A Micro Aerial Vehicle for Vessel Visual Inspection Assistance

Alberto Ortiz, University of Balearic Islands, alberto.ortiz@uib.es
Emilio García-Fidalgo, University of Balearic Islands, emilio.garcia@uib.es
Francisco Bonnín-Pascual, University of Balearic Islands, xisco.bonnin@uib.es

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 Micro Aerial Vehicle (MAV) to be used as part of this fleet, and describes the control software approach that has been adopted, with special emphasis on self-localization and obstacle avoidance. Experimental results in this regard are included and discussed at the end of the paper.

1. Introduction

The movement of goods by ships is today one of the most time and cost effective methods of transportation. The safety of these vessels is overseen by the so-called *Classification Societies*, who are continually seeking to improve standards and reduce the risk of maritime accidents. Structural failures are a major cause of accidents, and can usually be prevented through timely maintenance. As such, vessels undergo annual inspections, with intensive *Special* and *Docking Surveys* every five years, which ensures that the hull structure and related piping are all in satisfactory condition and are fit for the intended use over the next five years.

The main objective of the EU FP7 project MINOAS (*Marine Inspection rObotic Assistant System* [1]) is the effective virtual teleportation of the surveyor to the different areas of the vessel hull that need inspection, so that a reduction in the inspection time and the costs involved, as well as an increase in the safety of the operation, can be effectively achieved (see *Ortiz et al. (2010)* for a detailed discussion). To this end, the MINOAS project considers the adoption and development of a varied set of robotic technologies with different locomotion capabilities, including magnetic crawlers, *remotely operated vehicles* (ROV) and *unmanned aerial vehicles* (UAV).

This paper describes the configuration of a *Micro Aerial Vehicle* (MAV) to be adopted as part of the re-engineered MINOAS inspection process, reviews the requirements imposed by such an application, and comments on a control architecture intended to fulfill them. Experimental results towards fulfilling the intended inspection mission are as well provided and discussed.

The rest of the paper is organized as follows: Sections 2 and 3 describe the platform and the control software as well as discusses its adequacy regarding the requirements imposed by the inspection application, Section 4 reports results from some laboratory and field experiments, and, finally, Section 5 summarizes the contributions.

2. Platform description

As part of the MINOAS concept, the aerial platform is intended to provide a detailed survey of the vertical structures that can be found in vessel holds (see Figure 1 for an example). Therefore, the main requirements stem directly 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 flight. These requirements rapidly discard fixed-wing aircrafts and focus the search on helicopter-type UAVs, naturally capable

[1] http://www.minoasproject.eu

of manoeuvres such as hovering and *vertical take-off and landing* (VTOL).

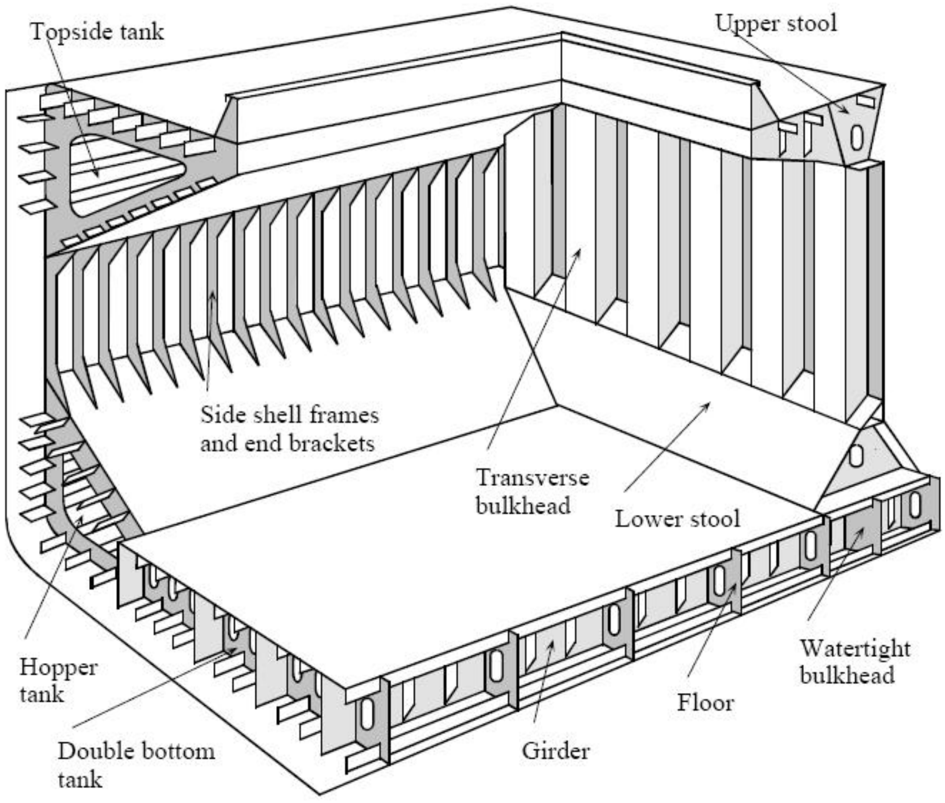

Fig. 1: Typical structure of a bulk carrier.

Among the different kinds of helicopter designs that have been proposed, multi-rotor configurations present several advantages over comparably scale helicopters (see, among others, *Pounds et al. (2010)*): (1) they do not require mechanical linkages to vary rotor angle of attack as they spin, what 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 makes them store less kinetic energy during flight, what reduces the damage in case the rotors hit any object.

Nowadays, the four-rotor, or quadrotor, is emerging as the most popular multi-rotor configuration for unmanned MAVs. This kind of vehicle consists of four rotors in total, with two pairs of counter-rotating, fixed-pitch blades located at the four corners of the aircraft. In this platform, the control of the vehicle motion is achieved by varying the relative speed of each rotor. Moreover, because each pair of rotor blades spin in opposite directions, they cancel out any torque, keeping the helicopter flying straight. As a result, precise flight and stable hovering can be achieved, even in narrow spaces. Finally, counter rotating propellers increase efficiency and flight times, as no extra thrust is needed to compensate for unwanted rotation.

Our MAV prototype is based on the well-known Pelican quadrotor from Ascending Technologies (see Figure 2). This is a 50 cm-diameter platform with 10-inch propellers, able to carry a payload of 500g, 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 scanning range finder (Figures 2 and 3 show the MAV carrying a Hokuyo URG-04LX-UG01, 5.60 m range device, although, for increased operation range, it can be replaced by a Hokuyo URG-UTM-30LX ─ up to 60 m range). In this way, platform motion can be estimated by computing the roto-translation that makes consecutive laser scans match, what is usually known as laser-based odometry. Obstacle avoidance is also implemented using the information provided by this sensor.

The laser device is also used, by deflection of lateral laser beams using mirrors, to estimate distance to the floor as well as to the ceiling (see Figure 3). 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 front-looking cameras forming a stereo vision system, one camera looking to the front and the other looking to the ceiling, or, to save weight, a single camera looking to the front (Figure 2 depicts a configuration consisting of two front-looking 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), as well as, if needed, contribute to the platform self-localization by means of appropriate visual odometry solutions.

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 configuration shown in Figure 2 includes a Kontron pITX-SP board equipped with an Intel Atom 1.6GHz processor and 2GB RAM.)

Table I summarizes the technical specifications for the platform and the payload units.

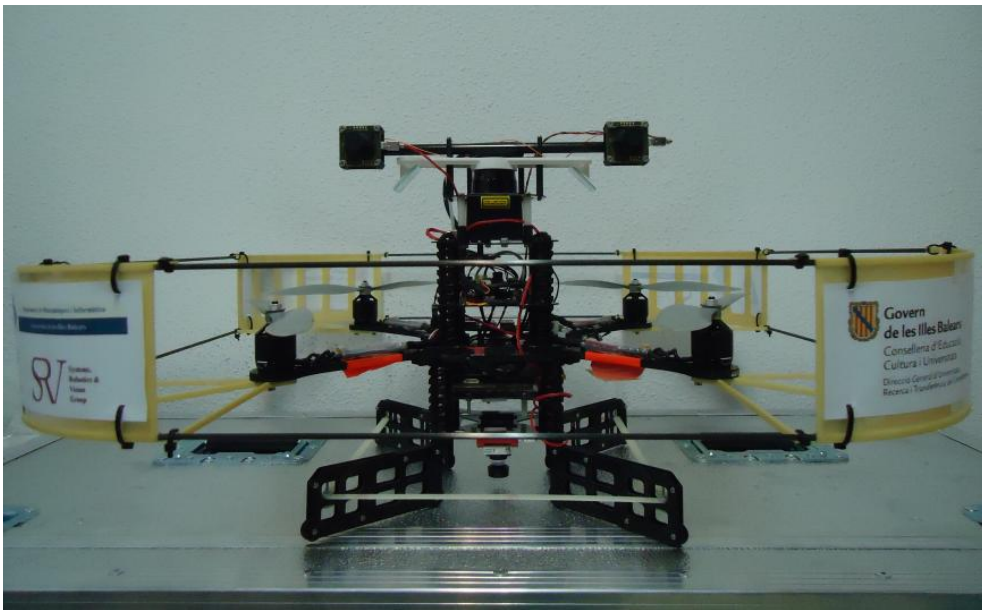

Fig. 2: The MAV prototype in one of its multiple configurations.

Table I: Technical Specifications of the MAV.

Mechanical / Power	
Size	50 × 50 × 20 cm (∅ 53 cm, without propellers protection)
Weight	750 g
Number of propellers	4
Propeller size	25.40 cm (10”)
Payload	500 g
Power	11.1V Lithium Polymer 4500 / 6000 mAh
Computational aspects	
Low-level controller(s)	2 × ARM7
High-level processor	Intel Atom Z530 1.6GHz, 2GB RAM, uSD bootable, detachable USB flash memory for mass storage
Navigation sensors	- General motion: GPS, 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer/compass. - Height estimation: (1) GPS, (2) barometric pressure sensor , (3) laser-based sensor through lateral beam deflection.
Application-oriented sensors	- Flexible structure able to support two lightweight cameras (CMOS, 3.6mm focal length, 752 × 480 pixels) in a number of configurations: forward-looking stereo vision system (13-23 cm baseline), one camera looking to the front and another looking to the ceiling, or a single front-looking camera. - Bottom-looking camera (CMOS, wide angle 2.22mm focal length, 752 × 480 pixels) with either fixed or active pan-tilt mount. - Forward-looking laser scanner (available ranges: 5.60 or 60 m, covering 240º or 270º around the robot).
Interface	- Remote control: 2.4GHz – 7 channels - Command/telemetry channel: XBee Pro ISM 2.4GHz (ZigBee Pro) – 16/10 5MHz channels allowing for 250 kbps - range of operation: 100m (indoors), 1600m (outdoors) - Generic communications: 802.11x WiFi
Capabilities	
Cruise / max / min speed	10 / 15 / 0 m/s
Max. altitude (recommended)	50 m
Flight time (without payload / with maximum payload)	25 min / 12 min (without battery replacement)
Motion control	Attitude and position controllers available. The latter can be fed from either GPS, in GPS-enabled environments, or using internal motion estimates.
High-level motion primitives	Take off Hover Go to waypoint(s) Land
Other capabilities	Still image grabbing

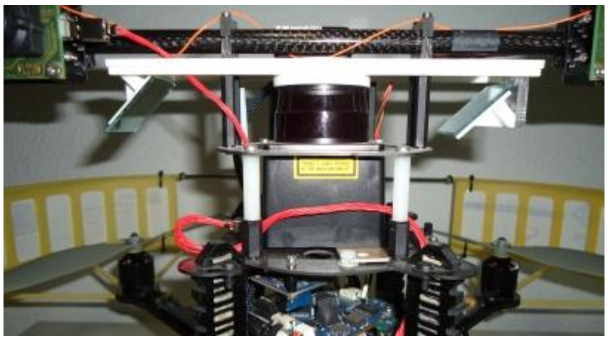

Fig. 3: Laser-based altimeter (left), as well as distance to ceiling estimator (right), by deflecting selected laser beams using mirrors.

3. The control software

3.1 Inspection mission requirements

As already mentioned, the main goal of the MINOAS project is to allow the surveyor to be teleported to the hull parts that require inspection. 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 ─particularly from the vessel areas identified as risky─ so as to allow him to make proper *repair/no repair decisions*.

As part of the inspection process, 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, or even be compared in a posterior inspection.

Hence, the motion capabilities of the MINOAS flying robot comprise waypoint navigation and automatic take-off and landing. Besides, since from one waypoint to the next it is not necessarily ensured there is a line of sight, the vehicle must be able to overcome the obstacles if might find during flight.

3.2 The control architecture

Figure 4 shows the control architecture of the MAV, intended to cope with the control requirements described above. It is expressed in terms of the generic components of the *Generic LOosely-Coupled Component-based Control software architecture* (GLOC3), *Ortiz et al. (2011)*. GLOC3 defines up to a total of five types of components, although all of them are implemented over the basis of a single generic component encapsulating common functionality: *sensor samplers* (SS), which implement the interface with the corresponding physical sensor and sample it at the requested rate; *data processors* (DP), which process the available sensor data and produce the elaborate estimations about the surrounding environment required by the current mission; *controllers* (CC), which, from the data produced by the data processors, generate commands to the vehicle actuators, so that the platform can progress towards achieving the mission goals; *actuator drivers* (AD), which implement the interfaces with the physical actuators; and *command interfaces* (CI), which take care of the communication between a base station (client side) and a platform (server side). Clearly, the first four types correspond to the typical elements of a control system as considered by classical control theory. Consequently, they form a chain along which information flows unidirectionally, from sensor samplers to actuators. Regarding the command interfaces, they not only encapsulate the functionality required to support the distribution of the control software, but also are intended to make easier the interaction with the platform. To this end, they are meant to be implemented around a finite state machine that interprets and responds to a set of well defined commands that directly refer to the vehicle capabilities, i.e. the sort of tasks it is able to carry out.

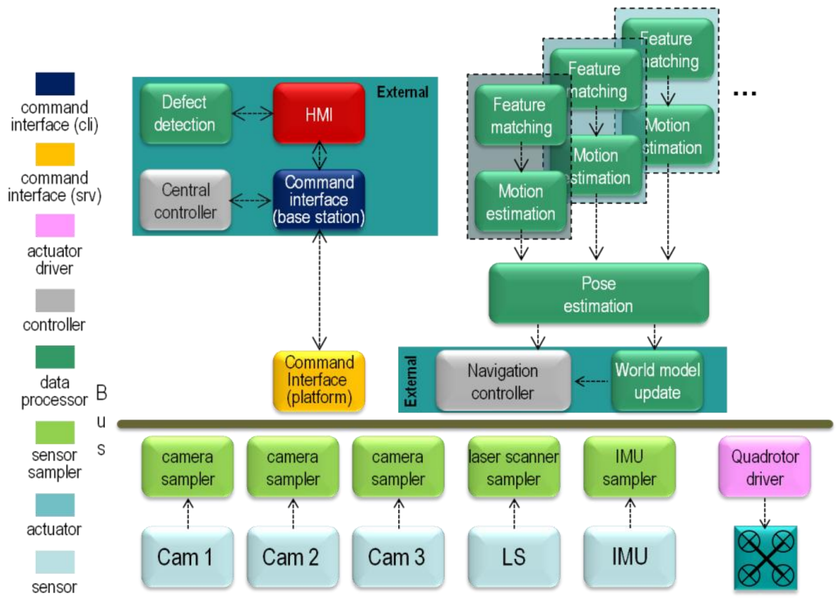

Fig. 4: Control architecture of the MAV expressed in terms of the GLOC3 generic components (*external* means the software runs on a base station)

As can be observed in Figure 4, the control architecture foresees the use of a navigation strategy based on combining, by selection or by fusion, different sources of motion estimation, e.g. a laser scan matching-based odometer, a visual odometer using a front-looking stereo vision system, or a visual odometer fed by a ground-looking camera. Such a redundant strategy is intended to provide robust positioning information able to tolerate the failure of any of the positioning modules when the vehicle motion cannot be estimated from the data provided by the specific sensor, e.g. lack of visual features when imaging non-textured surfaces or lack of distinctive structures within laser scans in long straight walls. The activation of this redundant scheme (involving more or less motion estimation modules) keeps conditional on the computational power available onboard. This is because part of the sensor data processing will have to be placed off-board if the onboard processor(s) are not able to run on time all the calculations required by the different motion estimation algorithms, what means sending data through a wireless data link which, being lossy under normal conditions, experiences larger error transmission rates due to multiple signal reflections caused by the surrounding metallic surfaces. This forces a larger number of re-transmissions, and, thus, increases the communications latency, what can seriously compromise the effectiveness of critical control loops depending on motion estimation, such as the position controller.

Figures 5 and 6 next describe in a more comprehensive way the part of the control architecture related with vehicle navigation. This comprises self-localization, computation of a world model, obstacle avoidance, and the onboard platform control that tries to make it attain the desired state at every time instant. Being a robotic platform with six degrees of freedom, the state in this case consists of a three-dimensional pose comprising (x, y, z) coordinates and the roll (γ), pitch (ϕ) and yaw (ψ) angles.

Although not explicitly stated in the drawings, the control architecture here described allows two modes of operation for the platform: *semi-autonomous* and *autonomous*. While in the latter mode the system just requires a description of the mission to accomplish and it takes care of the mission execution without the intervention of any operator, in semi-autonomous operation, an operator is expected to send velocity commands in x, y and z using the sticks of an R/C transmitter, while the vehicle provides hovering and height control functionalities using onboard sensors and controllers.

3.3 Organization of the control software

As can be observed in Figure 4, the control architecture comprises at least two physically separated agents, the MAV itself and a base/ground station (which in turn can consist of one or several machines), so that the control software is inherently distributed.

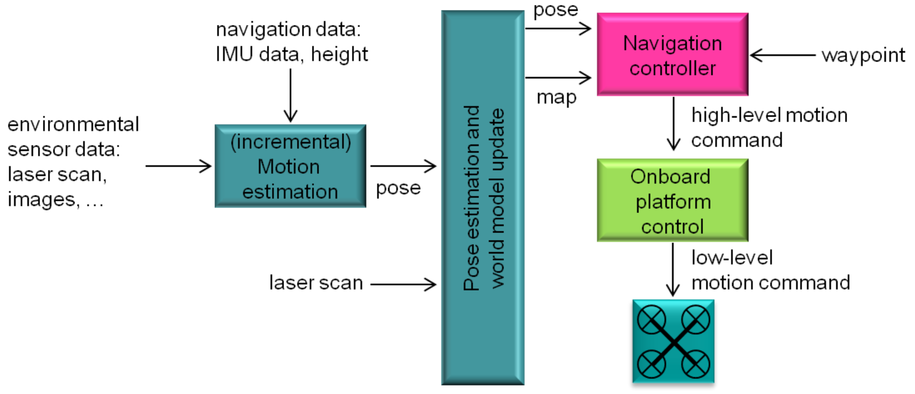
navigation data:
IMU data, height
environmental sensor data: laser scan, images, …
(incremental) Motion estimation
pose
laser scan
Pose estimation and world model update
pose
map
Navigation controller
waypoint
high-level motion command
Onboard platform control
low-level motion command

Fig. 5: Relationship between high-level and low-level (onboard) control for platform navigation.

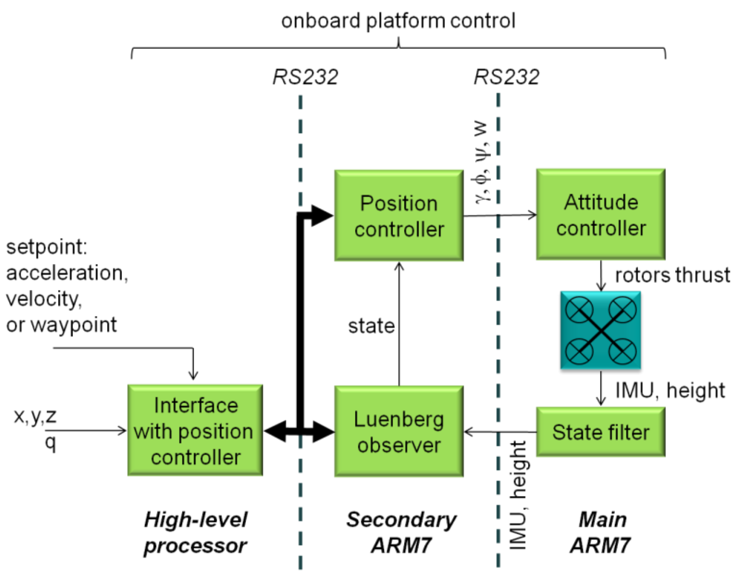

Fig. 6: Onboard platform control components and interface (x, y, z, q is the vehicle current pose, where q stands for a quaternion specifying platform attitude; γ, ϕ, ψ, w are low-level control commands for attaining the set point by means of changes in, respectively, roll, pitch, yaw and thrust).

More specifically, the different computational resources of the MAV run the control algorithms as either firmware or software as detailed next:

- The main ARM7 controller runs the low level software taking care of direct motor control, IMU and air pressure sensors sampling and ZigBee - R/C communications, as well as roll/pitch/yaw stabilization, as described in *Gurdan et al. (2007)*.

- The secondary ARM7 controller runs the position controller described in *Achtelik et al. (2011)*.

- The high-level processor runs the Electric distribution of the Robot Operating System (ROS, *Quigley et al. (2009)*) over Linux Ubuntu 10.10. This processor executes the ROS nodes providing sensor sampling (cameras and laser scanner), platform motion estimation, interaction with the onboard platform controllers and WiFi communication with the base station.

Apart from this, the base station supporting the MAV also runs Linux Ubuntu 10.10 and ROS Electric. As mentioned above, processes which can tolerate (up to a certain extent) latency in the communications are executed on the base station, as it is the case of the world model update, the navigation controller and the user interaction; the others run onboard the vehicle in order to ensure minimum delay.

Under these conditions, ROS becomes particularly relevant as it supplies the middleware functionality for transparent messages exchange between processes running not only in different processors, but also in the same processor, providing optimized mechanisms for reducing the execution times towards real-time operation.

Finally, the control software has been implemented so as to favour as much as possible its modularity, so that adapting the platform for different missions involving different payloads, or activating one or more motion estimation modules, can be performed in a fast and reliable way. In this regard, ROS has proved to be especially useful.

3.4 Mission control

Mission control is a further functionality that runs on the base station, which is in charge of the accomplishment of the mission objectives, provided a specification of the mission exists. Further, this specification has to be performed in terms of the capabilities of the platform, which depends on the control architecture that has been implemented. Regarding the MAV, its control architecture endows it with the capabilities of essentially hovering and attaining three-dimensional waypoints, avoiding obstacles if necessary, as well as automatic take off and landing, since the latter are just particular cases of waypoint achievement.

In our case, a mission is described in an XML file as a sequence of the following actions: (1) *go to*, which makes the vehicle attain a 3D pose (x, y, z, ψ) relative to the home position; (2) *navigate to*, similar to the *go to* action, but ensuring obstacle avoidance; and (3) *take photo*, which requests the system to grab a picture from one of the cameras attached to the vehicle. Clearly, these actions are enough to cover the expected global functionality of the platform as for the implementation of the aforementioned inspection missions.

Just by way of illustration, the mission specification below makes the vehicle take off at a height of 1 meter and hover for 5 seconds (line 2), take a picture and hover for 5 more seconds (line 3), move 2 meters in *x* and hover for 5 seconds (line 4), take a picture and hover for 5 more seconds (line 5), go home and hover for 5 seconds (line 6), and, finally, land (line 7):

```
1   <mission>
2   <goto x="0.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
3   <takephoto camera="1" path="picture1.jpg" stay_time="5.0"/>
4   <goto x="2.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
5   <takephoto camera="1" path="picture2.jpg" stay_time="5.0" />
6   <goto x="0.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
7   <goto x="0.0" y="0.0" z="0.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.3" accori="0.0" timeout="30" />
8   </mission>
```

Every *goto* line above specifies the pose to be achieved (*x*, *y*, *z* and *yaw* fields), the speed to be used while navigating from one waypoint to another (*spx*, *spy* and *spz* fields), the accuracy required in position and orientation so as to consider the waypoint achieved (*accpos* and *accori* fields), a timeout for giving up the waypoint in case it is unreachable (*timeout* field) and the hovering time (*stay_time* field).

4. Navigation capabilities demonstrations

To demonstrate the navigation capabilities of the MAV, this section reports on the execution of several missions. The first mission was performed within the laboratory of the Systems, Robotics and Vision group of the University of Balearic Islands and consists in a sweeping-like task implemented as a zig-zag path over a wall parallel to the XZ plane. At the scale of the laboratory, it would be similar to the kind of mission to be performed by the MAV on a real ship for providing the surveyor with an overall view of the state of a certain area of the vessel hull.

First of all, the mission specification file for that experiment is as follows:

```
<mission>
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="0.0" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="1.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="1.57" accpos="0.1" accori="0.0" timeout="30" stay_time="1.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="3.0" />
<takephoto camera="1" path="photo01.jpg" stay_time="2.0" />
<goto x="0.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo02.jpg" stay_time="2.0" />
<goto x="1.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo03.jpg" stay_time="2.0" />
<goto x="1.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo04.jpg" stay_time="2.0" />
<goto x="2.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo05.jpg" stay_time="2.0" />
<goto x="2.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo06.jpg" stay_time="2.0" />
<goto x="3.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo07.jpg" stay_time="2.0" />
<goto x="3.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<takephoto camera="1" path="photo08.jpg" stay_time="2.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="2.36" accpos="0.1" accori="0.0" timeout="30" stay_time="2.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="1.57" accpos="0.1" accori="0.0" timeout="30" stay_time="1.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="1.0" />
<goto x="0.0" y="0.0" z="0.5" spx="0.2" spy="0.2" spz="0.2" yaw="0.0" accpos="0.1" accori="0.0" timeout="30" stay_time="20.0" />
<goto x="0.0" y="0.0" z="0.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.0" accpos="0.3" accori="0.0" timeout="30" />
</mission>
```

The intended path is as indicated in Figure 7, where waypoints are specified as white numbers over gray background and locations where pictures are taken are indicated as black numbers over green background. More specifically, in this mission, the MAV takes off, reaches a waypoint directly situated above it (0.5 m), hovers for a certain amount of time (2 seconds), changes yaw and starts visiting every waypoint, hovering (for 2 seconds) and taking a picture at each place.

Figures 8 and 9 respectively plot the path estimated by the vehicle and the map built during the flight. Additionally, Table 1 puts together quantitative data for the three stages of the mission, namely take off (from t = 0 until the vehicle reaches 95% of the height corresponding to waypoint 1), navigation and landing (from the time the vehicle is at the 95% of the height for waypoint 1 until getting in contact with the floor). More precisely, the table provides average deviations in X and Y while ascending and descending (in the table, $\delta_X^{(x0)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_0)^2}$ and $\delta_Y^{(y0)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_0)^2}$), as well as range of XYZ motion during navigation, to show the spatial volume covered.

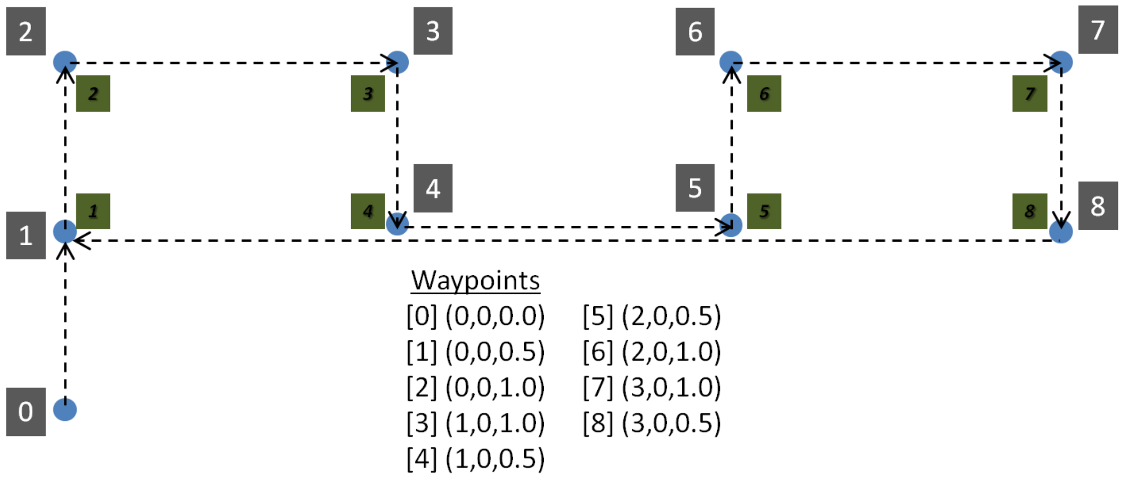

Fig. 7: Waypoints to be attained and pictures to be taken during the sweeping mission.

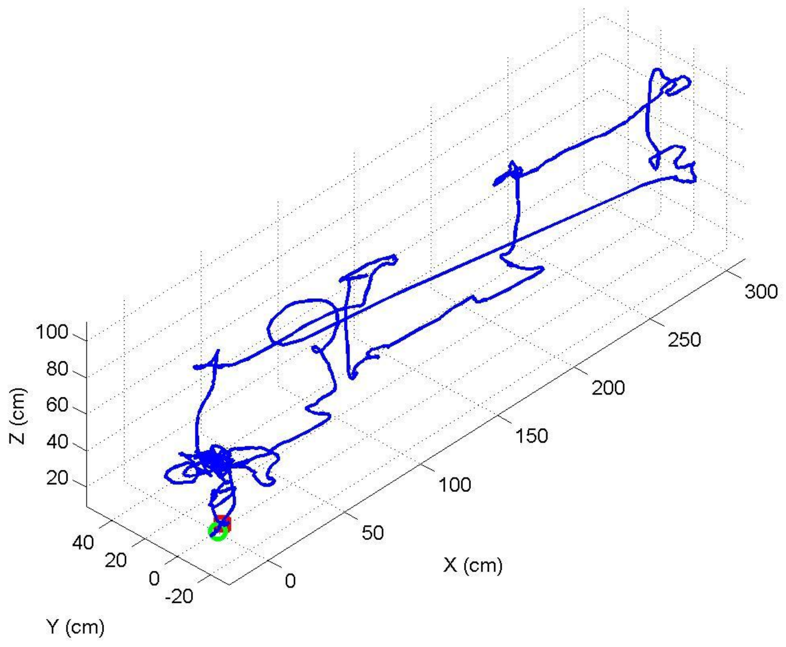

Fig. 8: 3D path estimated by the MAV. (The red square and the green circle correspond, respectively, to the start and the end positions.)

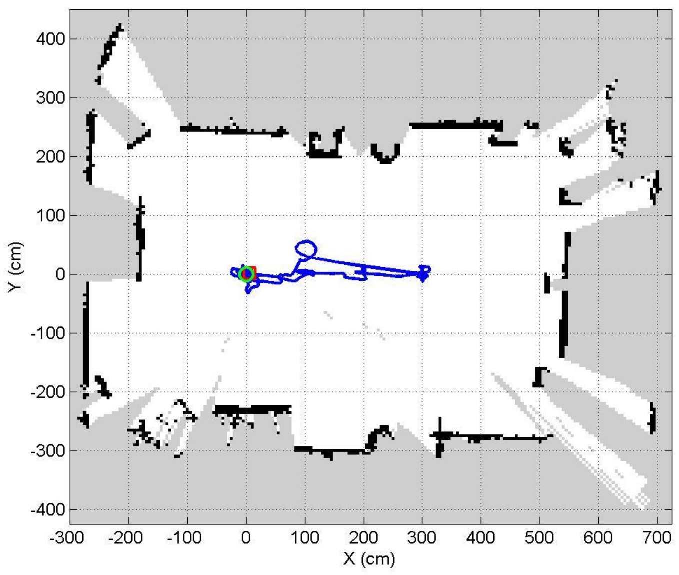

Fig. 9: 2D map built and used by the vehicle during the flight, and the path estimated superimposed. (The red square and the green circle correspond, respectively, to the start and the end positions. In the map, black, white and gray mean, respectively, occupied space, free space and unknown.)

Table 1: Quantitative data for the sweeping task (laboratory experiment).

Total time	171.24 s				
Take-off	**Time**	28.94 s			
	$\delta_X^{(0)}$	6.01 cm	**Range$_X$**	-15.40 cm	+18.99 cm
	$\delta_Y^{(0)}$	2.43 cm	**Range$_Y$**	-4.05 cm	+7.37 cm
Navigation	**Time**	114.62 s			
	Total$_X$	336.91 cm	**Range$_X$**	-25.07 cm	+311.84 cm
	Total$_Y$	87.03 cm	**Range$_Y$**	-31.37 cm	+55.66 cm
	Total$_Z$	70.75 cm	**Range$_Z$**	+40.13 cm	+110.89 cm
Landing	**Time**	27.68 s			
	$\delta_X^{(0)}$	2.63 cm	**Range$_X$**	-3.80 cm	+5.17 cm
	$\delta_Y^{(0)}$	1.33 cm	**Range$_Y$**	-6.85 cm	+0.96 cm
	End point	**X**	+0.63 cm	**Y**	-0.35 cm

As can be observed from the table and the plots, the platform can effectively move from one waypoint to another. Regarding the final position attained by the vehicle it is of the order of 1 cm away from the final waypoint (0, 0). Although it is true that, in this case, the same instrument is used for self-localization and performance assessment, the match between the path estimated and the environment, together with the visual observation of a reduced deviation between the initial and the final positions, are favorable clues about the good performance of the platform motion estimation capability.

Figures 10 and 11 provide, respectively, the pictures taken for the previous experiment and for a second execution of the same mission. Notice that the similarity of the images taken for the same waypoint in both executions can be used as a metric to assess the performance of the MAV when a place has to be re-visited. In this particular case, slight deviations in the point of view for the same waypoint are observed. Notice, however, that such a minimum difference is likely to happen due to the pose tolerance indicated in the mission specification file.

To finish, Figure 12 plots results for also a sweeping-like task, but this time performed onboard a vessel during a field trials campaign that took place in Dolphin PLC shipyard facilities (Varna, Bulgaria). Field testing was performed inside the front cargo hold of a 16,800 DWT general cargo ship using the semi-autonomous mode of operation. The picture in Figure 12 (upper,left) allows realizing the details of the testing area, which comprised approximately 30 m × 18 m, and around 12 m from the floor of the hold to the deck.

Table 2 puts together quantitative data for the three stages of the experiment as before. As can be observed from the data supplied, height control effectively keeps the different heights attained along the almost 12 meters covered during the flight. Since the semi-autonomous mode of operation was activated, when travelling from one point to another, the path finally obtained depends on the ability of the operator for sending the appropriate velocity commands using the R/C sticks. Nevertheless, the height control and the hovering assistance provided by the system in this mode simplifies significantly the control of the platform through the R/C transmitter, and makes it possible to fly the platform indoors, along a rather long path relatively close to the walls while capturing good quality pictures.

5. 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 has been described and discussed. Results for a number of experiments performed under laboratory conditions and also onboard a vessel have as well been reported.

Acknowledgements

The authors are very thankful to Kontron and Quality By Measurement (QBM) for supplying the pITX-SP board. This work is partially supported by project SCP8-GA-2009-233715 (EU FP7 MINOAS).

References

ACHTELIK, M.; ACHTELIK, M.; WEISS, S.; SIEGWART, R. (2011), *Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments*, IEEE International Conference on Robotics and Automation, pp. 3056 - 3063.

GURDAN, D.; STUMPF, J.; ACHTELIK, M.; DOTH, K.-M.; HIRZINGER, G.; RUS, D. (2007), *Energy-efficient autonomous four-rotor flying robot controlled at 1 khz*, IEEE International Conference on Robotics and Automation, pp. 361–366.

ORTIZ, A.; BONNIN-PASCUAL, F.; GARCIA-FIDALGO, E.; BELTRAN, J.P. (2011), *A control software architecture for autonomous unmanned vehicles inspired in generic components*, IEEE Mediterranean Conference on Control and Automation, pp. 1217–1222.

ORTIZ, A.; BONNIN, F.; GIBBINS, A.; APOSTOLOPOULOU, P.; BATEMAN, W.; EICH, M.; SPADONI, F.; CACCIA, M.; DRIKOS, L. (2010), *First steps towards a roboticized visual inspection system for vessels,* IEEE Conference on Emerging Technologies and Factory Automation.

POUNDS, P.; MAHONY, R.; CORKE, P. (2010), *Modelling and control of a large quadrotor robot,* J. Control Engineering Practice 18/7, pp. 691–699.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.P.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A.Y. (2009), *ROS: an open-source Robot Operating System*, IEEE International Conference on Robotics and Automation, Workshop on Open Source Software.

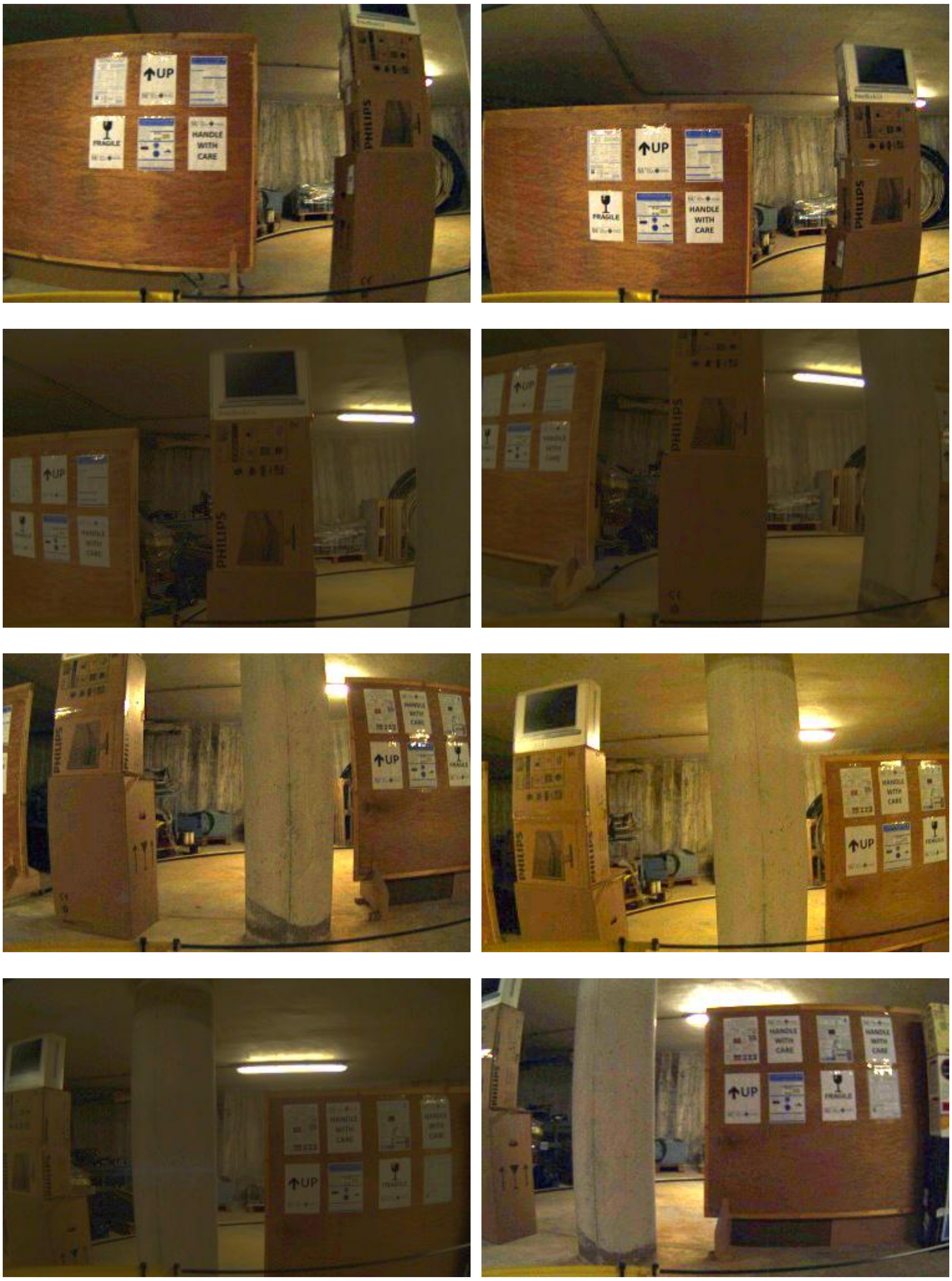

Fig. 10: Sweeping task under laboratory conditions: Pictures taken during the first flight.

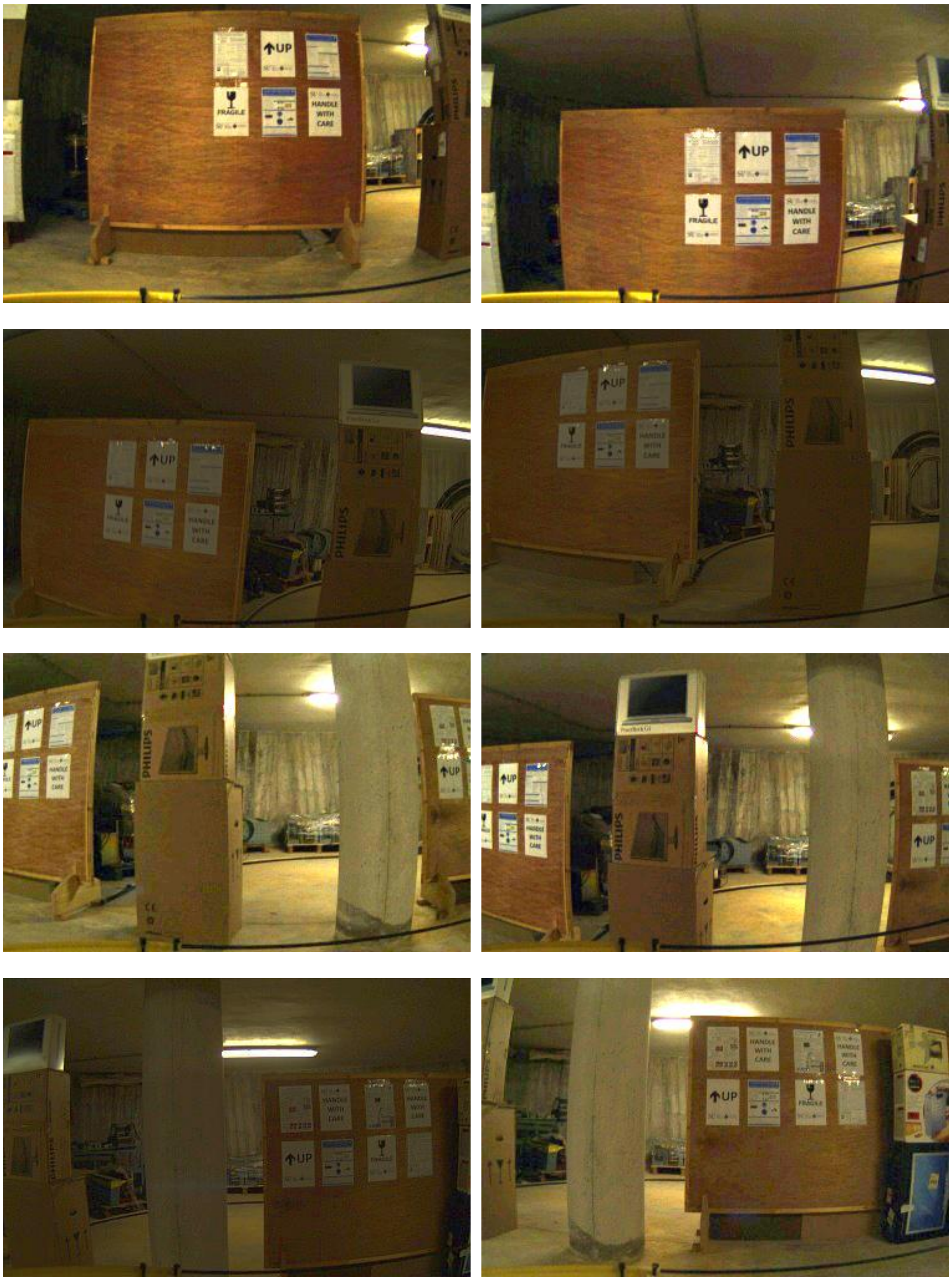

Fig. 11: Sweeping task under laboratory conditions: Pictures taken during the second flight.

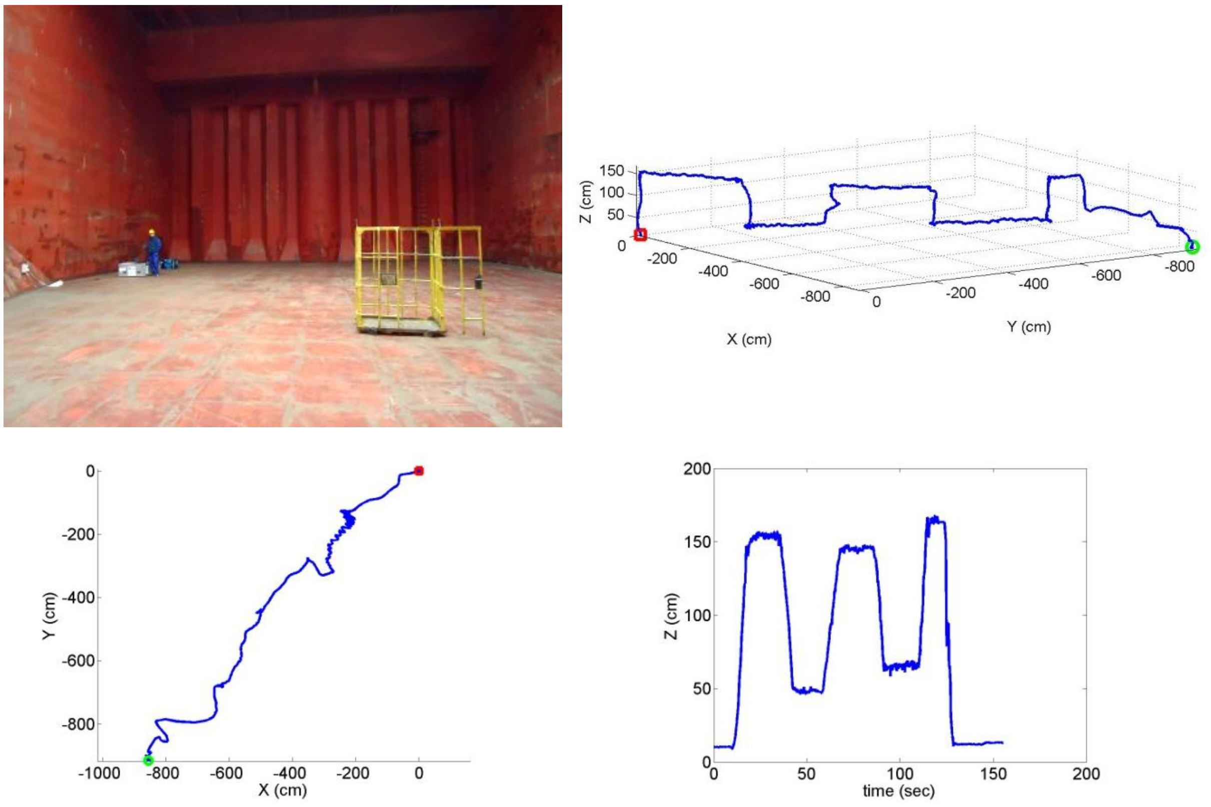

Fig. 12: Path followed by the MAV during a sweeping task performed over the front wall of the forward hold of a general cargo vessel [upper-left corner picture]. (The red square and the green circle correspond, respectively, to the start and the end positions.)

Table 2: Quantitative data for the sweeping task (field experiment, (x_T,y_T) and (x_L,y_L) are the 2D coordinates of, respectively, the take-off and landing points).

Total time	155.22 s				
Take-off	**Time**	13.54 s			
	$\delta_X^{(xT)}$	2.20 cm	**Range$_X$**	-0.43 cm	+8.48 cm
	$\delta_Y^{(yT)}$	1.52 cm	**Range$_Y$**	-3.89 cm	+5.66 cm
Navigation	**Time**	121.86 s			
	Total$_X$	865.92 cm	**Range$_X$**	-857.28 cm	+8.64 cm
	Total$_Y$	891.92 cm	**Range$_Y$**	-888.54 cm	+3.39 cm
	Total$_Z$	121.44 cm	**Range$_Z$**	+46.27 cm	+167.71 cm
Landing	**Time**	28.24 s			
	$\delta_X^{(xL)}$	8.64 cm	**Range$_X$**	-863.69 cm	-845.33 cm
	$\delta_Y^{(yL)}$	25.18 cm	**Range$_Y$**	-917.60 cm	-888.54 cm