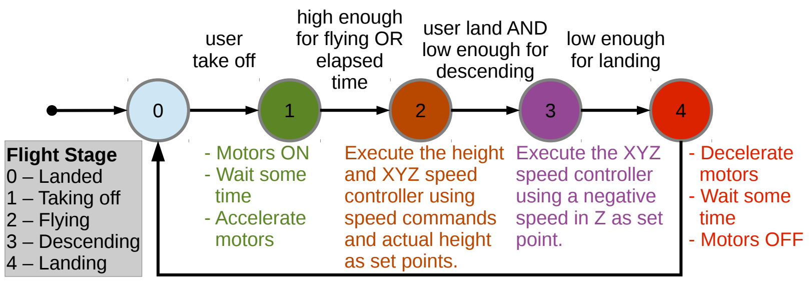
user take off
high enough for flying OR elapsed time
user land AND low enough for descending
low enough for landing
0
1
2
3
4
Flight Stage
0 – Landed
1 – Taking off
2 – Flying
3 – Descending
4 – Landing
- Motors ON
- Wait some time
- Accelerate motors
Execute the height and XYZ speed controller using speed commands and actual height as set points.
Execute the XYZ speed controller using a negative speed in Z as set point.
- Decelerate motors
- Wait some time
- Motors OFF

Fig. 4. Flight control state machine.

IV. EXPERIMENTAL RESULTS

This section reports about some of the experiments that have been carried out to check the usefulness of the control architecture for the inspection application. In a first kind of experiments, our concern has been the behaviour of the lower-level controllers, in particular what respects to the vehicle hovering capability since this maneouver becomes a key component within the SA approach, as this is the reaction of an aerial platform while flying and waiting for new commands. In this regard, we provide some performance results in Fig. 5. This figure plots several histograms of the speed values provided by the optical-flow sensors for the four state estimation modes (see Table I). As can be observed, all the histograms are approximately zero-centered, as expected.

In a second kind of experiments, we have evaluated the performance of the different behaviors. In a first experiment, we move the platform forward towards a wall to check how the platform behaves in a situation of imminent collision. The plot for this experiment is provided in Fig. 6. It shows how the longitudinal speed command provided by the *MAV_behaviors* node coincides with a user command of around 0.4 m/s until the wall in front of the vehicle becomes closer than 1.5 m (instant A), moment at which the user-desired velocity is attenuated by the *attenuated_go* behavior making the speed command decrease in accordance to the closeness of the wall. When the wall becomes closer than 1 m (instant B), which is the minimum distance allowed, the user-desired speed is completely cancelled by the *prevent_collision* behavior, and the platform stops. Notice that the user desired speed is still around 0.4 m/s until instant C.

A second experiment, reported in Fig. 7, checks the performance of the *go_ahead* behavior. At the beginning, the user indicates a longitudinal desired speed of 0.4 m/s and then activates the *go_ahead* behavior (instant A). At this moment, in accordance to the behavior definition, the speed command produced by the *MAV_behaviors* node keeps at 0.4 m/s although the user-desired speed returns to zero. This value is kept until the wall in front of the vehicle becomes closer than 1 m (instant B), when the *prevent_collision* behavior cancels the *go_ahead* command and stops the platform. This behavior is

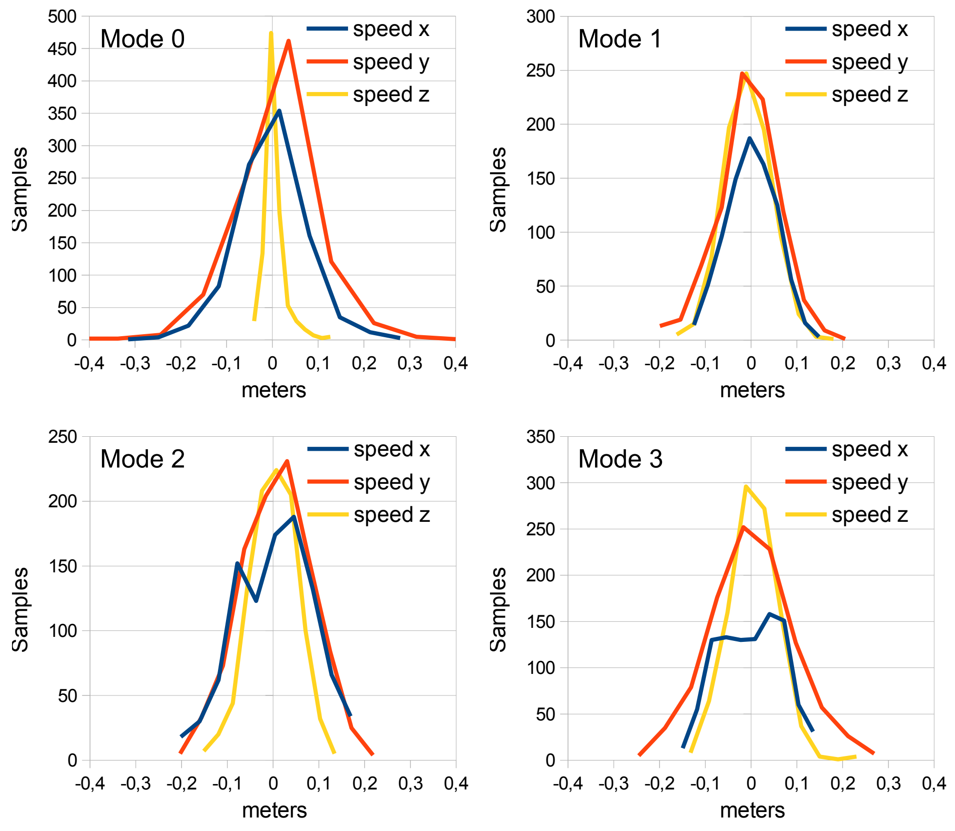

Fig. 5. Histograms of estimated speeds for 1-minute hovering flights for the four state estimation modes.

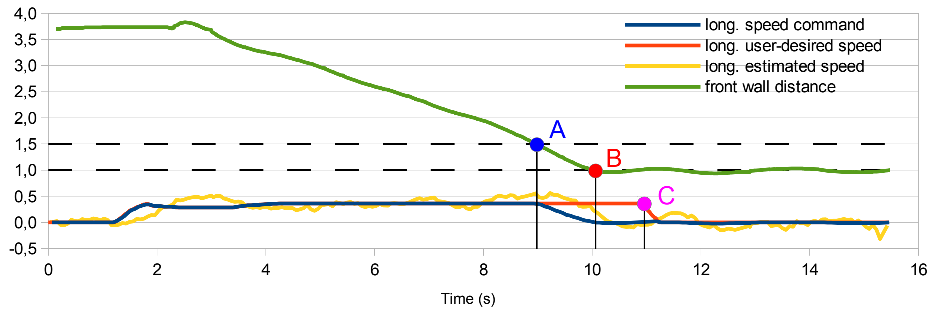

Fig. 6. Performance of the *attenuated_go* and the *prevent_collision* behaviors: the user-desired speed is obeyed (→A), it is attenuated (A→B) and cancelled to prevent an imminent collision (B→C) until the user-desired speed does become zero (C→). All units are in SI (m or m/s accordingly).

also in charge of producing the negative speed command that separates the platform from the wall until it is again at the safe distance (instant C).

In a third experiment, we check the performance of the *limit_max_height* behavior. Figure 8 shows how the vehicle ascends following the vertical user-desired speed (and the vertical speed command) until the platform reaches a height of 2.5 m (instant A), which was set as the maximum height for this experiment. From time instants A to B, the behavior prevents the platform from going higher ignoring the vertical user-desired speed until it becomes zero (instant B). Next, the platform descends since the user asks for a negative vertical speed between instants C and D.

Results for a fourth experiment are plotted in Fig 9. This case involves the *waiting_for_connectivity* behavior. At the beginning of the experiment, the user orders a negative lateral speed to move the platform to the right. During the displacement, the communication with the base station is lost (instant A), so that the user-desired speed signal is no longer available at the platform. As a consequence, the behavior takes control and makes the vehicle hover while waits for a reconnection. After 5 seconds (instant B), the communication link has not been restored and the behavior decides to make the platform land.

Figure 10 corresponds to a fifth experiment, aiming at checking the performance of the *low_battery_land* behavior. During

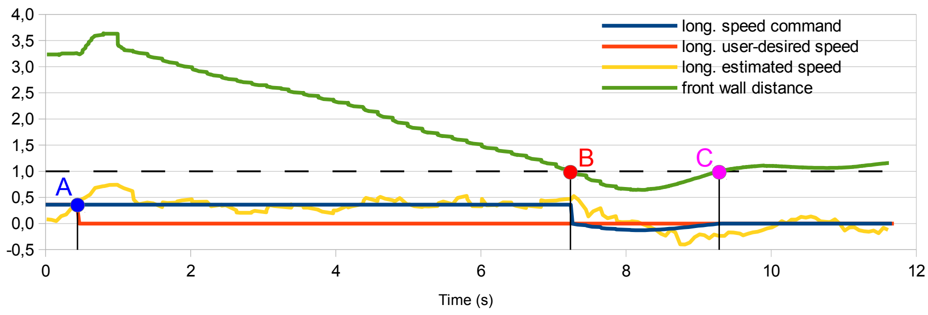

Fig. 7. Performance of the *go_ahead* and the *prevent_collision* behaviors: the user-desired speed is sustained while the wall is at enough distance (A→B), it is cancelled and even forced to be negative to prevent an imminent collision (B→C) until the platform is again at the safe distance (C→). All units are in SI (m or m/s accordingly).

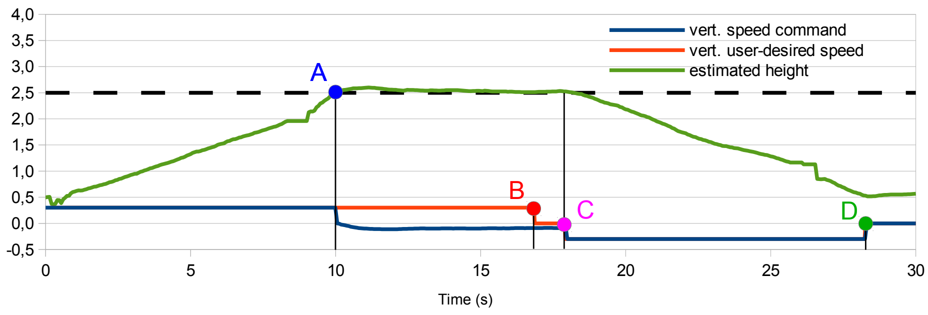

Fig. 8. Performance of the *limit_max_height* behavior: the user-desired vertical speed is obeyed until the platform reaches the maximum allowed height (→A), then the desired vertical speed is ignored (A→B) until it becomes zero (B→C), and finally the platform descends following again the desired speed (C→D). All units are in SI (m or m/s accordingly).

this experiment, the platform is left hovering at 1.5 m until the battery voltage becomes lower than 10 V (instant A). At this moment, the behavior takes control of the platform to make it land. As indicated in Section III-.3, the landing manoeuver starts with a *descending* stage, to make the platform reduce its height up to 0.5 m, and finishes with the deceleration of the motors thrust (instant B).

Figure 11 describes a sixth experiment aiming at checking the performance of the *ensure_reference_surface_detection* behavior. The experiment starts with the platform flying at a certain distance to the front wall so that the vehicle only makes use of the ground-looking optical-flow sensor to estimate its velocity (state estimation mode 0). Within this mode, the vehicle is allowed to ascend (action 1) until the maximum distance to the ground, the reference surface, is attained (the maximum distance was set to 1.5 m for this experiment). The next ascending orders (action 2) are ignored. Next, the vehicle moves towards the front wall (action 3) until the vehicle is close enough so as to also use this wall to estimate its state (state estimation mode 1). Once the vehicle is close to the wall, it moves upwards (action 4) until the ground becomes too far for the ground-looking optical-flow sensor (2 m for this experiment) and the front wall becomes the only reference surface (state estimation mode 2). Subsequently, the user tries to turn the vehicle to the right (action 5) but this action is ignored since the right wall is too far for the front-looking optical-flow sensor. Then, the user moves the platform towards the right wall to decrease that distance (action 6) and repeats the turning action (action 7), which is now allowed. Figure 12 plots sensor data for the full operation for the longitudinal, lateral, vertical and angular speeds. Distances to the front wall/right wall/ground are shown to make evident the corresponding motion.

In a last experiment involving specific behaviors, we show the performance of the platform during a typical inspection task. The operation starts when the platform takes off and the operator makes it approach the wall under inspection; at more or less 1 m distance, the inspection mode is activated, and, hence, longitudinal motion as well as rotations in yaw are not allowed to ensure better image capture conditions. The operator next orders lateral and vertical motion commands to sweep the surface taking pictures from time to time (10 Hz). The plots of Fig. 13 illustrate the full operation for the longitudinal (top), lateral (middle) and vertical (bottom) motions. These velocities are shown at the bottom of the plots, while distances to, respectively, the front wall/left wall/ground are shown at the top in green, to make evident the corresponding motion. Notice that, when the inspection mode is enabled (between instants A and B): (1) the longitudinal user-desired speed is ignored, (2) the user only has the option of selecting hovering or motion in the vertical or lateral direction, but the speed command is set to ±0.2 m/s. The plots also show repulsive speed commands produced when the platform is below 1 m

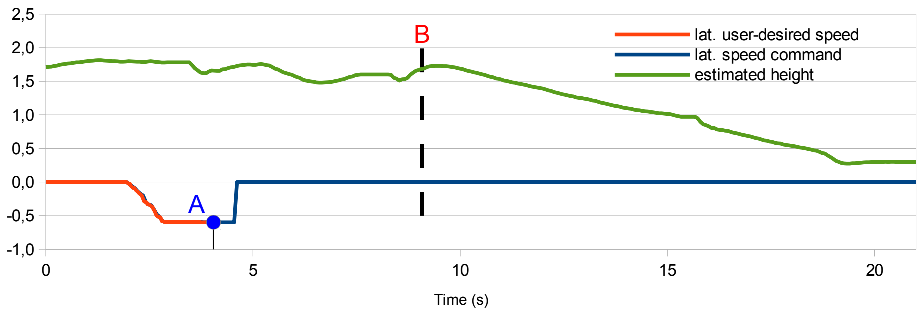

Fig. 9. Performance of the *waiting_for_connectivity* behavior: the communication with the base station is lost during the flight ($\rightarrow$A), what makes the behavior force a hovering maneouver (A$\rightarrow$B) while the vehicle tries to reconnect; after waiting for five seconds without success, the behavior makes the platform land (B$\rightarrow$). All units are in SI (m or m/s accordingly).

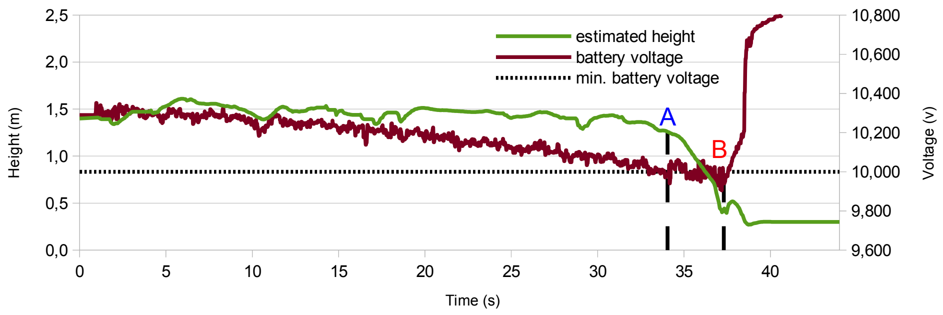

Fig. 10. Performance of the *low_battery_land* behavior: the platform hovers at 1.5 m until the battery voltage is below 10 V ($\rightarrow$A), what makes the behavior initiate the descending (A$\rightarrow$B) and landing manoeuvers (B$\rightarrow$). All units are in SI (m or m/s accordingly).

regarding the front or left wall (see instants C, D and E).

By way of conclusion, in a third kind of experiment, we assess the platform regarding its visual inspection performance, by flying in front of defective surfaces (e.g. the previous experiment) and processing later the images collected. Figure 14 provides results for one of these experiments. For this case, the MAV was flown in front of a 2.5×4 m surface containing corroded areas while the vision system was taking pictures at 10 Hz. During the flight, the platform was operated in *inspection mode* in order to prevent fast movements and hence reduce the image blurring. The 87 collected images were then processed by the image mosaicing algorithm described in [24], which managed to produce the seamless composite shown in Fig. 14 (left). Finally, the mosaic was analysed by the defect detector described in [25], whose result is available in Fig. 14 (right), where a successful detection of the defective areas can be observed.

Some complementary videos are available at https://www.youtube.com/channel/UCuEpP8RIsBqh4ZlDU6pC4eQ.

V. CONCLUSIONS

A Micro-Aerial Vehicle to be used for vessel visual inspection has been presented. The MAV control approach is based on the SA paradigm, and hence the user is introduced in the platform control loop. For the specific problem of visual inspection, we have proposed and fully described a behaviour-based control architecture tightly linked to the SA paradigm, including aspects suth as vehicle state estimation, behavior-based command generation and flight control. Successful experimental results, regarding control architecture validity and usefulness for visual inspection, have been reported.

In comparison with a previous MAV described in [12], the use of the SA paradigm has enhanced the usability of the platform as an inspection device, making easier, faster and more flexible the inspection process (e.g. mission specification files are no longer needed). Furthermore, the new platform does not require any assumption about the vertical structure of the vessel hull (the previous platform operated over a 2D map built by means of a laser scanner and a 2D SLAM algorithm, under the assumption of vertical similarity for the operating area) since its combination of sensors and behaviors take care of the safety of the platform while obeying (as far as possible) the user commands.

As future work, we plan to integrate new autonomous functionalities to enhance the inspection process, and intensive field tests with ship surveyors in order to incorporate their feedback in the platform design and improve its usability regarding the problem at hand.

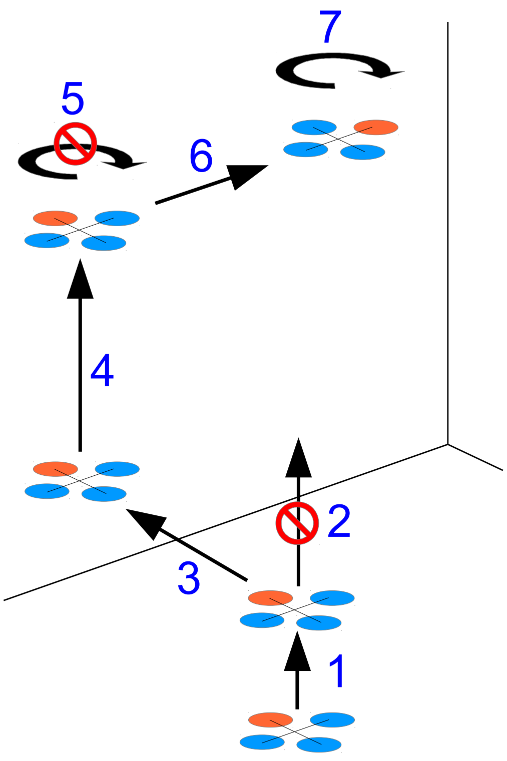

Fig. 11. Illustration of the performance of the *ensure_reference_surface_detection* behavior (I): see text for the explanation. All units are in SI (m or m/s accordingly).

References

[1] I. Dryanovski, R. G. Valenti, and J. Xiao, "An Open-source Navigation System for Micro Aerial Vehicles," *Autonomous Robots*, vol. 34, no. 3, pp. 177–188, 2013.

[2] S. Grzonka, G. Grisetti, and W. Burgard, "A Fully Autonomous Indoor Quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[3] A. Bachrach, S. Prentice, R. He, and N. Roy, "RANGE-Robust Autonomous Navigation in GPS-denied Environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

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

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

[6] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor Using Minimal Sensing for Autonomous Indoor Flight," in *European Conf. on Computer Vision*, 2007.

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

[8] G. Chowdhary, E. Johnson, D. Magree, A. Wu, and A. Shein, "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, 2013.

[9] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV Navigation through Indoor Corridors Using Optical Flow," in *IEEE Int. Conf. on Robotics and Automation*, 2010.

[10] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision," *Autonomous Robots*, vol. 33, no. 1–2, pp. 21–39, 2012.

[11] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1–4, pp. 221–238, 2011.

[12] F. Bonnin-Pascual, E. Garcia-Fidalgo, and A. Ortiz, "Semi-autonomous Visual Inspection of Vessels Assisted by an Unmanned Micro Aerial Vehicle," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 3955–3961.

[13] M. Eich, F. Bonnin-Pascual, E. Garcia-Fidalgo, A. Ortiz, G. Bruzzone, Y. Koveos, and F. Kirchner, "A Robot Application to Marine Vessel Inspection," *Journal of Field Robotics*, vol. 31, no. 2, pp. 319–341, 2014.

[14] G. Cheng and A. Zelinsky, "Supervised Autonomy: A Framework for Human-Robot Systems Development," *Autonomous Robots*, vol. 10, pp. 251–266, 2001.

[15] R. C. Arkin, *Behavior-based Robotics*. MIT press, 1998.

[16] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications," in *IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 1736–1741.

[17] M. Ruffo, M. D. Castro, L. Molinari, R. Losito, A. Masi, J. Kovermann, and L. Rodrigues, "New Infrared Time-of-flight Measurement Sensor for Robotic Platforms," in *IMEKO TC4 Int. Symposium and Int. Workshop on ADC Modelling and Testing*, 2014, pp. 13–18.

[18] 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 *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 361–366.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an Open-Source Robot Operating System," in *ICRA workshop on open source software*, 2009.

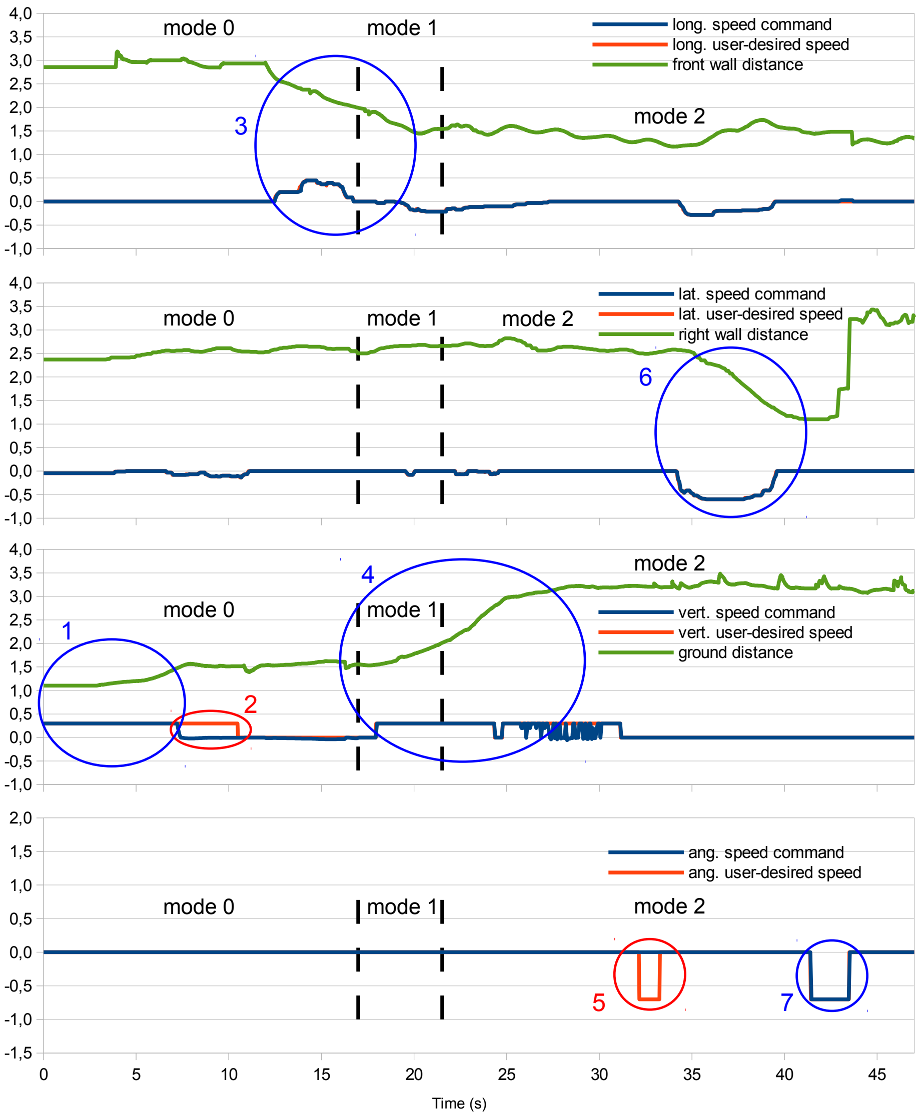

Fig. 12. Illustration of the performance of the *ensure_reference_surface_detection* behavior (II): see text for the explanation. All units are in SI (m or m/s accordingly).

[20] F. Kendoul, I. Fantoni, and K. Nonami, “Optic Flow-based Vision System for Autonomous 3D Localization and Control of Small Aerial Vehicles,” *Robotics and Autonomous Systems*, vol. 57, no. 6-7, pp. 591–602, 2009.

[21] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, “Hovering Flight and Vertical Landing Control of a VTOL Unmanned Aerial Vehicle Using Optical Flow,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 801–806.

[22] A. Briod, J.-C. Zufferey, and D. Floreano, “Optic-Flow Based Control of a 46g Quadrotor,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[23] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, “The Navigation and Control Technology Inside the Ar.Drone Micro UAV,” in *IFAC World Congr.*, 2011, pp. 1477–1484.

[24] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, “Vessel Visual Inspection: A Mosaicing Approach,” Department of Mathematics and Computer Science, University of the Balearic Islands, Palma de Mallorca, Tech. Rep. A-01-2015, March 2015. [Online]. Available: http://dmi.uib.es/~egarcia/TRA012015.pdf

[25] F. Bonnin-pascual and A. Ortiz, “A Probabilistic Approach for Defect Detection Based on Saliency Mechanisms,” in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Barcelona, Spain, 2014.

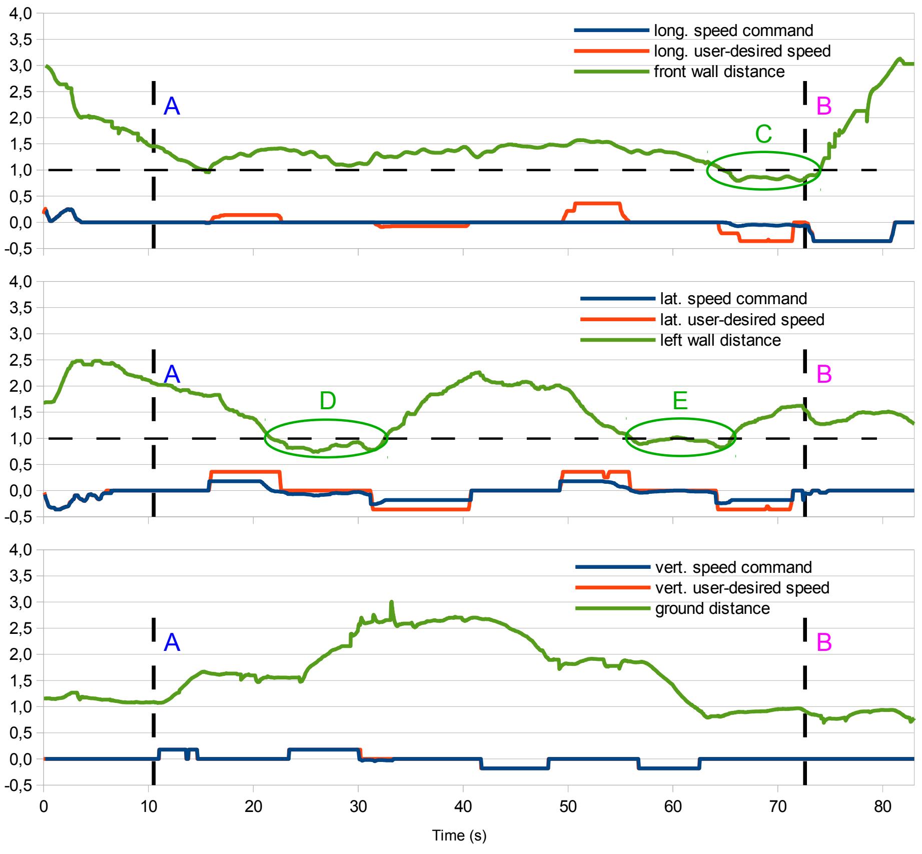

Fig. 13. Performance of the platform during an inspection task using the inspection mode (A→B). All units are in SI (m or m/s accordingly).

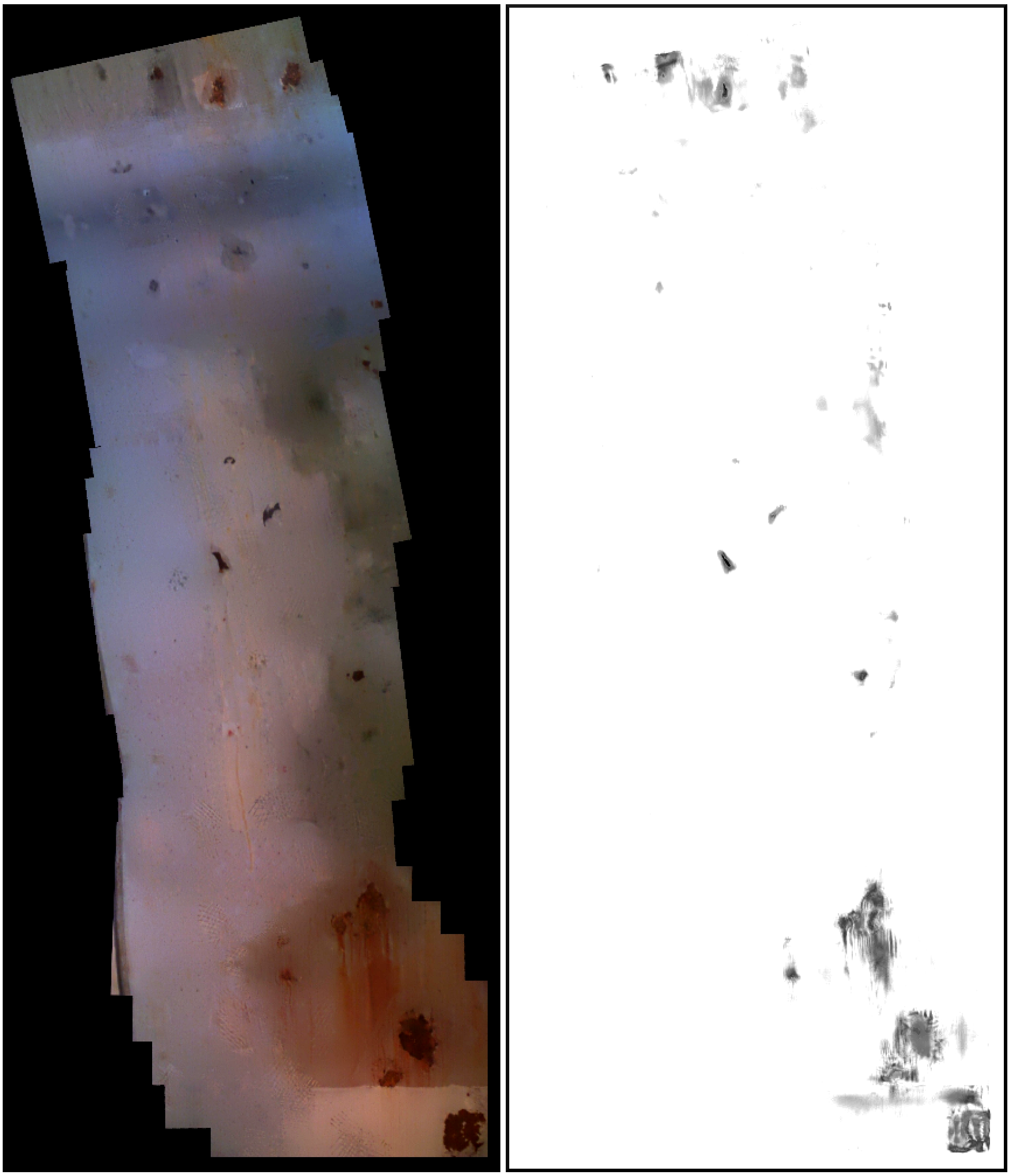

Fig. 14. Visual inspection performance: (left) mosaic built from images collected by the MAV [contrast tuned for visualization purposes], (right) defect detection result [darker pixels are likelier to correspond to defects].