Vessel Inspection: A Micro-Aerial Vehicle-based Approach

**Alberto Ortiz · Francisco Bonnin-Pascual ·
Emilio Garcia-Fidalgo**

Abstract Vessel maintenance entails periodic visual inspections of the internal and external parts of the hull in order to detect the typical defective situations affecting metallic structures, such as coating breakdown, corrosion, cracks, etc. The main goal of 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 an approach to the inspection problem based on an autonomous *Micro Aerial Vehicle* (MAV) which, as part of this fleet, is in charge of regularly supplying images that can teleport the surveyor from a base station to the areas of the hull to be inspected. The control software approach adopted for the MAV is fully described, with a special emphasis on the self-localization capabilities of the vehicle. Experimental results showing the suitability of the platform to the application are as well reported and discussed.

Keywords MAV · Visual odometry ·
Visual inspection

A. Ortiz (✉) · F. Bonnin-Pascual · E. Garcia-Fidalgo
Department of Mathematics and Computer Science,
University of Balearic Islands, Cra. Valldemossa
km 7.5, 07122 Palma de Mallorca, Spain
e-mail: alberto.ortiz@uib.es

F. Bonnin-Pascual
e-mail: xisco.bonnin@uib.es

E. Garcia-Fidalgo
e-mail: emilio.garcia@uib.es

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 *Classification Societies*, who are continually seeking to improve standards and reduce the risk of maritime accidents. However, despite the efforts on reducing them, they still occur and, from time to time, have catastrophic consequences in personal, environmental and/or financial terms. Structural failures are a major cause of accidents, and can usually be prevented through timely maintenance. Consequently, vessels undergo annual inspections, with intensive *Special* and *Docking Surveys* every five years, which ensure that the hull structure and related piping are all in satisfactory condition and are fit for the intended use over the next five years.

Fig. 1 (*left*) Staging required during a vessel inspection. (*right*) Oil tanker in shipyard during construction
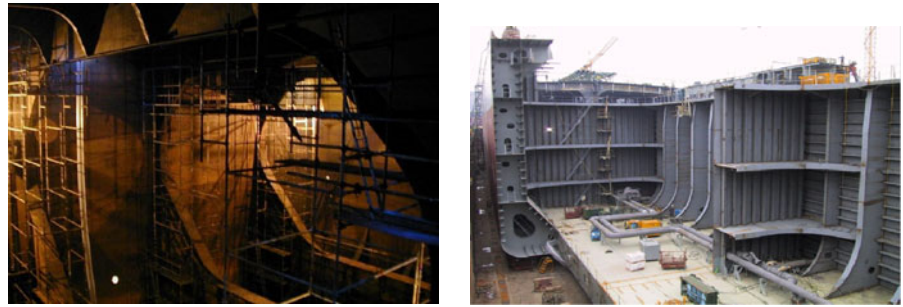
An important part of the vessel maintenance has to do with the visual inspection of the external and internal parts of the vessel hull. They can be affected by different kinds of defects typical of steel surfaces and structures, such as coating breakdown and/or corrosion, and cracks. These defects are indicators of the state of the metallic surface and, as such, an early detection prevents the structure from buckling and/or fracturing.

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 860 m of web frames (primary stiffening members) and approximately 3.2 km of longitudinal stiffeners. Furthermore, this surveying is performed in a potentially hazardous environment with both flammable and toxic gases and significant heights involved. Due to these complications, the total cost of a single surveying can exceed $1 M once you factor in the vessel's preparation, use of yard's facilities, cleaning, ventilation, and provision of access arrangements (see Fig. 1(left)). In addition, the owners experience significant lost opportunity costs while the ship is inoperable.

The main objective of the EU-funded 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 [27]

for a detailed discussion). Contrary to similar past projects (ROTIS and its follow-up ROTIS-II [32]) or commercial solutions such as [26], the scope of MINOAS comprises both dry and wet areas of the vessel, and not only flooded ballast tanks or the external hull. The MINOAS project is neither limited to tele-operated floating tethered vehicles, but considers a varied set of robotic technologies with different locomotion capabilities, including *magnetic crawlers*, *remotely operated vehicles* (ROV) and *unmanned aerial vehicles* (UAV).

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. Due to its inherent properties for flying indoors and close to other structures, a quadrotor has resulted in the platform of election for this application. The MAV is described in this paper, although the focus is mostly set on its control architecture and specially on its self-localization capabilities. As it is detailed later, self-localization is achieved by means of a proper combination of vision- and laser-based motion estimation. In this regard, an *almost-closed-form* solution to visual odometry is contributed as part of the development of the navigation strategy.

The rest of the paper is structured as follows: Section 2 discusses on the requirements imposed by the application and reviews related work; Sections 3 and 4 describe, respectively, the platform and the control architecture, focusing on self-localization issues; Section 5 describes a monocular odometer to be used as part of the localization strategy; Section 6 provides experimental results showing the platform performance; and, finally, Section 7 concludes the paper.

---

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

2 Background

2.1 Inspection Problem and Requirements

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—i.e. to the reach of a hand—of the different metallic surfaces and structures. For those ships where there is a real cost saving, i.e. the inspection is likely to result in no repair, so that the preparation of the vessel for a human inspection with a non-subsequent repair is less justified (see [27] for a deeper analysis), the MINOAS concept involves the use of different robotic platforms through a series of stages that altogether can replace the *in-situ* human inspection.

Among others, the vertical structures that can be found in vessel holds are of prime importance (see Fig. 1). To make proper *repair/no repair decisions*, the surveyor must be provided with, among others, imagery detailed enough so as to enable the remote visual assessment of these structures. The MINOAS aerial platform is precisely intended to provide this kind of data by implementing the first stage of the inspection mission. In this stage, the platform sweeps the relevant metallic surfaces and grabs pictures at a rate compatible with its speed, in order to provide an overall view of their 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, and/or taking thickness measurements (by means of other platforms of the robot fleet).

Therefore, the main requirements for the aerial platform 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 of manoeuvres such as hovering and *vertical take-off and landing* (VTOL). Besides being a VTOL vehicle, the inspection mission requires from the platform, apart from automatic take-off and landing, flying through waypoints and obstacle avoidance (i.e. a full navigation solution), since, from one waypoint to the next, it is not necessarily ensured there is a line of sight, and thus the vehicle must be able to overcome the obstacles it might find during the flight. Finally, the platform should not rely on GPS data for positioning because it could be required to operate indoors or in poor GPS reception areas.

2.2 Related Work

Among the different kinds of helicopter designs that have been proposed, multi-rotor configurations present several advantages over comparably scale helicopters [9]: (1) they do not require mechanical linkages to vary the 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. This kind of UAV presents also some disadvantages deriving from the multi-rotor configuration, namely a larger weight and larger energy consumption due to the extra motors. However, all in all, their particular suitability for indoor flights and interaction in close proximity to other objects (and also in obstacle-dense environments), with low risk of damaging the vehicle, its operators, or its surroundings, seem to overcome the aforementioned disadvantages.

Among other multi-rotor UAVs, the four-rotor, or *quadrotor*, is emerging as the most popular multi-rotor configuration. 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. Finally, counter rotating propellers increase efficiency and flight times, as no extra thrust is needed to compensate for unwanted rotation.

Lately, a number of navigation solutions comprising platform stabilization, self-localization, mapping and obstacle avoidance have been proposed for small-size multi-rotors, a.k.a. *Micro Aerial Vehicles* (MAV). They mainly differ in the sensor 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 extensively used due to its accuracy and speed. For instance, [4, 11, 16, 18] propose full navigation systems using laser scan matching and IMU fusion for motion estimation embedded within a SLAM framework. Among them, [4, 11] describe multi-level approaches.

Infrared or ultrasounds are other possible sensors to be used for navigation tasks. Although they present less noise tolerance and accuracy, several researchers [7, 24, 30] have used them to perform navigation tasks in indoor environments, being a cheaper option than laser scanners.

Vision-based navigation has become quite popular for MAVs lately. Cameras' success in general robotics come mainly from the richness of the sensor data supplied, combined with low weight, low power designs and a relatively low price. Nevertheless, the associated higher computational cost has made MAV researchers find optimized solutions that can run over low-power processors. Among the most recent works published in this regard, some propose visual SLAM solutions based on feature tracking, either adopting a frontal mono or stereo camera configuration, e.g. [13, 14], or choosing a ground-looking orientation, e.g. [3, 10]. Others focus on efficient implementations of optical flow calculations, either dense or sparse, and mostly from ground-looking cameras (e.g. [19, 35]), or develop methods for landing, tracking and taking off using passive (e.g. [12, 25]) or active markers (e.g. [21, 34]), also using a ground-looking camera.

Among the different tasks to solve and implement, this paper contributes with a platform motion estimation and self-localization strategy adapted to the application at hand (obstacle avoidance issues are not addressed). In this regard, since the vertical structures that are found in vessel holds do not vary in shape in a continuous manner but are essentially the same along a large part of their vertical extent and only change occasionally (see Fig. 1), a full 3D mapping and navigation solution has not been deemed a critical requirement. Instead, a laser/vision combination has been adopted: horizontal motion is estimated by means of laser scan-matching, while, during vertical displacements, the vehicle makes use of a ground-looking camera and a novel visual odometer based on image feature tracking. This combination, together with the closed-form formulation of the visual odometer, has resulted in a self-localization strategy that is compatible with the computational resources onboard. This, in turn, has allowed reducing the use of wireless datalinks, which are not favoured within vessel holds due to the surrounding metallic structures.

3 Platform Description

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 650 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 features a lightweight laser scanner. Figure 2 shows the MAV carrying a Hokuyo URG-UTM-30LX—up to 30 m range. As can be seen, 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. This method has been found more adequate for

Fig. 2 The micro aerial vehicle

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 ground-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 upwards, or, to save weight, a single camera facing forward. Figure 2 shows the second configuration, comprising two uEye 1226-LE-C cameras, and a third uEye 1226-LE-C fitted with a wide-angle lens 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. A further analog video camera operating at 5.8 GHz is also attached to provide live video during flight.

Finally, the vehicle carries an additional processing board which avoids sending sensor data to a base station, but process it onboard avoiding 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. 2 includes a CoreExpress board equipped with an Intel Atom 1.6 GHz processor and 1 GB RAM).
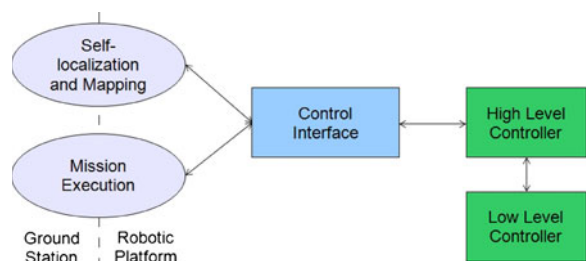
4 Control Software

The control software architecture comprises at least two physically separated agents: the MAV itself and a base/ground station. The different computational resources of the MAV run the control algorithms (either as firmware or as software) that are detailed next: (1) the main ARM7 controller essentially runs the low-level software taking care of attitude stabilization and direct motor control [17] (in Fig. 3, it appears as the low-level controller); (2) the secondary ARM7 controller runs the position controller described in [1] (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) running over Linux Ubuntu, ROS nodes providing sensor sampling and platform motion estimation as well as platform safety, interaction with the onboard platform controllers and WiFi communication with the base station.

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 irrespective of whether they run on the same or in different machines.

Those processes that can tolerate communications latency are executed on the base station, while critical control loops run onboard the vehicle in order to ensure minimum delay, a requirement also reported by other authors [2] to permit autonomous flying.

As well as the self-localization and mapping solution described in [11], our MAV control

Fig. 3 Control architecture overview

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 depicted in Fig. 3.

4.1 Self-Localization and Mapping

Figure 4 depicts the pose estimation subsystem. 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 subsystem, 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) splitting the laser scans into several fragments, so that beams reflected by the lateral mirrors are separated from beams providing information on the environment structure ahead; and (3) projecting the scans comprising the surviving beams onto the ground, using the attitude information provided by the IMU.

The *Vertical Position* module estimates the distance of the robot to the ground and to the ceiling. It uses, respectively, 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 *Laser* and *Vision Odometer* components contribute to estimating the MAV 3D pose depending on the operation mode: horizontal motion (HM) and vertical motion (VM). In the HM mode, the vehicle moves over a plane parallel to the ground, while, in the VM mode, the vehicle ascends/descends along a vertical path to the desired height. The former mode makes use of laser scans and a 2D map to estimate the vehicle displacement, by means of laser-based odometry and laser-based *Simultaneous Localization and Mapping* (SLAM). The second mode activates a ground-looking vision-based odometer to keep the vehicle within a vertical column during the motion without the use of any map. This is a simple procedure that avoids running a computationally intensive 3D mapping and navigation solution. Notice also that the laser-based odometer and the vision-based odometer do not need to run simultaneously, what permits saving computational resources of the MAV's high-level processor.

Regarding the laser odometer, projected laser scans are supplied to a scan matcher, which

Fig. 4 Self-localization and mapping
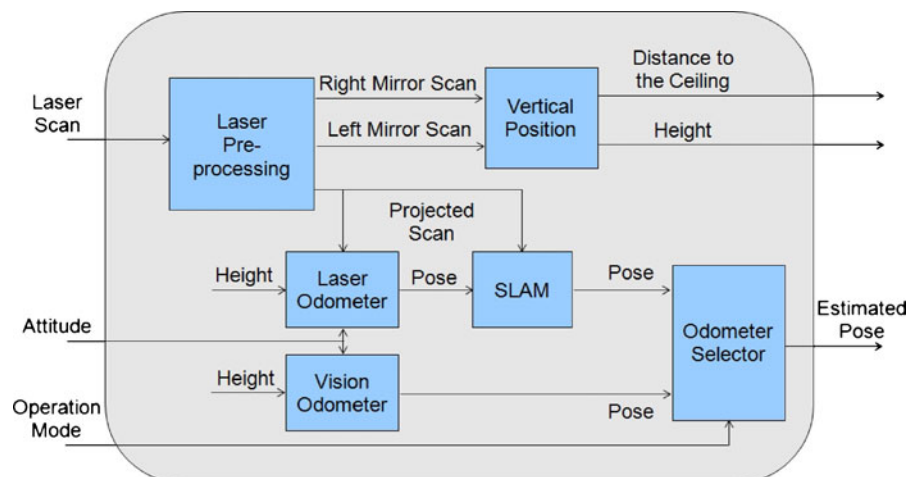
computes the platform roto-translation between consecutive scans and estimates a new 2D pose (x, y, ψ), using the yaw angle provided by the IMU as initial estimate for ψ. The 2D pose so obtained is then combined with the height and the roll and pitch angles (ϕ, θ), provided by, respectively, the laser altimeter and the IMU, to obtain a 3D pose for the vehicle.

In order to compensate the drift in the estimations produced by the scan matcher, a SLAM 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 in a too aggressive way within close 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 [15], provides the SLAM functionality.

About the vision-based odometer, it estimates the vehicle roto-translation by matching visual features from one frame to the next. Full details are given in Section 5.

4.2 Mission Execution

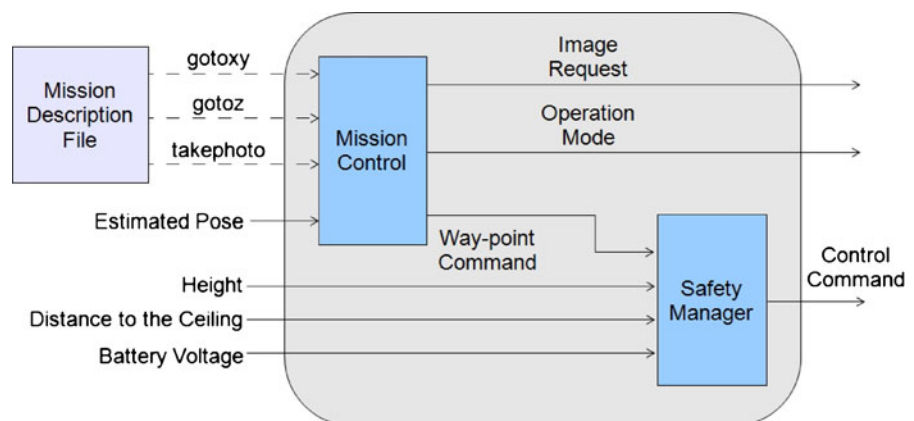This module is responsible for the execution of the different actions which an inspection mission can consist of. Every action can refer to either sensor sampling or to platform motion, as required at every time instant. The sequence of actions is specified in a mission specification file. See Fig. 5 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 the following actions: (1) *gotoxy*, which specifies a 2D pose to be attained by the vehicle, together with maximum speeds and thresholds for considering the motion primitive as accomplished; (2) *gotoz*, which specifies the height to be attained by the vehicle, together with maximum speeds and action-accomplishment thresholds; and (3) *take-photo*, which requests for a picture to be taken by the robot using one of the attached cameras.

Just by way of illustration, the mission specification shown in Fig. 6 makes the vehicle take off at a height of 1 m and hover for 5 s (line 2), take a picture and hover for 5 more sec (line 3), move 2 m in x and hover for 5 s (line 4), take a picture and hover for 5 more s (line 5), go home and hover for 5 s (line 6), and, finally, land (line 7).

A client process, 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 *goto* and *take-photo* actions. Actions of the first kind are pre-processed by the *Safety Manager*

Fig. 5 Mission execution

```
1    <mission>
2    <gotoz z="1.0" 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    <gotoxy x="2.0" y="0.0" spx="0.2" spy="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    <gotoxy x="0.0" y="0.0" spx="0.2" spy="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
7    <gotoz z="0.0" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" />
8    </mission>
```

Fig. 6 Example of a mission specification file

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.

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.

5 An Almost-closed-form Solution to Visual Odometry

This section describes a fast vision-based odometer for camera motion estimation using local image features. It allows estimating motion in x (Δx), y (Δy) and yaw ($\Delta \psi$). In short, once a set of matchings between features of consecutive frames is available, the algorithm backprojects features into the ground using previous knowledge of the camera height t_z, and the roll ϕ and pitch θ angles, all three supplied by onboard sensors. As will be seen, the odometer can be posed as a linear least squares problem, from which two closed-form solutions can be derived: the first one estimates $\cos \Delta \psi$ and $\sin \Delta \psi$ assuming nothing about $\Delta \psi$, while the second one estimates $\Delta \psi$ directly, although requires $\Delta \psi$ to be small. The former solution is developed in this paper, while the latter can be found in [28]. A further step refines the obtained solution in a non-linear iterative way, reason by which the whole algorithm is regarded

as an *almost-closed-form* solution to visual odometry.

5.1 Derivation of the Odometer in Closed-Form

Assuming *right-handed counter-clockwise frames* for both the vehicle and the camera, Eq. 1 transforms from world (x, y, z) to image coordinates (x_p, y_p): (In the following, $c\alpha$ and $s\alpha$ stand for, respectively, $\cos \alpha$ and $\sin \alpha$.)

$$
\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}}^{P} \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\phi & s\phi & 0 \\ 0 & -s\phi & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta & 0 \\ 0 & 1 & 0 & 0 \\ s\theta & 0 & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{T_{Wb}^R}
$$
$$
\underbrace{\begin{bmatrix} c\psi & s\psi & 0 & 0 \\ -s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T_{Wa}^R} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$
(1)

The corresponding inverse perspective transformation is thus:

$$
\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underbrace{\left(T_{Wa}^R \right)^{-1}}_{M^*} \underbrace{\left(T_{Wb}^R \right)^{-1} P^{-1}}_{M} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}
$$
$$
= \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}
$$
(2)

Equation 3 is obtained next by developing Eq. 2:

$$
\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\times \begin{bmatrix} M_1\,x_p + M_2\,y_p + M_3\,z_p + M_4 \\ M_5\,x_p + M_6\,y_p + M_7\,z_p + M_8 \\ M_9\,x_p + M_{10}\,y_p + M_{11}\,z_p + M_{12} \\ \dfrac{f - z_p}{f} \end{bmatrix}
$$

$$
= \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a\,c\psi - b\,s\psi + d\,t_x \\ b\,c\psi + a\,s\psi + d\,t_y \\ c + d\,t_z \\ d \end{bmatrix}
$$

$$(3)$$

Dividing the expressions resulting for the first to the third components by the fourth component (in Eq. 3), to pass from homogeneous to Cartesian coordinates, Eq. 4 results:

$$
v = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \frac{f}{f - z_p} \begin{bmatrix} a\,c\psi - b\,s\psi \\ b\,c\psi + a\,s\psi \\ c \end{bmatrix} \quad (4)
$$

Equation 4 involves coordinate z_p, which turns out to be a free parameter that determines the plane where the 2D point (x_p, y_p) is located, or,

in other words, the backprojection of the image point (x_p, y_p), so that $z_p = -\infty$ corresponds to the lens center point v_l, and $z_p = 0$ is for a point v_p situated over the image plane. Using points v_l and v_p, Eq. 4 can be put in a more useful form:

$$
v = v_l + \lambda(v_p - v_l) \quad (5)
$$

Taking into account that:

$$
\lim_{z_p \to -\infty} \frac{f}{f - z_p} = 0
$$

$$
\lim_{z_p \to -\infty} \frac{f}{f - z_p}\,z_p = \lim_{z_p \to -\infty} \frac{f}{\frac{f}{z_p} - 1} = -f
$$

v_l and v_p can be obtained evaluating Eq. 4 for, respectively, $z_p = -\infty$ and $z_p = 0$:

$$
v_l = \lim_{z_p \to -\infty} v = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - f \begin{bmatrix} M_3\,c\psi - M_7\,s\psi \\ M_7\,c\psi + M_3\,s\psi \\ M_{11} \end{bmatrix}
$$

$$
v_p = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}
$$

$$
+ \begin{bmatrix} (M_1 x_p + M_2 y_p + M_4)c\psi - (M_5 x_p + M_6 y_p + M_8)s\psi \\ (M_5 x_p + M_6 y_p + M_8)c\psi + (M_1 x_p + M_2 y_p + M_4)s\psi \\ M_9\,x_p + M_{10}\,y_p + M_{12} \end{bmatrix}
$$

Consequently, Eq. 5 becomes into the following 3D line in world coordinates:

$$
\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} -M_3\,f\,c\psi + M_7\,f\,s\psi \\ -M_7\,f\,c\psi - M_3\,f\,s\psi \\ -M_{11}\,f \end{bmatrix}
$$

$$
+ \lambda \cdot \begin{bmatrix} (M_1\,x_p + M_2\,y_p + M_3\,f + M_4)c\psi - (M_5\,x_p + M_6\,y_p + M_7\,f + M_8)s\psi \\ (M_5\,x_p + M_6\,y_p + M_7\,f + M_8)c\psi + (M_1\,x_p + M_2\,y_p + M_3\,f + M_4)s\psi \\ M_9\,x_p + M_{10}\,y_p + M_{11}\,f + M_{12} \end{bmatrix} \quad (6)
$$

The previous line corresponds to the ray back-projecting an image point (x_p, y_p) for a camera located at world coordinates (t_x, t_y, t_z) and oriented ψ in yaw, θ in pitch and ϕ in roll.

Notice now that, for image features corresponding to points on the floor, $z = 0$ and, consequently:

$$
\lambda = -\frac{t_z - M_{11}\,f}{M_9\,x_p + M_{10}\,y_p + M_{11}\,f + M_{12}} \quad (7)
$$

Using Eq. 7, Eq. 6 can be written as follows for time k:

$$
\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}^{(k)} + \begin{bmatrix} C + \lambda\,A & -(D + \lambda\,B) \\ D + \lambda\,B & (C + \lambda\,A) \end{bmatrix}^{(k)} \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} \quad (8)
$$

with:

$$
\lambda^{(k)} = -\frac{t_z^{(k)} - M_{11}^{(k)}\,f}{M_9^{(k)}\,x_p + M_{10}^{(k)}\,y_p + M_{11}^{(k)}\,f + M_{12}^{(k)}}
$$

$$A^{(k)} = M_1^{(k)} x_p + M_2^{(k)} y_p + M_3^{(k)} f + M_4^{(k)}$$

$$B^{(k)} = M_5^{(k)} x_p + M_6^{(k)} y_p + M_7^{(k)} f + M_8^{(k)}$$

$$C^{(k)} = -M_3^{(k)} f$$

$$D^{(k)} = -M_7^{(k)} f$$

Defining $\alpha^{(k)} = C^{(k)} + \lambda^{(k)} A^{(k)}$ and $\beta^{(k)} = D^{(k)} + \lambda^{(k)} B^{(k)}$, Eq. 8 can be written in a more compact form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}^{(k)} + \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}^{(k)} \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} \tag{9}$$

At time $k + 1$, if the vehicle has rotated in yaw $\Delta\psi$ and has moved at $(t_x^{(k)} + \Delta t_x, t_y^{(k)} + \Delta t_y, t_z^{(k+1)})$, and changed from pitch $\theta^{(k)}$ to $\theta^{(k+1)}$ and from roll $\phi^{(k)}$ to $\phi^{(k+1)}$, for the same scene point at world coordinates $(x, y, z = 0)$, we have:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x^{(k)} + \Delta t_x \\ t_y^{(k)} + \Delta t_y \end{bmatrix} + \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}^{(k+1)}$$

$$\times \begin{bmatrix} c\psi^{(k)} c\Delta\psi - s\psi^{(k)} s\Delta\psi \\ s\psi^{(k)} c\Delta\psi + c\psi^{(k)} s\Delta\psi \end{bmatrix} = \begin{bmatrix} t_x^{(k)} + \Delta t_x \\ t_y^{(k)} + \Delta t_y \end{bmatrix}$$

$$+ \begin{bmatrix} \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} & -\beta^{(k+1)} c\psi^{(k)} - \alpha^{(k+1)} s\psi^{(k)} \\ \beta^{(k+1)} c\psi^{(k)} + \alpha^{(k+1)} s\psi^{(k)} & \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \end{bmatrix}$$

$$\times \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix} \tag{10}$$

Equating 9 and 10 we can write:

$$\begin{bmatrix} \alpha c\psi - \beta s\psi \\ \beta c\psi + \alpha s\psi \end{bmatrix}^{(k)} = \begin{bmatrix} \Delta t_x \\ \Delta t_y \end{bmatrix}$$

$$+ \begin{bmatrix} \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} & -\beta^{(k+1)} c\psi^{(k)} - \alpha^{(k+1)} s\psi^{(k)} \\ \beta^{(k+1)} c\psi^{(k)} + \alpha^{(k+1)} s\psi^{(k)} & \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \end{bmatrix}$$

$$\times \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix} \tag{11}$$

Equation 12 can next be obtained rearranging Eq. 11:

$$\begin{bmatrix} 1 & 0 & \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} & -\beta^{(k+1)} c\psi^{(k)} - \alpha^{(k+1)} s\psi^{(k)} \\ 0 & 1 & \beta^{(k+1)} c\psi^{(k)} + \alpha^{(k+1)} s\psi^{(k)} & \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \end{bmatrix}$$

$$\times \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \alpha c\psi - \beta s\psi \\ \beta c\psi + \alpha s\psi \end{bmatrix}^{(k)} \tag{12}$$

If t_z, ϕ and θ are available at times k and $k + 1$ (from onboard sensors or derived from image data), then a least squares framework can be developed from Eq. 12 to determine Δt_x, Δt_y, $c\Delta\psi$ and $s\Delta\psi$ if, at least for two scene points $(x_i, y_i, z_i = 0)$, their image points counterparts $(x_{p,i}, y_{p,i})$ have been matched between frames k and $k + 1$:

$$\begin{bmatrix} 1 & 0 & \alpha_1^{(k+1)} c\psi^{(k)} - \beta_1^{(k+1)} s\psi^{(k)} & -\beta_1^{(k+1)} c\psi^{(k)} - \alpha_1^{(k+1)} s\psi^{(k)} \\ 0 & 1 & \beta_1^{(k+1)} c\psi^{(k)} + \alpha_1^{(k+1)} s\psi^{(k)} & \alpha_1^{(k+1)} c\psi^{(k)} - \beta_1^{(k+1)} s\psi^{(k)} \\ 1 & 0 & \alpha_2^{(k+1)} c\psi^{(k)} - \beta_2^{(k+1)} s\psi^{(k)} & -\beta_2^{(k+1)} c\psi^{(k)} - \alpha_2^{(k+1)} s\psi^{(k)} \\ 0 & 1 & \beta_2^{(k+1)} c\psi^{(k)} + \alpha_2^{(k+1)} s\psi^{(k)} & \alpha_2^{(k+1)} c\psi^{(k)} - \beta_2^{(k+1)} s\psi^{(k)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \alpha_n^{(k+1)} c\psi^{(k)} - \beta_n^{(k+1)} s\psi^{(k)} & -\beta_n^{(k+1)} c\psi^{(k)} - \alpha_n^{(k+1)} s\psi^{(k)} \\ 0 & 1 & \beta_n^{(k+1)} c\psi^{(k)} + \alpha_n^{(k+1)} s\psi^{(k)} & \alpha_n^{(k+1)} c\psi^{(k)} - \beta_n^{(k+1)} s\psi^{(k)} \end{bmatrix} \times \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \alpha_1 c\psi - \beta_1 s\psi \\ \beta_1 c\psi + \alpha_1 s\psi \\ \alpha_2 c\psi - \beta_2 s\psi \\ \beta_2 c\psi + \alpha_2 s\psi \\ \cdots \\ \alpha_n c\psi - \beta_n s\psi \\ \beta_n c\psi + \alpha_n s\psi \end{bmatrix}^{(k)} \tag{13}$$

Equation 13 can be expressed in a more compact form:

$$\Omega \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \Theta \tag{14}$$

In this way, $(\Delta t_x, \Delta t_y, \Delta\psi)$ can be estimated as the solution to the following least squares problem:

$$\min_{\substack{\Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi}} \left(\Omega \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} - \Theta \right)^2 \tag{15}$$

for which $(\Delta t_x, \Delta t_y, c\Delta\psi, s\Delta\psi)^T = (\Omega^T\Omega)^{-1}\Omega^T\Theta$, where:

$$\Omega^T\Omega = \begin{bmatrix} n & 0 & \mathcal{A} & \mathcal{B} \\ 0 & n & -\mathcal{B} & \mathcal{A} \\ \mathcal{A} & -\mathcal{B} & \mathcal{C} & 0 \\ \mathcal{B} & \mathcal{A} & 0 & \mathcal{C} \end{bmatrix} \quad \text{and} \quad \Omega^T\Theta = \begin{bmatrix} \mathcal{D} \\ \mathcal{E} \\ \mathcal{F} \\ \mathcal{G} \end{bmatrix}$$

and

$$\mathcal{A} = \left(\sum \alpha_i^{(k+1)}\right) c\psi^{(k)} - \left(\sum \beta_i^{(k+1)}\right) s\psi^{(k)}$$

$$\mathcal{B} = -\left(\sum \beta_i^{(k+1)}\right) c\psi^{(k)} - \left(\sum \alpha_i^{(k+1)}\right) s\psi^{(k)}$$

$$\mathcal{C} = \sum \left(\alpha_i^{(k+1)}\right)^2 + \sum \left(\beta_i^{(k+1)}\right)^2$$

$$\mathcal{D} = \left(\sum \alpha_i^{(k)}\right) c\psi^{(k)} - \left(\sum \beta_i^{(k)}\right) s\psi^{(k)}$$

$$\mathcal{E} = \left(\sum \beta_i^{(k)}\right) c\psi^{(k)} + \left(\sum \alpha_i^{(k)}\right) s\psi^{(k)}$$

$$\mathcal{F} = \sum \alpha_i^{(k)}\alpha_i^{(k+1)} + \sum \beta_i^{(k)}\beta_i^{(k+1)}$$

$$\mathcal{G} = -\sum \alpha_i^{(k)}\beta_i^{(k+1)} + \sum \beta_i^{(k)}\alpha_i^{(k+1)}$$

The solution to Eq. 15 is given by:

$$\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \mathcal{AF} - \mathcal{CD} + \mathcal{BG} \\ \mathcal{AG} - \mathcal{CE} - \mathcal{BF} \\ \mathcal{AD} - n\mathcal{F} - \mathcal{BE} \\ \mathcal{AE} - n\mathcal{G} + \mathcal{BD} \end{bmatrix} / (\mathcal{A}^2 + \mathcal{B}^2 - n\mathcal{C})$$

(16)

Observe that, since $\cos\Delta\psi$ and $\sin\Delta\psi$ are estimated separately, it is not ensured $\cos^2\Delta\psi + \sin^2\Delta\psi = 1$. We deal with this problem in the posterior refinement step.

5.2 Non-Linear Refinement and Filtering

As discussed in the previous section, the odometer could not produce an optimal estimation (in the least-squares sense) of the 2D roto-translation $(\Delta t_x, \Delta t_y, \Delta\psi)$ performed by the vehicle because it does not ensure the solution meets $\cos^2\Delta\psi + \sin^2\Delta\psi = 1$. However, the solution can be used as the point of departure within a non-linear iterative optimization framework. To this end, Eq. 15 is re-arranged as a non-linear

optimization problem aiming at minimizing directly for $(\Delta t_x, \Delta t_y, \Delta\psi)$:

$$\min_{\substack{\Delta t_x \\ \Delta t_y \\ \Delta\psi}} \left(\Omega \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} - \Theta \right)^2 \tag{17}$$

The Levenberg–Marquardt algorithm is used to this purpose, and more precisely the implementation described in [22].

After the addition of this step, the algorithm cannot be regarded as a closed-form solution to visual odometry. However, since experiments show that a few iterations (5–10) are enough to attain the minimum, i.e. the solution of the odometer tends to be close to the optimum $(\Delta t_x, \Delta t_y, \Delta\psi)$, we consider it as an *almost-closed-form* solution.

5.3 Visual Odometer Structure

Figure 7 summarizes the structure of the visual odometer. As can be observed, matchings are filtered before the closed-form motion estimation stage runs in order to remove false matchings that can distort the least-squares estimation process. Matchings are discarded if the image locations of previous- and current-frame features are incompatible with the vehicle speed (i.e. they are too far away from one another), and also if the distance between feature descriptors does not verify the distance ratio test [23].

Finally, the last stage of the motion estimation process involves a discrete Kalman filter to counteract sensor noise. The filter assumes constant acceleration, so that the state and measurement vectors are, respectively, given by $(\dot{\mathbf{x}}, \ddot{\mathbf{x}})^T = (\dot{t}_x, \dot{t}_y, \dot{\psi}, \ddot{t}_x, \ddot{t}_y, \ddot{\psi})^T$ and $\Delta\mathbf{x} = (\Delta t_x, \Delta t_y, \Delta\psi)^T$:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^k = \begin{bmatrix} \mathbf{I} & \Delta\mathbf{T} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^{k-1} + Q,$$

$$[\Delta\mathbf{x}]^k = \begin{bmatrix} \Delta\mathbf{T} & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^k + R$$

where $\Delta\mathbf{T} = (\Delta t)\mathbf{I}_{3\times3}$ and Δt is the filter sampling period.

Fig. 7 Summary of the visual odometer
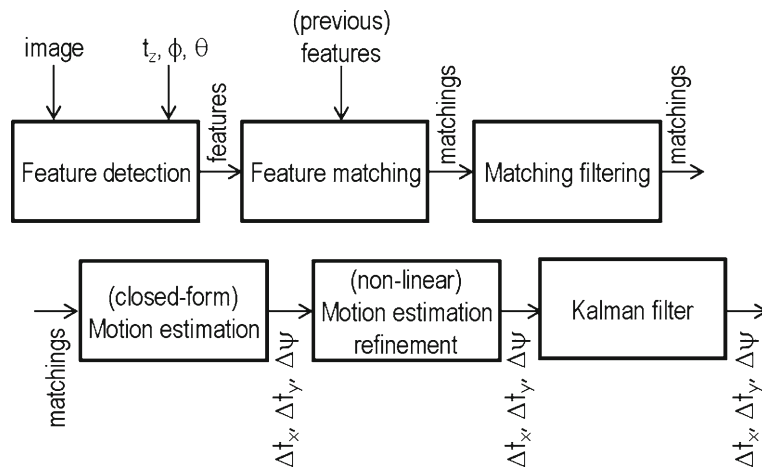
6 Experimental Results

This section reports performance results for the visual odometer described in Section 5, as well as for the full system during a field experiment. Results for the laser-based odometer alone can be found in [6].

6.1 Assessment of the Visual Odometer

The publicly available datasets *1LoopDown* (DS1), *2LoopsDown* (DS2), *3LoopsDown* (DS3) and *hoveringDown* (DS4) made public by project sFly[2] have been used to quantitatively assess the visual odometer. The datasets were collected from a quadrotor flying, respectively, 1, 2 and 3 loops, and hovering within a space of approximately 6 m × 3 m. These datasets comprise ground truth from a Vicon system and images from a ground-looking camera, as well as height and IMU data. See [20] for a description of the datasets.

To measure the performance of the visual odometry approach described in Section 5, different conditions of operation have been considered, comprising a total of nine combinations of feature detectors and descriptors. The combinations of feature detectors and descriptors consider both classical, well-known methods such as SIFT [23] and SURF [5], and relatively new fast detectors and descriptors with less computational requirements: FAST [31], ORB [33], BRIEF [8], and FREAK [29]. The *Good Features To Track* (GFTT) detector available in OpenCV has also been included in the set of combinations. Regarding roll ϕ and pitch θ angles, the odometer is fed using both Vicon data (best conditions of operation) or IMU data (typical conditions of operation), depending on the experiment, while height t_z is always taken from Vicon data.

Table 1 shows the performance data collected, measured as *error in x*, *in y* and *in yaw*, all averaged along the path and using Vicon data as ground truth. Frames per second for every combination is also provided. The related execution times correspond to an Intel Core i5 @2.53 GHz processor. In the table, DSi refers to dataset *i*, while *All* refers to the full set of frames (almost 6,700 in total).

As can be observed, best motion estimation results are obtained for the combination consisting of ORB both for feature detection and description when using IMU data for the roll and pitch angles, while FAST-ORB turns out to be the best when using Vicon data. In general, ORB used as a descriptor tends to produce the best estimations. In view of the quantitative data collected, a less computationally intensive option is FAST as detector and BRIEF or FREAK as descriptors. Under typical conditions of operation, the ORB-ORB combination leads to 11-cm-error in X, 10-cm-error in Y and 2-degree-error in yaw, while FAST-BRIEF/FREAK increases the average error in

---

[2]http://www.sfly.org/mav-datasets

Table 1 Performance measures for the visual odometer and various combinations of feature detectors and descriptors

|	DS1	DS2	DS3	DS4	All	DS1	DS2	DS3	DS4	All
No. frames	640	1653	2373	2026	6692	640	1653	2373	2026	6692
	(a)					(b)				
FAST-BRIEF	0.21	0.18	0.23	0.06	0.17	0.11	0.14	0.15	0.11	0.13
FAST-FREAK	0.18	0.10	0.16	0.10	0.13	0.11	0.13	0.16	0.08	0.12
FAST-ORB	0.16	0.08	0.09	0.05	0.08	0.10	0.07	0.07	0.14	0.09
GFTT-BRIEF	0.18	0.20	0.25	0.05	0.17	0.11	0.12	0.13	0.12	0.12
GFTT-FREAK	0.07	0.20	0.17	0.06	0.13	0.12	0.08	0.12	0.11	0.11
GFTT-ORB	0.17	0.10	0.13	0.05	0.10	0.09	0.06	0.06	0.11	0.08
ORB-ORB	0.30	0.23	0.26	0.05	0.19	0.12	0.11	0.12	0.10	0.11
SIFT-SIFT	0.24	0.28	0.36	0.06	0.24	0.12	0.16	0.18	0.13	0.15
SURF-SURF	0.35	0.39	0.46	0.07	0.32	0.13	0.16	0.19	0.11	0.16
	(c)					(d)				
FAST-BRIEF	6.70	6.57	8.19	2.54	5.94	195.60	188.01	180.46	131.76	164.86
FAST-FREAK	5.84	4.04	5.54	2.25	4.20	100.66	123.02	108.05	75.40	97.51
FAST-ORB	5.14	2.44	2.90	2.08	2.75	125.61	204.93	126.36	100.10	128.25
GFTT-BRIEF	5.88	6.40	8.32	2.53	5.86	83.22	111.59	83.16	74.96	85.72
GFTT-FREAK	2.66	2.06	2.87	3.22	2.76	79.47	101.14	76.67	65.16	77.42
GFTT-ORB	5.28	2.95	3.42	2.27	3.13	85.76	106.21	80.00	72.90	83.15
ORB-ORB	8.43	7.52	8.88	2.59	6.60	56.77	64.89	52.53	44.12	52.35
SIFT-SIFT	7.37	9.76	12.55	1.69	8.08	18.87	20.05	18.54	15.72	17.93
SURF-SURF	9.18	11.80	15.42	1.51	9.72	14.94	15.19	14.29	12.62	13.99
	(e)					(f)				
FAST-BRIEF	0.10	0.31	0.37	0.14	0.26	0.16	0.10	0.09	0.12	0.11
FAST-FREAK	0.14	0.27	0.38	0.20	0.27	0.17	0.11	0.10	0.09	0.11
FAST-ORB	0.15	0.32	0.35	0.14	0.26	0.14	0.08	0.09	0.14	0.11
GFTT-BRIEF	0.23	0.43	0.45	0.13	0.32	0.16	0.08	0.07	0.12	0.10
GFTT-FREAK	0.28	0.47	0.49	0.14	0.36	0.17	0.10	0.09	0.12	0.11
GFTT-ORB	0.26	0.33	0.34	0.11	0.26	0.12	0.09	0.09	0.11	0.10
ORB-ORB	0.12	0.07	0.13	0.11	0.11	0.13	0.09	0.08	0.11	0.10
SIFT-SIFT	0.09	0.15	0.25	0.16	0.18	0.18	0.16	0.16	0.13	0.15
SURF-SURF	0.12	0.18	0.19	0.16	0.17	0.17	0.16	0.16	0.12	0.15
	(g)					(h)				
FAST-BRIEF	1.85	2.93	2.57	1.61	2.30	143.53	137.24	131.61	101.59	122.84
FAST-FREAK	2.04	4.85	4.20	1.50	3.34	113.90	112.79	107.67	75.62	96.84
FAST-ORB	2.28	7.49	7.91	1.13	5.21	132.09	131.07	126.59	101.16	119.01
GFTT-BRIEF	2.60	5.31	4.58	2.06	3.81	83.80	83.80	82.66	74.75	80.46
GFTT-FREAK	5.88	10.54	11.78	3.04	8.26	75.26	76.69	76.41	65.41	72.67
GFTT-ORB	3.13	8.78	9.31	1.65	6.27	70.54	80.23	79.56	73.53	76.87
ORB-ORB	3.57	1.82	2.03	1.61	2.00	53.83	54.19	52.62	44.30	50.23
SIFT-SIFT	3.45	4.61	7.21	2.32	4.73	18.76	18.99	18.53	15.73	17.70
SURF-SURF	3.68	6.44	7.46	2.09	5.22	14.89	14.58	14.23	12.60	13.83

Height t_z comes from Vicon data, while roll ϕ and pitch θ angles are set from Vicon data (a–d) or from IMU data (e–h). Performance is measured as the average absolute difference with ground truth data along the path: (a,e) error in X [m], (b,f) error in Y [m], (c,g) error in yaw [deg], and (d,h) frames per second

X and Y up to, respectively, 26/27 and 11 cm, and 2.30/3.34 degrees in yaw.

By way of illustration, 2D plots of the estimated and ground truth (Vicon) paths for DS2 and the ORB-ORB combination using IMU data can be found in Fig. 9(left), while a histogram of the magnitude of the effort performed by the refinement step can be found in Fig. 8 also for this case,
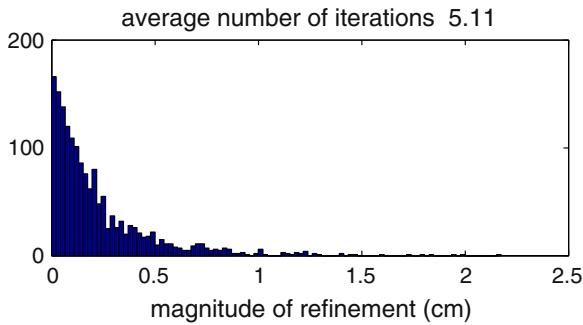
**Fig. 8** Illustration of the effort performed by the refinement step

measured as the norm of $(\Delta t_x, \Delta t_y)^{ar} - (\Delta t_x, \Delta t_y)^{br}$, where ar and br stand for, respectively, *after refinement* and *before refinement*. In general, during the off-line experiments, the refinement step required no more than 7–8 iterations to attain convergence, in accordance with Fig. 8.

To finish, Fig. 9(right) shows results for an on-line experiment where the MAV is controlled by the output of the visual odometer. The different colours in the figure correspond to the paths estimated during three different repetitions of the mission, consisting in taking off, hovering at a certain altitude, increase height, hover at the new

height, and finally land. For the whole set of repetitions, standard deviations in X and Y during the full experiment (resp. σ_X and σ_Y) were around 7 cm in X and 6 cm in Y.
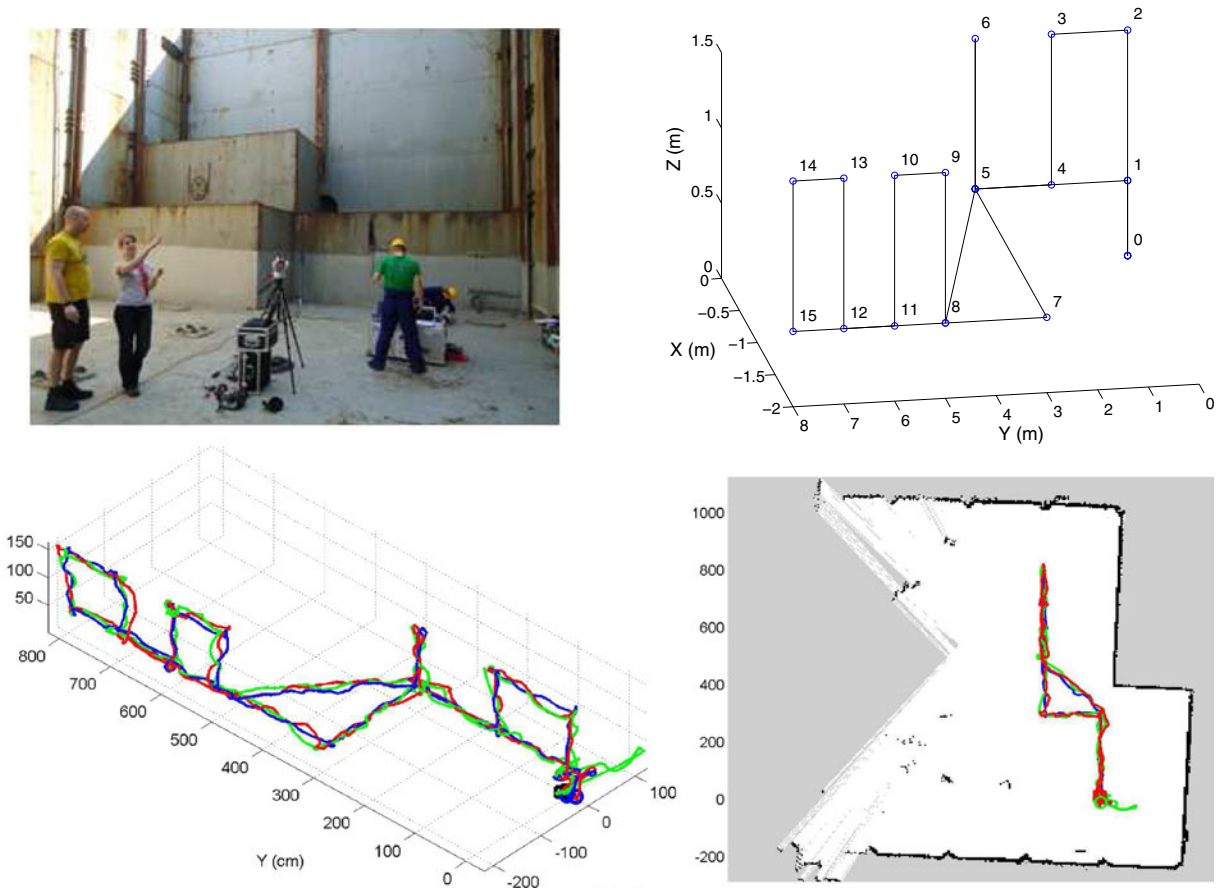
6.2 Field Experiment

This section presents results for an online experiment intended to show the suitability of the full hardware/software solution adopted for the inspection tasks described in previous sections. This experiment took place onboard a cargo hold of a containership, comprising a total volume of 2572 m^3 (approx. 20 × 12.50 × 10 m (L×W×H), see Fig. 10(top, left)). The mission description file specified a sweeping task, consisting in achieving a collection of waypoints along a zig-zag-like path (see Fig. 10(top, right)). The experiment, whose results can be found in Fig. 10(bottom) as the paths estimated by the MAV during the autonomous flights, was performed several times, as before, to compare results between consecutive executions. No ground truth data were available on this occasion for quantitative assessment. Nevertheless, as can be noticed, the estimated paths are consistent with the environment, as well as the map produced by the laser-based SLAM
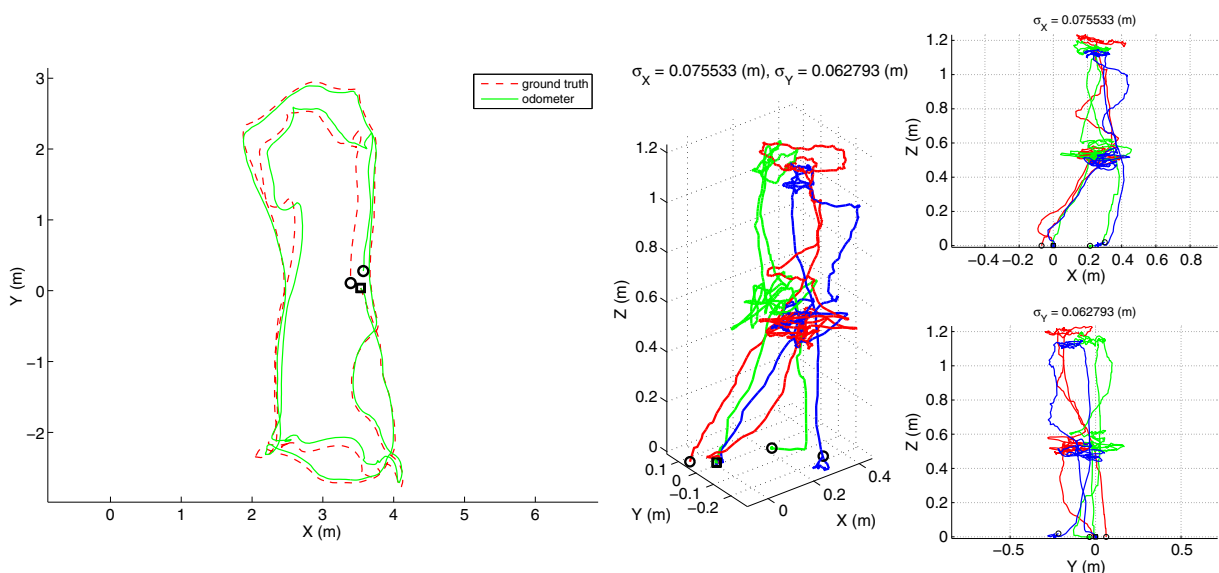
Fig. 9 Visual odometer results: (*left*) for DS2, 2D plot of estimated and ground truth paths for the visual odometer, using IMU data, and ORB as detector and descriptor; (*right*) estimated paths for three repetitions of the online experiment, using FAST as detector and FREAK as descriptor

Fig. 10 Field experiment results: (*top*, *left*) vessel hold where the experiment took place; (*top*, *right*) waypoints sequence: 0 - 6 / 5 / 7 - 15 / 8 / 5 / 1 / 10; (*bottom*) three executions of the experiment, shown in different colours:

(*left*) 3D plot of estimated MAV trajectories, (*right*) map produced by the laser-based SLAM module with the estimated MAV 2D paths superimposed

strategy adopted, what suggests the validity of the approach. The small differences which can be observed in these paths are due to, among other factors, the accuracy parameter included in the mission specification file that determines when an action has succeeded or not (set to 0.1 meters for this experiment). A video of this mission and others can be found in the MINOAS project Youtube channel http://www.youtube.com/user/ MINOASProject.

7 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 required to reply to these requirements have been described and discussed. A vision-based odometer contributing to the MAV self-localization has been derived and thoroughly tested off-line and online. Results for the platform performing a typical mission have also been reported. All in all, the different experimental results gathered suggest the platform proposed (hardware and software) are appropriate for the visual inspection mission described in this paper. More details can be found in [28].

Some improvements are foreseen for the near future, such as advancing on the fusion of the laser- and vision-based odometers, what would

require, in turn, upgrading the computational capabilities onboard the platform to be able to run them both simultaneously.

References

1. Achtelik, M., Achtelik, M., Weiss, S., Siegwart, R.: Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In: Intl. Conf. Robotics and Automation (2011)
2. Achtelik, M., Bachrach, A., He, R., Prentice, S., Roy, N.: Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments. In: Intl. Aerial Robotics Competition (2009)
3. Achtelik, M.W., Lynen, S., Weiss, S., Kneip, L., Chli, M., Siegwart, R.: Visual-inertial SLAM for a small helicopter in large outdoor environments. In: Intl. Conf. Intell. Robots and Systems (2012)
4. Bachrach, A., Prentice, S., He, R., Roy, N.: RANGE-Robust autonomous navigation in GPS-denied environments. J. Field Robotics **28**(5), 644–666 (2011)
5. Bay, H., Tuytelaars, T., Gool, L.V.: SURF: Speeded up robust features. In: European Conf. Computer Vision (2006)
6. Bonnin-Pascual, F., Garcia-Fidalgo, E., Ortiz, A.: Semi-autonomous visual inspection of vessels assisted by an unmanned micro aerial vehicle. In: Intl. Conf. Intell. Robots and Systems (2012)
7. Bouabdallah, S., Murrieri, P., Siegwart, R.: Towards Autonomous Indoor Micro VTOL. Auton. Robot. **18**, 171–183 (2005)
8. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: computing a local binary descriptor very fast. Trans. Pattern Anal. Mach. Intell. **34**(7), 1281–1298 (2012)
9. Castillo, P., Lozano, R., Dzul, A.: Modelling and control of mini-flying machines. Advances in Industrial Control. Springer (2005)
10. Chowdhary, G., Johnson, E.N., Magree, D., Wu, A., Shein, A.: GPS-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft. J. Field Robotics **30**(3), 415–438 (2013)
11. Dryanovski, I., Valenti, R.G., Xiao, J.: An open-source navigation system for micro aerial vehicles. Auton. Robot. **34**(3), 177–188 (2013)
12. Eberli, D., Scaramuzza, D., Weiss, S., Siegwart, R.: Vision based position control for MAVs using one single circular landmark. J. Intel. Robotic Systems **61**(1–4), 495–512 (2011)
13. Engel, J., Sturm, J., Cremers, D.: Camera-based navigation of a low-cost quadrocopter. In: Intl. Conf. Intell. Robots and Systems (2012)
14. Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Lorenz, Meier, Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor MAV. In: Intl. Conf. Intell. Robots and Systems (2012)
15. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. Trans. Robotics **23**(1), 34–46 (2007)
16. Grzonka, S., Grisetti, G., Burgard, W.: A fully autonomous indoor quadrotor. Trans. Robotics **28**(1), 90–100 (2012)
17. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.M., Hirzinger, G., Rus, D.: Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In: Intl. Conf. Robotics and Automation (2007)
18. He, R., Prentice, S., Roy, N.: Planning in information space for a quadrotor helicopter in a GPS-denied environment. In: Intl. Conf. Robotics and Automation (2008)
19. Honegger, D., Meier, L., Tanskanen, P., Pollefeys, M.: An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In: Intl. Conf. Robotics and Automation (2013)
20. Lee, G., Achtelik, M., Fraundorfer, F., Pollefeys, M., Siegwart, R.: A benchmarking tool for MAV visual pose estimation. In: Intl. Conf. Control, Automation, Robotics and Vision (2010)
21. Li, W., Zhang, T., Kuhnlenz, K.: A vision-guided autonomous quadrotor in an air-ground multi-robot system. In: Intl. Conf. Robotics and Automation (2011)
22. Lourakis, M.: levmar: Levenberg–Marquardt nonlinear least squares algorithms in C/C++. http://www.ics.forth.gr/~lourakis/levmar/. Accessed 24 July 2011
23. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Intl. J. Computer Vision **60**(2), 91–110 (2004)
24. Matsue, A., Hirosue, W., Tokutake, H., Sunada, S., Ohkura, A.: Navigation of small and lightweight helicopter. Trans. Japan Soc. for Aeronautical and Space Sciences **48**(161), 177–179 (2005)
25. Meier, L., Tanskanen, P., Heng, L., Lee, G.H., Fraundorfer, F., Pollefeys, M.: PIXHAWK: a micro aerial vehicle design for autonomous flight using onboard computer vision. Auton. Robot. **33**(1–2), 21–39 (2012)
26. Newsome, S., Rodocker, J.: Effective technology for underwater hull and infrastructure inspection: the SeaBotix LBC. In: Oceans Conf. (2009)
27. Ortiz, A., Bonnin, F., Gibbins, A., Apostolopoulou, P., Bateman, W., Eich, M., Spadoni, F., Caccia, M., Drikos, L.: First steps towards a roboticized visual inspection system for vessels. In: Intl. Conf. Emerging Technologies and Factory Automation (2010)
28. Ortiz, A., Bonnin-Pascual, F., Garcia-Fidalgo, E.: Vessel inspection assistance by means of a micro-aerial vehicle: control architecture and self-localization issues. Tech. Rep. A-02-2013, Dep. of Mathematics and Comp. Science - UIB (2013). http://dmi.uib.es/aortiz/publ/tr0213.pdf
29. Ortiz, R.: FREAK: fast retina keypoint. In: Conf. Computer Vision and Pattern Recognition (2012)
30. Roberts, J.F., Stirling, T., Zufferey, J.C., Floreano, D.: Quadrotor using minimal sensing for autonomous indoor flight. In: Proc. European Micro Air Vehicle Conf. and Flight Competition (2007)

31. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: Intl. Conf. Computer Vision (2005)
32. ROTISII: Publishable Final Activity Report. http://cordis.europa.eu/projects/index.cfm?fuseaction=app.details&REF=74284. Accessed 10 Dec 2012
33. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: Intl. Conf. Computer Vision (2011)
34. Wenzel, K.E., Masselli, A., Zell, A.: Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. J. Intel. Robotic Systems **61**(1–4), 221–238 (2011)
35. Zingg, S., Scaramuzza, D., Weiss, S., Siegwart, R.: MAV navigation through indoor corridors using optical flow. In: Intl. Conf. Robotics and Automation (2010)