Journal of Intelligent & Robotic Systems manuscript No.
(will be inserted by the editor)

Vessel Inspection: A Micro-Aerial Vehicle-based Approach

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

Received: date / Accepted: date

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 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) to be used as part of this fleet and which 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 reported and discussed.

Keywords MAV · Visual odometry · Visual inspection · Vessels maintenance

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 both in personal, environmental and financial terms. 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 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.

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 860m of web frames (primary stiffening members) and approximately 3.2km 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 $1M 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

This work has been partially supported by the European Social Fund through grants FPI10-43175042V and FPI11-4312 3621R (Conselleria d'Educació, Cultura i Universitats, Govern de les Illes Balears)

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, xisco.bonnin@uib.es, emilio.garcia@uib.es

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

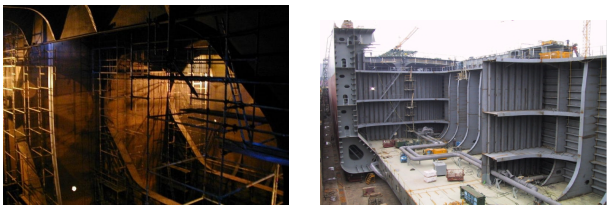


Fig. 1 (left) Staging required during a vessel inspection. (right) Oil tanker in shipyard during construction

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 [24] for a detailed discussion). Contrary to similar past projects (ROTIS and its follow-up ROTIS-II [1]) or commercial solutions such as [23], 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 mentioned, 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 two monocular odometers 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.

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 [24] 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, taking thickness measurements (by means of other platforms of the robot fleet), or even be compared in a posterior inspection.

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 (e.g. due to satellites being occluded by the vessel structures, multi-path effects, etc.).

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 (see e.g. [12, 26]): (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 make 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 this kind of platforms. 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 is the most commonly used sensing device, due to its accuracy and speed. In this regard, Grzonka et al [15] present an open-source solution which enables a small sized flying vehicle to operate indoors. Dryanovski et al [13] design a system also based in open-source components, showing experimental results for SLAM and 3D mapping tasks. He et al [17] propose a solution for planning vehicle trajectories using an unscented Kalman filter and a laser range finder. Bachrach et at [5] describe another approach using the same kind of sensing devices.

Infrared or ultrasounds are other possible sensors to be used for navigation tasks. Although they present less noise tolerance and accuracy, several researchers [9,22,27] have used them to perform navigation tasks in indoor environments, being a cheaper option than laser scanners.

Vision cameras have also been used. They are the cheapest option, although at a higher computational cost. Some examples of this approach can be found in [3,7,10,18].

Some authors have combined different sources in order to improve the accuracy of the localization process. For instance, Achtelik et al [4] present a platform which combines laser scanners and vision cameras to autonomously navigate in indoor environments.

Fig. 2 The Micro Aerial Vehicle

As will be described later, in our case, a combination of vision- and laser-based motion estimation has been adopted for self-localization. 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 does not result to be a critical requirement, what contributes to save time for critical processes. Further, all flight safety-related processing is performed onboard in order to reduce the use of wireless datalinks, which are not favoured within vessel holds due to the surrounding metallic structures.

Among the different tasks to solve and implement, this paper focuses on platform motion estimation and self-localization, but does not address the obstacle avoidance issues.

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 the application at hand (the accuracy is around 1-3% of the distance travelled by the beam), instead of using the barometric pressure sen-

sor or the GPS, which tend to show large variations around the true height, making height stabilization difficult when the platform navigates indoors or relatively close to other objects.

Visual information is collected by means of a flexible vision system devised around an appropriate structure for supporting one bottom-looking camera and two additional units, which can be tailored for the particular inspection mission to be performed as: two forward-facing cameras forming a stereo vision system, one camera facing forward and the other facing up, or, to save weight, a single camera facing forward. 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 5.8GHz is also attached to provide real-time imagery 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 1.6GHz processor and 1GB 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 [16] (in Fig. 3, it appears as the low-level controller); (2) the secondary ARM7 controller runs the position controller described in [3] (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 [2]) 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

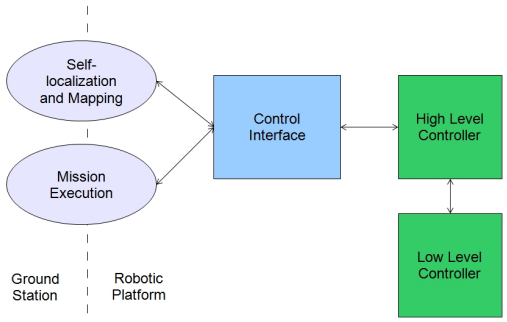


Fig. 3 Control architecture overview.

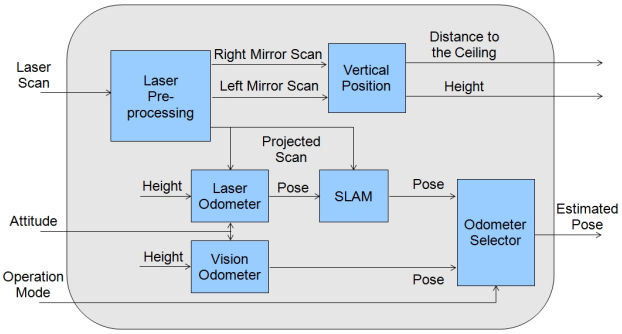


Fig. 4 Self-localization and mapping

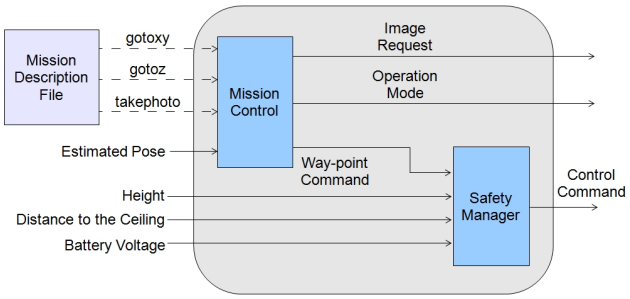


Fig. 5 Mission execution

loops run onboard the vehicle in order to ensure minimum delay, a requirement also reported by other authors [4] to permit autonomous flying.

As well as the self-localization and mapping solution described in [13], our MAV control software has been designed around open-source components and following modularity and software reutilization principles. In this way, adapting the platform for different missions involving different payloads, or the selective activation of software modules, can be performed in a fast and reliable way. In this regard, ROS has also proved to be specially useful and, in fact, has guided the control software modularization.

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

```
<mission>
<gotoz z="1.0" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
<takephoto camera="1" path="picture1.jpg" stay_time="5.0" />
<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" />
<takephoto camera="1" path="picture2.jpg" stay_time="5.0" />
<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" />
<gotoz z="0.0" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" />
</mission>
```

Fig. 6 Example of a mission specification file

4.1 Self-Localization and Mapping

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

Within this system, the *Laser Pre-processing* module prepares raw laser scans for the rest of the system. More precisely, it is in charge of: (1) filtering the laser beams that are of no interest; (2) 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 passed onto a scan matcher, which computes the platform roto-translation between consecutive scans and estimates a new 2D pose (x, y, ψ), using the yaw angle provided by the IMU as initial estimate for ψ. The 2D pose so obtained is then combined with the height and the roll and pitch angles (ϕ, θ), provided by, respectively, the laser altimeter and the IMU, to obtain a 3D pose for the vehicle.

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 [14], 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 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).

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* 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 on-board sensors. As will be seen, the odometer finally becomes into a least squares problem for 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.

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.

Although within the vessel inspection application the vision-based odometer is employed only during vertical motion, it is of general application, as will be shown in the experimental results section.

5.1 Derivation of the Odometer in Closed-Form Solution for Generic Yaw Rotation

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 $\sin\alpha$.)

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}}_{P} \overbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\phi & s\phi & 0 \\ 0 & -s\phi & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_x \text{ (roll)}} \underbrace{\begin{bmatrix} c\theta & 0 & -s\theta & 0 \\ 0 & 1 & 0 & 0 \\ s\theta & 0 & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_y \text{ (pitch)}}}^{T^R_{Wb}} \overbrace{\underbrace{\begin{bmatrix} c\psi & s\psi & 0 & 0 \\ -s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_z \text{ (yaw)}} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{T^R_{Wa}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

The corresponding inverse perspective transformation is thus:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underbrace{\left(T^R_{Wa}\right)^{-1}}_{M^*} \underbrace{\left(T^R_{Wb}\right)^{-1} P^{-1}}_{M} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \quad (2)$$

where:

$$M^* = \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} = \begin{bmatrix} c\theta & s\theta s\phi & s\theta c\phi & 0 \\ 0 & c\phi & -s\phi & 0 \\ -s\theta & c\theta s\phi & c\theta c\phi & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix}$$

Equation 3 is obtained next by developing Eq. 2:

$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} &= M^* M \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_1 x_p + M_2 y_p + M_3 z_p + M_4 \\ M_5 x_p + M_6 y_p + M_7 z_p + M_8 \\ M_9 x_p + M_{10} y_p + M_{11} z_p + M_{12} \\ \frac{f - z_p}{f} \end{bmatrix} \\ &= \begin{bmatrix} c\psi & -s\psi & 0 & t_x \\ s\psi & c\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a\,c\psi - b\,s\psi + d\,t_x \\ b\,c\psi + a\,s\psi + d\,t_y \\ c + d\,t_z \\ d \end{bmatrix} \end{aligned} \quad (3)$$

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:

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

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:

$$v = v_l + \lambda (v_p - v_l) \tag{5}$$

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 $z_p = 0$:

$$v_l = \lim_{z_p \to -\infty} v = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - f \begin{bmatrix} M_3\,c\psi - M_7\,s\psi \\ M_7\,c\psi + M_3\,s\psi \\ M_{11} \end{bmatrix} \tag{6}$$

$$v_p = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} (M_1 x_p + M_2 y_p + M_4)c\psi - (M_5 x_p + M_6 y_p + M_8)s\psi \\ (M_5 x_p + M_6 y_p + M_8)c\psi + (M_1 x_p + M_2 y_p + M_4)s\psi \\ M_9 x_p + M_{10} y_p + M_{12} \end{bmatrix} \tag{7}$$

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

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} -M_3 f\,c\psi + M_7 f\,s\psi \\ -M_7 f\,c\psi - M_3 f\,s\psi \\ -M_{11} f \end{bmatrix} + \lambda \cdot \begin{bmatrix} (M_1 x_p + M_2 y_p + M_3 f + M_4)c\psi - (M_5 x_p + M_6 y_p + M_7 f + M_8)s\psi \\ (M_5 x_p + M_6 y_p + M_7 f + M_8)c\psi + (M_1 x_p + M_2 y_p + M_3 f + M_4)s\psi \\ M_9 x_p + M_{10} y_p + M_{11} f + M_{12} \end{bmatrix} \tag{8}$$

The previous line corresponds to the ray backprojecting 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:

$$\lambda = -\frac{t_z - M_{11} f}{M_9 x_p + M_{10} y_p + M_{11} f + M_{12}} \tag{9}$$

Using Eq. 9, Eq. 8 can be written as follows for time k:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}^{(k)} + \begin{bmatrix} C + \lambda A & -(D + \lambda B) \\ D + \lambda B & (C + \lambda A) \end{bmatrix}^{(k)} \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} \tag{10}$$

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. 10 can be written in a more compact form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}^{(k)} + \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}^{(k)} \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} \tag{11}$$

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:

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} t_x^{(k)} + \Delta t_x \\ t_y^{(k)} + \Delta t_y \end{bmatrix} + \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}^{(k+1)} \begin{bmatrix} c\psi^{(k)} c\Delta\psi - s\psi^{(k)} s\Delta\psi \\ s\psi^{(k)} c\Delta\psi + c\psi^{(k)} s\Delta\psi \end{bmatrix} \\ &= \begin{bmatrix} t_x^{(k)} + \Delta t_x \\ t_y^{(k)} + \Delta t_y \end{bmatrix} + \begin{bmatrix} \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \\ \beta^{(k+1)} c\psi^{(k)} + \alpha^{(k+1)} s\psi^{(k)} \end{bmatrix} \\ &\qquad \begin{bmatrix} -\beta^{(k+1)} c\psi^{(k)} - \alpha^{(k+1)} s\psi^{(k)} \\ \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \end{bmatrix} \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix} \end{aligned} \tag{12}$$

Equating 11 and 12 we can write:

$$\begin{bmatrix} \alpha c\psi - \beta s\psi \\ \beta c\psi + \alpha s\psi \end{bmatrix}^{(k)} = \begin{bmatrix} \Delta t_x \\ \Delta t_y \end{bmatrix} + \begin{bmatrix} \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \\ \beta^{(k+1)} c\psi^{(k)} + \alpha^{(k+1)} s\psi^{(k)} \end{bmatrix} \begin{bmatrix} -\beta^{(k+1)} c\psi^{(k)} - \alpha^{(k+1)} s\psi^{(k)} \\ \alpha^{(k+1)} c\psi^{(k)} - \beta^{(k+1)} s\psi^{(k)} \end{bmatrix} \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix} \tag{13}$$

Equation 14 can next be obtained rearranging equation 13:

$$\begin{bmatrix} 1 & 0 & \alpha^{(k+1)}c\psi^{(k)} - \beta^{(k+1)}s\psi^{(k)} & -\beta^{(k+1)}c\psi^{(k)} - \alpha^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \beta^{(k+1)}c\psi^{(k)} + \alpha^{(k+1)}s\psi^{(k)} & \alpha^{(k+1)}c\psi^{(k)} - \beta^{(k+1)}s\psi^{(k)} \end{bmatrix} \times \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \alpha c\psi - \beta s\psi \\ \beta c\psi + \alpha s\psi \end{bmatrix}^{(k)} \quad (14)$$

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 equation 14 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 $k+1$:

$$\begin{bmatrix} 1 & 0 & \alpha_1^{(k+1)}c\psi^{(k)} - \beta_1^{(k+1)}s\psi^{(k)} & -\beta_1^{(k+1)}c\psi^{(k)} - \alpha_1^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \beta_1^{(k+1)}c\psi^{(k)} + \alpha_1^{(k+1)}s\psi^{(k)} & \alpha_1^{(k+1)}c\psi^{(k)} - \beta_1^{(k+1)}s\psi^{(k)} \\ 1 & 0 & \alpha_2^{(k+1)}c\psi^{(k)} - \beta_2^{(k+1)}s\psi^{(k)} & -\beta_2^{(k+1)}c\psi^{(k)} - \alpha_2^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \beta_2^{(k+1)}c\psi^{(k)} + \alpha_2^{(k+1)}s\psi^{(k)} & \alpha_2^{(k+1)}c\psi^{(k)} - \beta_2^{(k+1)}s\psi^{(k)} \\ \multicolumn{4}{c}{\dots} \\ 1 & 0 & \alpha_n^{(k+1)}c\psi^{(k)} - \beta_n^{(k+1)}s\psi^{(k)} & -\beta_n^{(k+1)}c\psi^{(k)} - \alpha_n^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \beta_n^{(k+1)}c\psi^{(k)} + \alpha_n^{(k+1)}s\psi^{(k)} & \alpha_n^{(k+1)}c\psi^{(k)} - \beta_n^{(k+1)}s\psi^{(k)} \end{bmatrix} \times \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \alpha_1 c\psi - \beta_1 s\psi \\ \beta_1 c\psi + \alpha_1 s\psi \\ \alpha_2 c\psi - \beta_2 s\psi \\ \beta_2 c\psi + \alpha_2 s\psi \\ \dots \\ \alpha_n c\psi - \beta_n s\psi \\ \beta_n c\psi + \alpha_n s\psi \end{bmatrix}^{(k)} \quad (15)$$

Equation 15 can be expressed in a more compact form as:

$$\Omega \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \Theta \quad (16)$$

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

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

which turns out to be:

$$\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = (\Omega^T\Omega)^{-1}\Omega^T\Theta \quad (18)$$

where

$$\Omega^T\Omega = \begin{bmatrix} n & 0 & \mathcal{A} & \mathcal{B} \\ 0 & n & -\mathcal{B} & \mathcal{A} \\ \mathcal{A} & -\mathcal{B} & \mathcal{C} & 0 \\ \mathcal{B} & \mathcal{A} & 0 & \mathcal{C} \end{bmatrix} \quad \text{and} \quad \Omega^T\Theta = \begin{bmatrix} \mathcal{D} \\ \mathcal{E} \\ \mathcal{F} \\ \mathcal{G} \end{bmatrix} \quad (19)$$

and

$$\begin{aligned} \mathcal{A} &= \left(\sum \alpha_i^{(k+1)}\right) c\psi^{(k)} - \left(\sum \beta_i^{(k+1)}\right) s\psi^{(k)} \\ \mathcal{B} &= -\left(\sum \beta_i^{(k+1)}\right) c\psi^{(k)} - \left(\sum \alpha_i^{(k+1)}\right) s\psi^{(k)} \\ \mathcal{C} &= \sum \left(\alpha_i^{(k+1)}\right)^2 + \sum \left(\beta_i^{(k+1)}\right)^2 \\ \mathcal{D} &= \left(\sum \alpha_i^{(k)}\right) c\psi^{(k)} - \left(\sum \beta_i^{(k)}\right) s\psi^{(k)} \\ \mathcal{E} &= \left(\sum \beta_i^{(k)}\right) c\psi^{(k)} + \left(\sum \alpha_i^{(k)}\right) s\psi^{(k)} \\ \mathcal{F} &= \sum \alpha_i^{(k)}\alpha_i^{(k+1)} + \sum \beta_i^{(k)}\beta_i^{(k+1)} \\ \mathcal{G} &= -\sum \alpha_i^{(k)}\beta_i^{(k+1)} + \sum \beta_i^{(k)}\alpha_i^{(k+1)} \end{aligned}$$

The solution to Eq. 17 in this case is given by:

$$\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \mathcal{A}\mathcal{F} - \mathcal{C}\mathcal{D} + \mathcal{B}\mathcal{G} \\ \mathcal{A}\mathcal{G} - \mathcal{C}\mathcal{E} - \mathcal{B}\mathcal{F} \\ \mathcal{A}\mathcal{D} - n\mathcal{F} - \mathcal{B}\mathcal{E} \\ \mathcal{A}\mathcal{E} - n\mathcal{G} + \mathcal{B}\mathcal{D} \end{bmatrix} / (\mathcal{A}^2 + \mathcal{B}^2 - n\mathcal{C}) \quad (20)$$

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 a posterior refinement step.

5.2 Derivation of the Odometer in Closed-Form Solution for Small Yaw Rotation

For small yaw rotations, $c\Delta\psi \approx 1$ and $s\Delta\psi \approx \Delta\psi$. Equation 15 then becomes:

$$\begin{bmatrix} 1 & 0 & -\beta_1^{(k+1)}c\psi^{(k)} - \alpha_1^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \alpha_1^{(k+1)}c\psi^{(k)} - \beta_1^{(k+1)}s\psi^{(k)} \\ 1 & 0 & -\beta_2^{(k+1)}c\psi^{(k)} - \alpha_2^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \alpha_2^{(k+1)}c\psi^{(k)} - \beta_2^{(k+1)}s\psi^{(k)} \\ \multicolumn{3}{c}{\dots} \\ 1 & 0 & -\beta_n^{(k+1)}c\psi^{(k)} - \alpha_n^{(k+1)}s\psi^{(k)} \\ 0 & 1 & \alpha_n^{(k+1)}c\psi^{(k)} - \beta_n^{(k+1)}s\psi^{(k)} \end{bmatrix} \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} (\alpha_1^{(k)} - \alpha_1^{(k+1)})c\psi^{(k)} - (\beta_1^{(k)} - \beta_1^{(k+1)})s\psi^{(k)} \\ (\beta_1^{(k)} - \beta_1^{(k+1)})c\psi^{(k)} + (\alpha_1^{(k)} - \alpha_1^{(k+1)})s\psi^{(k)} \\ (\alpha_2^{(k)} - \alpha_2^{(k+1)})c\psi^{(k)} - (\beta_2^{(k)} - \beta_2^{(k+1)})s\psi^{(k)} \\ (\beta_2^{(k)} - \beta_2^{(k+1)})c\psi^{(k)} + (\alpha_2^{(k)} - \alpha_2^{(k+1)})s\psi^{(k)} \\ \dots \\ (\alpha_n^{(k)} - \alpha_n^{(k+1)})c\psi^{(k)} - (\beta_n^{(k)} - \beta_n^{(k+1)})s\psi^{(k)} \\ (\beta_n^{(k)} - \beta_n^{(k+1)})c\psi^{(k)} + (\alpha_n^{(k)} - \alpha_n^{(k+1)})s\psi^{(k)} \end{bmatrix} \quad (21)$$

which avoids having to take care $\Delta\psi$ satisfies $(c\Delta\psi)^2 + (s\Delta\psi)^2 = 1$. In this case, the solution to the (reduced) least squares problem:

$$\min_{\substack{\Delta t_x \\ \Delta t_y \\ \Delta\psi}} \left(\Gamma \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\psi \end{bmatrix} - \Lambda \right)^2 \tag{22}$$

turns out to be:

$$\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\psi \end{bmatrix} = (\Gamma^T\Gamma)^{-1}\Gamma^T\Lambda \tag{23}$$

where

$$\Gamma^T\Gamma = \begin{bmatrix} n & 0 & \mathcal{B} \\ 0 & n & \mathcal{A} \\ \mathcal{B} & \mathcal{A} & \mathcal{C} \end{bmatrix} \quad \text{and} \quad \Gamma^T\Lambda = \begin{bmatrix} \mathcal{D}-\mathcal{A} \\ \mathcal{E}+\mathcal{B} \\ \mathcal{G} \end{bmatrix} \tag{24}$$

The solution to Eq. 17 is now given by:

$$\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} \frac{-\mathcal{A}\mathcal{B}\mathcal{E}+\mathcal{A}^2\mathcal{D}-\mathcal{A}^3-\mathcal{A}\mathcal{B}^2+n\mathcal{A}\mathcal{C}+n\mathcal{B}\mathcal{G}-n\mathcal{C}\mathcal{D}}{n} \\ \frac{-\mathcal{A}\mathcal{B}\mathcal{D}+\mathcal{A}^2\mathcal{B}+\mathcal{B}^3+\mathcal{E}\mathcal{B}^2-n\mathcal{B}\mathcal{C}+n\mathcal{A}\mathcal{G}-n\mathcal{C}\mathcal{E}}{n} \\ \mathcal{B}\mathcal{D}+\mathcal{A}\mathcal{E}-n\mathcal{G} \end{bmatrix} / (\mathcal{A}^2+\mathcal{B}^2-n\mathcal{C}) \tag{25}$$

5.3 Non-Linear Refinement and Filtering

As discussed in the previous sections, neither the odometer for generic yaw rotation nor the odometer for small yaw rotation have to necessarily 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 the former does not ensure the solution meets $\cos^2\Delta\psi + \sin^2\Delta\psi = 1$, while the latter has been dervied assuming $\Delta\psi$ is small. However, both solutions can be used as the point of departure within a non-linear iterative optimization framework. To this end, Eq. 17 is re-arranged as a non-linear optimization problem aiming at minimizing directly for $(\Delta t_x, \Delta t_y, \Delta\psi)$:

$$\min_{\substack{\Delta t_x \\ \Delta t_y \\ \Delta\psi}} \left(\Omega \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} - \Theta \right)^2 \tag{26}$$

The Levenberg-Marquardt algorithm is used to this purpose, and more precisely the implementation described in [20].

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 solutions of both odometers are close to the optimum values, we consider them both as *almost-closed-form* solutions.

5.4 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-frame and current-frame features for any matching are incompatible with 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 [21].

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 $(\dot{\mathbf{x}}, \ddot{\mathbf{x}})^T = (\dot{t}_x, \dot{t}_x, \dot{\psi}, \ddot{t}_x, \ddot{t}_y, \ddot{\psi})^T$ and $\Delta\mathbf{x} = (\Delta t_x, \Delta t_y, \Delta\psi)^T$:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^k = \begin{bmatrix} \mathrm{I} & \Delta\mathrm{T} \\ 0 & \mathrm{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^{k-1} + Q, \quad \begin{bmatrix} \Delta\mathbf{x} \end{bmatrix}^k = \begin{bmatrix} \Delta\mathrm{T} & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^k + R \tag{27}$$

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

6 Experimental Results

This section reports performance results for the two visual odometers described in Section 5, generic yaw rotation (GEN) and small yaw rotation (SMA, *Small Angle Approximation*), as well as for the full system during a field experiment. Results for the laser-based odometer alone can be found in [8].

6.1 Assessment of the Visual Odometers

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 two visual odometers. The datasets were collected from a quadrotor flying, respectively, 1, 2 and 3 loops, and hovering within a space of approximately 6m $\times$ 3m. These datasets comprise ground truth from a Vicon system and images from a ground-looking camera, as well as height and IMU data. See [19] for a detailed 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 and the two odometers. The combinations of feature detectors and descriptors consider both classical, well-known methods such

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

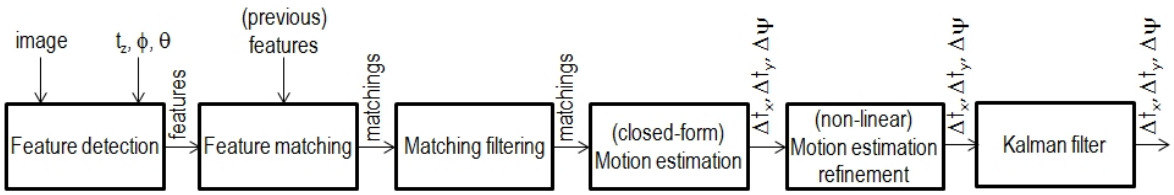


Fig. 7 Summary of the visual odometer

as SIFT [21] and SURF [6], and relatively new fast detectors and descriptors with less computational requirements: FAST [28], ORB [29], BRIEF [11], and FREAK [25]. The *Good Features To Track* (GFTT) detector available in OpenCV [3] has also been included in the set of combinations. Regarding roll ϕ and pitch θ angles, the odometers are 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.

During the off-line experiments, GEN and SMA both leaded to the same estimation results (the latter required a bit less computation), what indicates that both produce estimations close to the least-squares optimum that finally attains the posterior non-linear refinement step [4]. Results are thus only provided for GEN. Figure 8 and Tables 1 and 2 show the performance data collected, measured as *average error in x*, *average error in y* and *average error in yaw*, where Vicon data are used as ground truth. Frames per second for every combination is also provided. The related execution times correspond to an Intel Core i5 @2.53GHz processor. In the figures and in the tables, DSi refer to dataset *i*, while *All* refers to the full set of frames (almost 6700 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 X and Y up to, respectively, 26/27 cm 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(a).

To finish, Fig. 9(b-d) show results for an online experiment where the MAV is controlled by the output of the GEN 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.

[3] http://opencv.org/

[4] It typically required no more than 7-8 iterations to attain convergence.

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. $20 \times 12.50 \times 10$ m (L×W×H), see Fig. 10(a) and (b)). The mission description file specified a sweeping task, consisting in achieving a collection of waypoints along a zig-zag-like path (see Fig. 10(c)). The experiment, whose results can be found in Fig. 11 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 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). Pictures taken at some of the waypoints by the onboard cameras can be found in Fig. 12. A video of this mission and others in different environments can be found in the MINOAS 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. Two vision-based odometers contributing to the MAV self-localization have also 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.

Nevertheless, 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. A beneficial side effect of the latter would be the reduction of the dependence of the platform on the wireless link, a critical point of the MAV-based approach due to the metallic nature of the environment where the inspections take place.

References

1. ROTISII: Publishable Final Activity Report. [http://cordis.europa.eu/projects/index.cfm?fuseaction=app.details&REF=74284]. [Date of access: 10/12/2012]
2. ROS: an open-source Robot Operating System. In: Proc. of ICRA Workshop on Open Source Software (2009)
3. Achtelik, M., Achtelik, M., Weiss, S., Siegwart, R.: Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 3056–3063 (2011). DOI 10.1109/ICRA.2011.5980343
4. Achtelik, M., Bachrach, A., He, R., Prentice, S., Roy, N.: Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments. International Aerial Robotics Competition pp. 582–586 (2009)
5. Bachrach, A., de Winter, A., He, R., Hemann, G., Prentice, S., Roy, N.: RANGE - robust autonomous navigation in GPS-denied environments. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 1096–1097 (2010)
6. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: In Proceedings of the European Conference on Computer Vision, pp. 404–417 (2006)
7. Blosch, M., Weiss, S., Scaramuzza, D., Siegwart, R.: Vision based MAV navigation in unknown and unstructured environments. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 21–28 (2010)
8. Bonnin-Pascual, F., Garcia-Fidalgo, E., Ortiz, A.: Semi-autonomous visual inspection of vessels assisted by an unmanned micro aerial vehicle. In: Proc. IEEE Intl. Conf. on Intelligent Robots and Systems, pp. 3955–3961 (2012)
9. Bouabdallah, S., Murrieri, P., Siegwart, R.: Towards Autonomous Indoor Micro VTOL. Autonomous Robots **18**, 171–183 (2005)
10. Buskey, G., Roberts, J., Corke, P., Wyeth, G.: Helicopter automation using a low-cost sensing system. Computing Control Engineering Journal **15**(2), 8–9 (2004)
11. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a Local Binary Descriptor Very Fast. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1281–1298 (2012)
12. Castillo, P., Lozano, R., Dzul, A.: Modelling and control of mini flying machines. Advances in Industrial Control. Springer-Verlag (2005)
13. Dryanovski, I., Morris, W., Xiao, J.: An Open-Source Pose Estimation System for Micro-Air Vehicles. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 4449–4454 (2011)
14. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. Transactions on Robotics **23**(1), 34–46 (2007)
15. Grzonka, S., Grisetti, G., Burgard, W.: Towards a navigation system for autonomous indoor flying. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 2878–2883 (2009)
16. 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: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 361–366 (2007)
17. He, R., Prentice, S., Roy, N.: Planning in information space for a quadrotor helicopter in a GPS-denied environment. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 1814–1820 (2008)
18. Hrabar, S., Sukhatme, G.: Vision-based navigation through urban canyons. Journal of Field Robotics **26**(5), 431–452 (2009)
19. Lee, G., Achtelik, M., Fraundorfer, F., Pollefeys, M., Siegwart, R.: A benchmarking tool for mav visual pose estimation. In: Proceedings of the International Conference on Control, Automation, Robotics and Vision (2010)
20. Lourakis, M.: levmar: Levenberg-marquardt non-linear least squares algorithms in C/C++. http://www.ics.forth.gr/~lourakis/levmar/. [Date of access: 24/07/2011]
21. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
22. Matsue, A., Hirosue, W., Tokutake, H., Sunada, S., Ohkura, A.: Navigation of Small and Lightweight Helicopter. Transactions of the Japan Society for Aeronautical and Space Sciences **48**(161), 177–179 (2005)
23. Newsome, S., Rodocker, J.: Effective technology for underwater hull and infrastructure inspection: The SeaBotix LBC. In: Proc. Oceans, pp. 1–6 (2009)
24. 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: Proc. IEEE Intl. Conf. on Emerging Technologies and Factory Automation (2010)
25. Ortiz, R.: FREAK: Fast Retina Keypoint. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 510–517 (2012)
26. Pounds, P., Mahony, R., Cork, P.: Modelling and control of a quadrotor robot. In: Proceedings of the Australasian Conference on Robotics and Automation (2006)
27. 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)
28. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: IEEE International Conference on Computer Vision, pp. 1508–1511 (2005)
29. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An Efficient Alternative to SIFT or SURF. In: International Conference on Computer Vision (2011)

Table 1 Performance measures for the SMA odometer and various combinations of feature detectors and descriptors. Roll ϕ and pitch θ angles are set from Vicon data. Height t_z also comes from Vicon data. Performance is measured as difference with ground truth data: (a) error in X [m], (b) error in Y [m], (c) error in yaw [deg], and (d) frames per second

(a)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	0.21	0.18	0.23	0.06	0.17
FAST-FREAK	0.18	0.10	0.16	0.10	0.13
FAST-ORB	0.16	0.08	0.09	0.05	0.08
GFTT-BRIEF	0.18	0.20	0.25	0.05	0.17
GFTT-FREAK	0.07	0.20	0.17	0.06	0.13
GFTT-ORB	0.17	0.10	0.13	0.05	0.10
ORB-ORB	0.30	0.23	0.26	0.05	0.19
SIFT-SIFT	0.24	0.28	0.36	0.06	0.24
SURF-SURF	0.35	0.39	0.46	0.07	0.32

(b)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	0.11	0.14	0.15	0.11	0.13
FAST-FREAK	0.11	0.13	0.16	0.08	0.12
FAST-ORB	0.10	0.07	0.07	0.14	0.09
GFTT-BRIEF	0.11	0.12	0.13	0.12	0.12
GFTT-FREAK	0.12	0.08	0.12	0.11	0.11
GFTT-ORB	0.09	0.06	0.06	0.11	0.08
ORB-ORB	0.12	0.11	0.12	0.10	0.11
SIFT-SIFT	0.12	0.16	0.18	0.13	0.15
SURF-SURF	0.13	0.16	0.19	0.11	0.16

(c)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	6.70	6.57	8.19	2.54	5.94
FAST-FREAK	5.84	4.04	5.54	2.25	4.20
FAST-ORB	5.14	2.44	2.90	2.08	2.75
GFTT-BRIEF	5.88	6.40	8.32	2.53	5.86
GFTT-FREAK	2.66	2.06	2.87	3.22	2.76
GFTT-ORB	5.28	2.95	3.42	2.27	3.13
ORB-ORB	8.43	7.52	8.88	2.59	6.60
SIFT-SIFT	7.37	9.76	12.55	1.69	8.08
SURF-SURF	9.18	11.80	15.42	1.51	9.72

(d)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	195.60	188.01	180.46	131.76	164.86
FAST-FREAK	100.66	123.02	108.05	75.40	97.51
FAST-ORB	125.61	204.93	126.36	100.10	128.25
GFTT-BRIEF	83.22	111.59	83.16	74.96	85.72
GFTT-FREAK	79.47	101.14	76.67	65.16	77.42
GFTT-ORB	85.76	106.21	80.00	72.90	83.15
ORB-ORB	56.77	64.89	52.53	44.12	52.35
SIFT-SIFT	18.87	20.05	18.54	15.72	17.93
SURF-SURF	14.94	15.19	14.29	12.62	13.99

Table 2 Performance measures for the GEN odometer and various combinations of feature detectors and descriptors. Roll ϕ and pitch θ angles are set from IMU data. Height t_z comes from Vicon data. Performance is measured as difference with ground truth data: (a) error in X [m], (b) error in Y [m], (c) error in yaw [deg], and (d) frames per second

(a)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	0.10	0.31	0.37	0.14	0.26
FAST-FREAK	0.14	0.27	0.38	0.20	0.27
FAST-ORB	0.15	0.32	0.35	0.14	0.26
GFTT-BRIEF	0.23	0.43	0.45	0.13	0.32
GFTT-FREAK	0.28	0.47	0.49	0.14	0.36
GFTT-ORB	0.26	0.33	0.34	0.11	0.26
ORB-ORB	0.12	0.07	0.13	0.11	0.11
SIFT-SIFT	0.09	0.15	0.25	0.16	0.18
SURF-SURF	0.12	0.18	0.19	0.16	0.17

(b)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	0.16	0.10	0.09	0.12	0.11
FAST-FREAK	0.17	0.11	0.10	0.09	0.11
FAST-ORB	0.14	0.08	0.09	0.14	0.11
GFTT-BRIEF	0.16	0.08	0.07	0.12	0.10
GFTT-FREAK	0.17	0.10	0.09	0.12	0.11
GFTT-ORB	0.12	0.09	0.09	0.11	0.10
ORB-ORB	0.13	0.09	0.08	0.11	0.10
SIFT-SIFT	0.18	0.16	0.16	0.13	0.15
SURF-SURF	0.17	0.16	0.16	0.12	0.15

(c)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	1.85	2.93	2.57	1.61	2.30
FAST-FREAK	2.04	4.85	4.20	1.50	3.34
FAST-ORB	2.28	7.49	7.91	1.13	5.21
GFTT-BRIEF	2.60	5.31	4.58	2.06	3.81
GFTT-FREAK	5.88	10.54	11.78	3.04	8.26
GFTT-ORB	3.13	8.78	9.31	1.65	6.27
ORB-ORB	3.57	1.82	2.03	1.61	2.00
SIFT-SIFT	3.45	4.61	7.21	2.32	4.73
SURF-SURF	3.68	6.44	7.46	2.09	5.22

(d)

	DS1	DS2	DS3	DS4	All
no. frames	640	1653	2373	2026	6692
FAST-BRIEF	143.53	137.24	131.61	101.59	122.84
FAST-FREAK	113.90	112.79	107.67	75.62	96.84
FAST-ORB	132.09	131.07	126.59	101.16	119.01
GFTT-BRIEF	83.80	83.80	82.66	74.75	80.46
GFTT-FREAK	75.26	76.69	76.41	65.41	72.67
GFTT-ORB	70.54	80.23	79.56	73.53	76.87
ORB-ORB	53.83	54.19	52.62	44.30	50.23
SIFT-SIFT	18.76	18.99	18.53	15.73	17.70
SURF-SURF	14.89	14.58	14.23	12.60	13.83

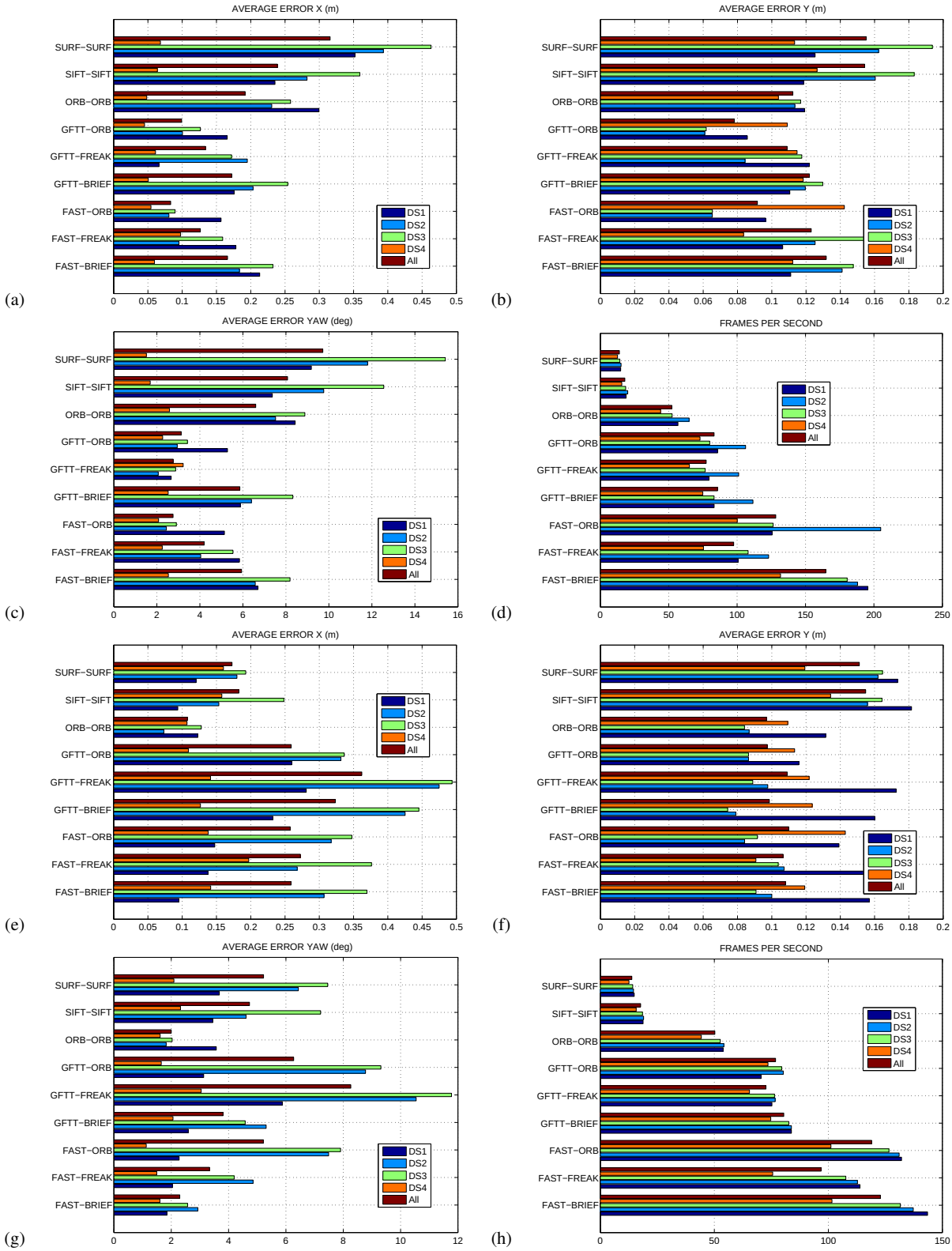


Fig. 8 Performance measures for the GEN odometer and various combinations of feature detectors and descriptors. Roll ϕ and pitch θ angles are set from Vicon (a-d) or from IMU data (e-h). Height t_z comes from Vicon data. Performance is measured as difference with ground truth data: (a,e) error in X [m], (b,f) error in Y [m], (c,g) error in yaw [deg], and (d,h) frames per second

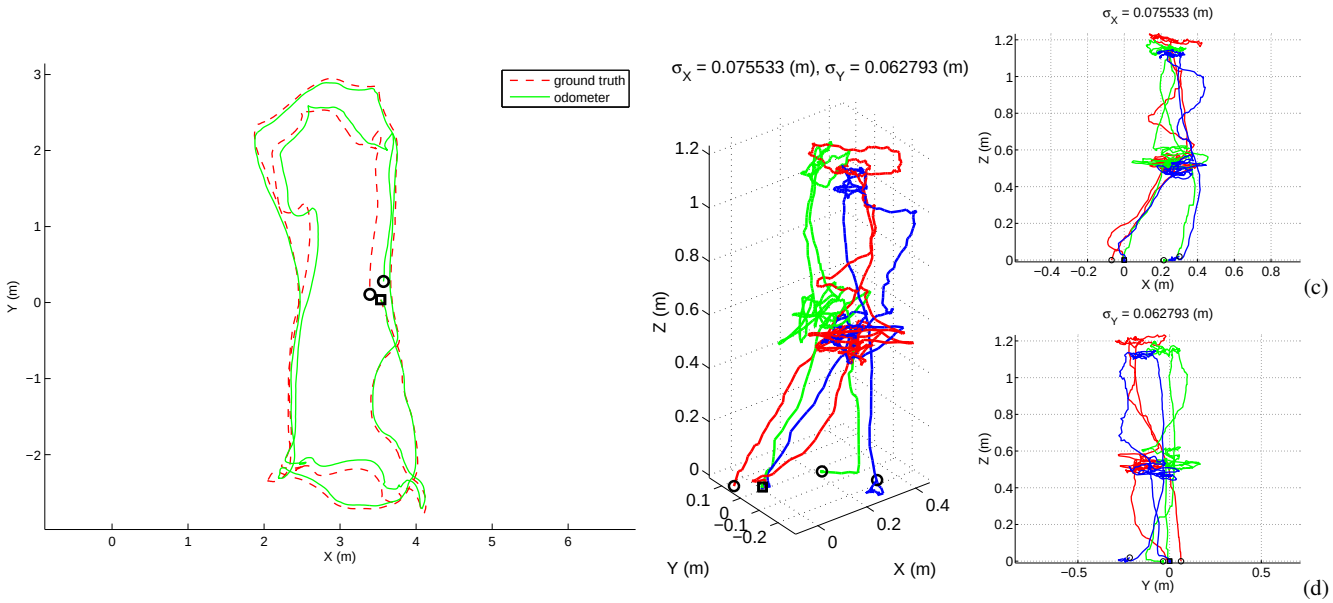


Fig. 9 (a) 2D plot for DS2 of estimated and ground truth paths for the GEN odometer, using IMU data, and ORB as detector and descriptor. (b) 3D plot of estimated paths for the online experiment, using FAST as detector and FREAK as descriptor, (c) XZ projection, (d) YZ projection.

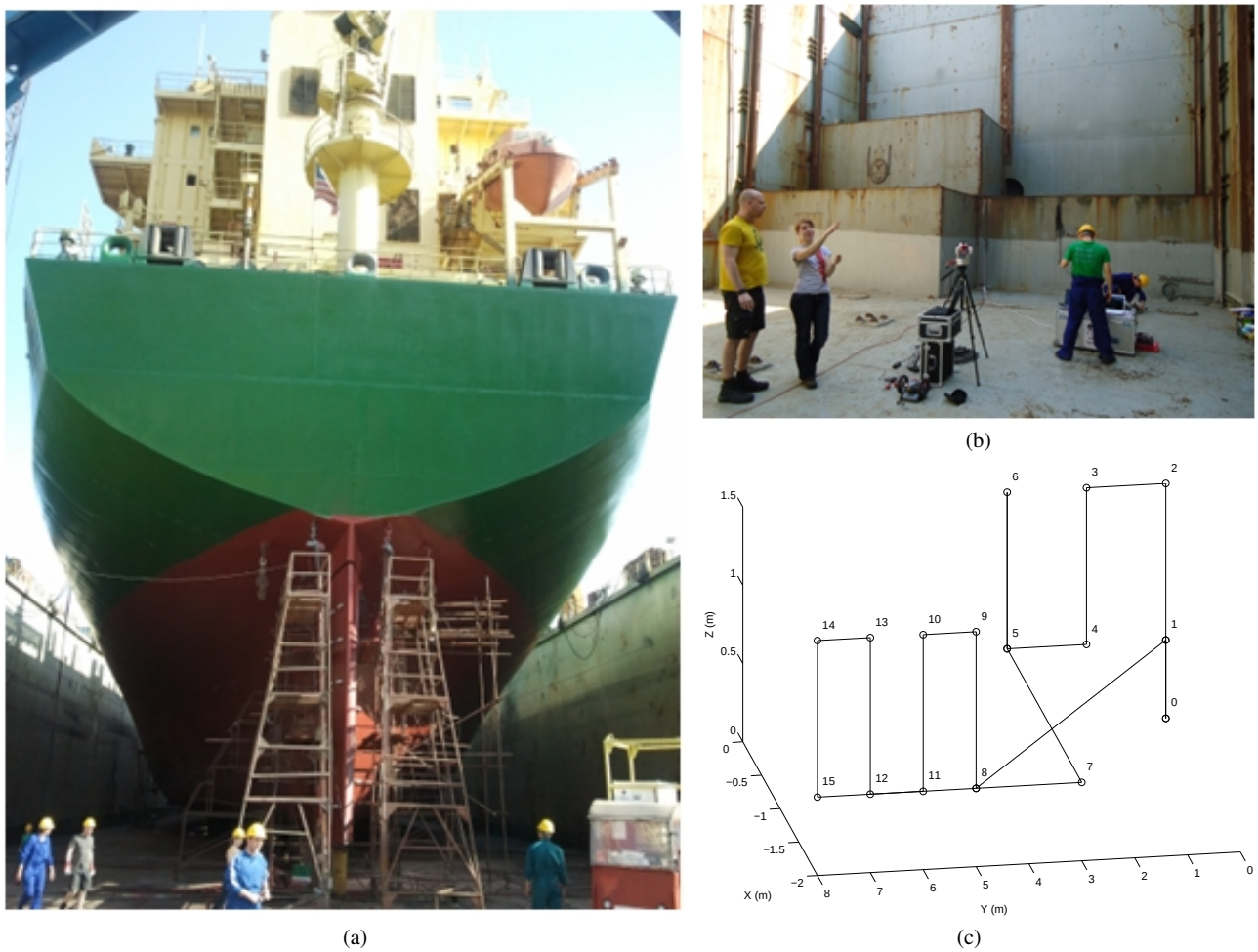
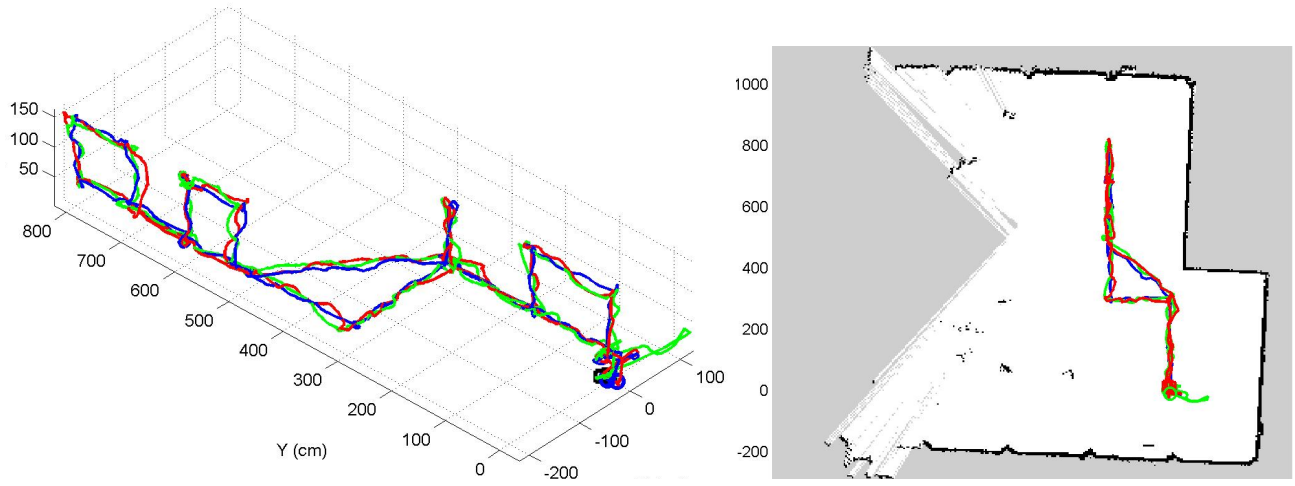


Fig. 10 (a) Containership and (b) hold where the field experiment took place. (c) Waypoints sequence for the online experiment: 0 / 1 / 2 / 3 / 4 / 5 / 6 / 5 / 7 / 8 / 9 / 10 / 11 / 12 / 13 / 14 / 15 / 8 / 1 / 0



(a) (b)

Fig. 11 Three executions of the field experiment, shown in different colours: (a) 3D plot of estimated MAV trajectories, (b) map produced by the laser-based SLAM module with the estimated MAV 2D paths superimposed

Fig. 12 Pictures taken (from left to right and from top to bottom) at waypoints 1, 2, 3, 4, 8 and 9 during the first repetition of the field experiment