On the Use of UAVs for Vessel Inspection Assistance*

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

Dep. of Mathematics and Computer Science, University of Balearic Islands

Cra. Valldemossa, km 7.5, 07122 Palma de Mallorca, Spain

email: alberto.ortiz@uib.es, xisco.bonnin@uib.es, emilio.garcia@uib.es

Abstract

Vessel maintenance entails periodic visual inspections of internal and external parts of the vessel hull in order to detect the typical defective situations affecting metallic structures, such as cracks, coating breakdown, corrosion, etc. The main goal of the EU-FP7 project MINOAS is the automation of the inspection process, currently undertaken by human surveyors, by means of a fleet of robotic agents. This paper overviews a UAV to be used as part of this fleet, and succinctly describes its control architecture as well as a self-localization solution for this vehicle. Promising experimental results are discussed in the experimental results section.

keywords: Vessels maintenance, UAV, visual inspection, visual odometry

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. Structural failures are a major cause of accidents, and can usually be prevented through timely maintenance. As such, vessels undergo annual inspections, with intensive Special and Docking Surveys every five years, which ensures that the hull structure and related piping are all in satisfactory condition and are fit for the intended use over the next five years.

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. As a result, although accidents are extremely rare, when they do arise they can have serious consequences. Due to these complications, the total cost of a single surveying can exceed \$1M once you factor in the

*This work is partially supported by projects SCP8-GA-2009-233715 (EU FP7 MINOAS), AAEE 0108/09 (Conselleria d'Innovacio, Interior i Justicia, Govern Balear), and FEDER funding.

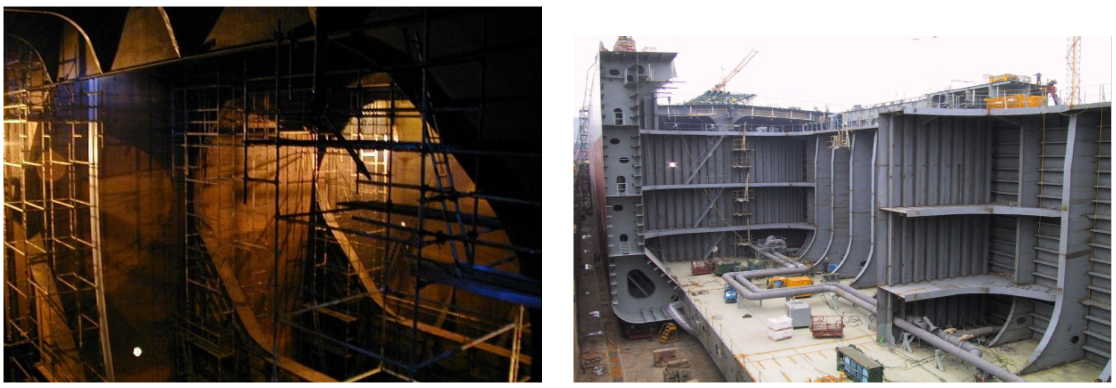

Figure 1: (left) Staging required during a vessel inspection. (right) Oil tanker in shipyard during construction.

vessel's preparation, use of yard's facilities, cleaning, ventilation, and provision of access arrangements (see figure 1[left]). In addition, the owners experience significant lost opportunity costs while the ship is inoperable.

The main objective of the EU FP7 project MINOAS [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 [1] for a detailed discussion).

Contrary to similar past projects (ROTIS and ROTIS-II), 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).

This paper presents the aerial platform adopted within the context of MINOAS, as well as discusses on the self-localization requirements of the application and how they are planned to be solved. As part of the development of the navigation strategy, an *almost-closed-form* solution to visual odometry is described, together with a set of experimental results which have been obtained from the different experiments performed.

The rest of the paper is structured as follows: section 2 discusses on the requirements imposed by the application and describes the UAV solution finally adopted, section 3 describes the intended control architecture and, particularly, focuses on the navigation strategy, section 4 describes a monocoluar odometer that has been developed as part of the localization strategy, and, finally, section 5 provides experimental results about the performance of this odometer.

[1] Marine INspection rObotic Assistant System, http://www.minoasproject.eu

2 PLATFORM DESCRIPTION

As part of the MINOAS concept, the aerial platforms are intended to provide detailed surveys of the vertical structures that can be found in vessel holds (see figure 1). Therefore, the main requirements stem directly from the very nature of the inspection process: the vehicle must be able to perform vertical, stationary and low speed flight, as well as permit indoor flight. These requirements rapidly discard fixed-wing aircrafts and focus the search on helicopter-type UAVs, naturally capable of manoeuvres such as hovering and vertical take-off and landing (VTOL).

Among the different kinds of helicopter designs that have been proposed, multi-rotor configurations present several advantages over comparably scale helicopters (see, for instance, [2]): (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.

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.

A Pelican quadrotor from Ascending Technologies is the platform finally chosen to implement the aerial vehicle for the MINOAS project. This is a 50cm-diameter *micro aerial vehicle* (MAV) with 10-inch propellers, able to carry a payload of 500g, and fitted with an *inertial measuring unit* (IMU) and a barometric pressure sensor. The vehicle is provided with attitude stabilization running on an ARM7 microprocessor [3].

Furthermore, for the application at hand, the MAV has been outfitted with the lightweight Hokuyo URG-04LX-UG01 laser scanner, a front-looking stereo rig comprising two lightweight uEye 1226-LE-C cameras and a third uEye 1226-LE-C bottom looking unit. For onboard processing, the vehicle carries a Kontron pITX-SP board equipped with an Intel Atom 1.6GHz processor and 2GB RAM. See figure 2(left) for a picture of the MAV in its current configuration. As can be observed, some of the beams of the laser scanner are deflected by two mirrors on the right and the left sides of the quadrotor to provide two height estimates.

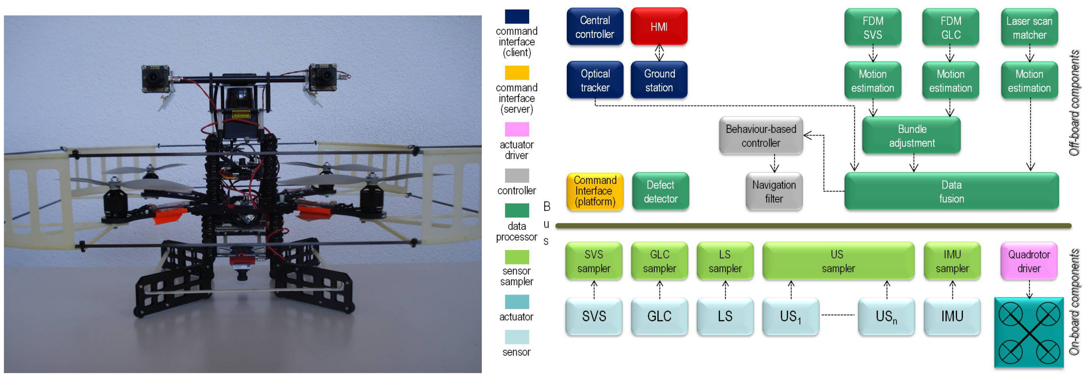

Figure 2: (left) MAV to be used in project MINOAS, in its current configuration. (right) The UAV control architecture expressed by means of GLOC3 generic components. Although not explicitly indicated, all the components are connected to the communication bus. Dotted lines represent logical connections between components. (FDM stands for *feature detection and matching*.)

3 ABOUT THE CONTROL ARCHITECTURE

As part of the inspection process, the UAV is expected to achieve a set of waypoints for acquiring images of the underlying metallic structures to provide the surveyor with an overall view of the state of the vessel. Those images must as well be tagged with pose information, so that the same UAV, or any other member of the robot fleet, can, on demand of the surveyor, re-visit the area for acquiring close-up images or else taking other sensor data, e.g. measure the thickness of the hull.

Hence, the motion capabilities of the MINOAS flying robot essentially comprise *waypoint navigation* and automatic *take-off* and *landing*. To this end, for localization purposes, the platform is intended to implement a navigation strategy based on combining, by selection or by fusion, visual odometers using the front-looking *stereo vision system* (SVS) and the *ground-looking camera* (GLC), as well as a laser scan matching-based odometer (LS). The robot pose estimation produced is finally complemented with the estimation provided by an external optical follower able to track a vehicle fitted with an ultra-bright LED, and comprising a camera and a laser pointer (see [4] for a first version of this device, which is currently being used to track a magnetic crawler). This redundant strategy is intended to provide robust positioning information able to tolerate the failure of any of the positioning subsystems in case of the vehicle getting out of the line of sight of the optical tracker or in case the vehicle motion cannot be estimated from the available sensor data, i.e. lack of features when imaging non-textured surfaces or lack of distinctive structures within laser scans in long straight walls. Although the vehicle is supported by a base station, due to the lossy nature of wireless communications, images are not transmitted, but are processed on-board; subsequent calculations can, however, be performed at the base station.

The control software is implemented following the *Generic LOosely-Coupled Component-based Control software architecture* (GLOC3) [5], while the *Robot Operating System* (ROS) is used as the middleware for software distribution. Figure 2(right) shows the control architecture for the whole system expressed in terms of the generic components of GLOC3.

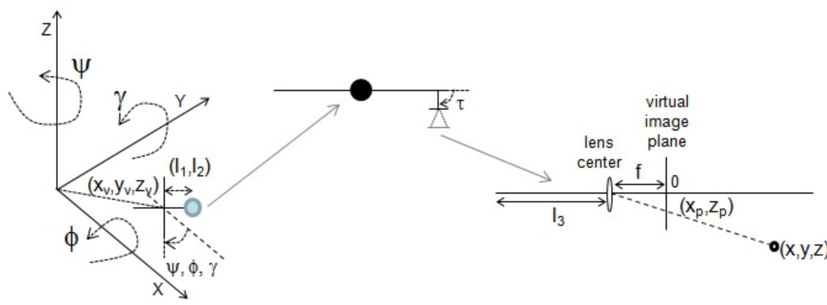

Figure 3: Coordinate frames assumed and symbols used by the visual odometer.

4 A VISUAL ODOMETRY SOLUTION

As part of the effort on the development of the UAV localization strategy for project MINOAS, this section describes a fast visual odometer for motion estimation using local image features for image point matching between camera frames and the *M-estimator SAmple and Consensus* (MSAC) [6], a RANSAC variant, for rejecting matchings disagreeing with an inter-frame homography transformation. Due to its robustness to motion blur and large displacements, we use the pyramidal implementation of the Kanade-Lucas-Tomasi (KLT) optical flow tracker [7] for feature tracking across images. Features are first located by means of the *good-features-to-track detector* (OpenCV implementation), replacing lost features as needed. The whole feature detection and tracking solution provides an adequate level of performance at reduced execution times, compatible with the computational power of the onboard processor.

Briefly speaking, once a set of matched features is available, the method backprojects features into the world using previous knowledge of the camera position with respect to the center of the vehicle (l_1, l_2, l_3), the camera focal distance f, the camera tilt τ, and the vehicle height z_v and pitch ϕ and roll γ angles, all three supplied by onboard sensors. The odometer finally becomes into a least squares problem in almost-closed-form solution that allows estimating motion in x, y and yaw, Δx, Δy and $\Delta\psi$. Figure 3 illustrates the different symbols involved in the derivation of the visual odometer.

According to figure 3, equation 1 transforms from world (x, y, z) to image coordinates (x_p, z_p) [2]:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{f} & 0 & 1 \end{bmatrix}}_{P} \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -(l_3+f) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\tau & s\tau & 0 \\ 0 & -s\tau & c\tau & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_x(-\tau)} \begin{bmatrix} 1 & 0 & 0 & -l_1 \\ 0 & 1 & 0 & -l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{T_R^C}$$

[2]In the following, $c\alpha$ and $s\alpha$ stand for, respectively, $\cos\alpha$ and $\sin\alpha$.

$$\overbrace{\underbrace{\begin{bmatrix} c\gamma & 0 & s\gamma & 0 \\ 0 & 1 & 0 & 0 \\ -s\gamma & 0 & c\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_y(-\gamma)} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\phi & s\phi & 0 \\ 0 & -s\phi & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_x(-\phi)}}^{T^R_{Wb}} \overbrace{\underbrace{\begin{bmatrix} c\psi & s\psi & 0 & 0 \\ -s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_z(-\psi)} \begin{bmatrix} 1 & 0 & 0 & -x_v \\ 0 & 1 & 0 & -y_v \\ 0 & 0 & 1 & -z_v \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{T^R_{Wa}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

where y_p turns out to be a free parameter that determines the plane where the 2D point (x_p, z_p) is located, so that $y_p = -\infty$ and $y_p = 0$ correspond to, respectively, the lens center and the sensor plane. The corresponding inverse perspective transformation is thus:

$$(x, y, z, 1)^T = \left(T^R_{Wa}\right)^{-1} \left(T^R_{Wb}\right)^{-1} \left(T^C_R\right)^{-1} P^{-1} (x_p, y_p, z_p, 1)^T \tag{2}$$

Equation 2 next becomes into the following 3D line in world coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} + \begin{bmatrix} -M_2\, f\, c\psi + M_6\, f\, s\psi \\ -M_6\, f\, c\psi - M_2\, f\, s\psi \\ -M_{10}\, f \end{bmatrix} + \lambda \begin{bmatrix} (M_1\, x_p + M_2\, f + M_3\, z_p + M_4) c\psi - (M_5\, x_p + M_6\, f + M_7\, z_p + M_8) s\psi \\ (M_5\, x_p + M_6\, f + M_7\, z_p + M_8) c\psi + (M_1\, x_p + M_2\, f + M_3\, z_p + M_4) s\psi \\ M_9\, x_p + M_{10}\, f + M_{11}\, z_p + M_{12} \end{bmatrix} \tag{3}$$

where:

$$\begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} \\ 0 & -\frac{1}{f} & 0 & 1 \end{bmatrix} = \left(T^R_{Wb}\right)^{-1} \left(T^C_R\right)^{-1} P^{-1}$$

For image features corresponding to points lying on the floor, $z = 0$ and consequently:

$$\lambda = -\frac{z_v - M_{10}\, f}{M_9\, x_p + M_{10}\, f + M_{11}\, z_p + M_{12}} \tag{4}$$

Accordingly, at time k, equation 3 can be written as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \end{bmatrix}^{(k)} + \left(\begin{bmatrix} C & -D \\ D & C \end{bmatrix}^{(k)} + \lambda^{(k)} \begin{bmatrix} A & -B \\ B & A \end{bmatrix}^{(k)} \right) \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} = \begin{bmatrix} x_v \\ y_v \end{bmatrix}^{(k)} + \begin{bmatrix} C + \lambda\, A & -(D + \lambda\, B) \\ D + \lambda\, B & C + \lambda\, A \end{bmatrix}^{(k)} \begin{bmatrix} c\psi \\ s\psi \end{bmatrix}^{(k)} \tag{5}$$

with:

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

$$A^{(k)} = M_1^{(k)}\, x_p + M_2^{(k)}\, f + M_3^{(k)}\, z_p + M_4^{(k)} \qquad C^{(k)} = -M_2^{(k)}\, f$$

$$B^{(k)} = M_5^{(k)}\, x_p + M_6^{(k)}\, f + M_7^{(k)}\, z_p + M_8^{(k)} \qquad D^{(k)} = -M_6^{(k)}\, f$$

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

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_v^{(k)} + \Delta x_v \\ y_v^{(k)} + \Delta y_v \end{bmatrix} + \left(\begin{bmatrix} C & -D \\ D & C \end{bmatrix}^{(k+1)} + \lambda^{(k+1)} \begin{bmatrix} A & -B \\ B & A \end{bmatrix}^{(k+1)} \right) \begin{bmatrix} c(\psi^{(k)} + \Delta\psi) \\ s(\psi^{(k)} + \Delta\psi) \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} x_v^{(k)} + \Delta x_v \\ y_v^{(k)} + \Delta y_v \end{bmatrix} + \left(\begin{bmatrix} C & -D \\ D & C \end{bmatrix}^{(k+1)} + \lambda^{(k+1)} \begin{bmatrix} A & -B \\ B & A \end{bmatrix}^{(k+1)} \right) \begin{bmatrix} c\psi^{(k)} c\Delta\psi - s\psi^{(k)} s\Delta\psi \\ s\psi^{(k)} c\Delta\psi + c\psi^{(k)} s\Delta\psi \end{bmatrix}$$

$$= \begin{bmatrix} x_v^{(k)} + \Delta x_v \\ y_v^{(k)} + \Delta y_v \end{bmatrix} + \begin{bmatrix} (C + \lambda\, A)^{(k+1)} c\psi^{(k)} - (D + \lambda\, B)^{(k+1)} s\psi^{(k)} & -(D + \lambda\, B)^{(k+1)} c\psi^{(k)} - (C + \lambda\, A)^{(k+1)} s\psi^{(k)} \\ (D + \lambda\, B)^{(k+1)} c\psi^{(k)} + (C + \lambda\, A)^{(k+1)} s\psi^{(k)} & (C + \lambda\, A)^{(k+1)} c\psi^{(k)} - (D + \lambda\, B)^{(k+1)} s\psi^{(k)} \end{bmatrix} \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix}.$$

Combining equations 5 and 6 we can write:

$$\begin{bmatrix} (C + \lambda\, A) c\psi - (D + \lambda\, B) s\psi \\ (D + \lambda\, B) c\psi + (C + \lambda\, A) s\psi \end{bmatrix}^{(k)} =$$

$$\begin{bmatrix} \Delta x_v \\ \Delta y_v \end{bmatrix} + \begin{bmatrix} (C + \lambda\, A)^{(k+1)} c\psi^{(k)} - (D + \lambda\, B)^{(k+1)} s\psi^{(k)} & -(D + \lambda\, B)^{(k+1)} c\psi^{(k)} - (C + \lambda\, A)^{(k+1)} s\psi^{(k)} \\ (D + \lambda\, B)^{(k+1)} c\psi^{(k)} + (C + \lambda\, A)^{(k+1)} s\psi^{(k)} & (C + \lambda\, A)^{(k+1)} c\psi^{(k)} - (D + \lambda\, B)^{(k+1)} s\psi^{(k)} \end{bmatrix} \begin{bmatrix} c\Delta\psi \\ s\Delta\psi \end{bmatrix} \tag{7}$$

Equation 8 can be next obtained rearranging equation 7:

$$\begin{bmatrix} 1 & 0 & (C+\lambda\,A)^{(k+1)}c\psi^{(k)} - (D+\lambda\,B)^{(k+1)}s\psi^{(k)} & -(D+\lambda\,B)^{(k+1)}c\psi^{(k)} - (C+\lambda\,A)^{(k+1)}s\psi^{(k)} \\ 0 & 1 & (D+\lambda\,B)^{(k+1)}c\psi^{(k)} + (C+\lambda\,A)^{(k+1)}s\psi^{(k)} & (C+\lambda\,A)^{(k+1)}c\psi^{(k)} - (D+\lambda\,B)^{(k+1)}s\psi^{(k)} \end{bmatrix} \begin{bmatrix} \Delta x_v \\ \Delta y_v \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} (C+\lambda\,A)c\psi - (D+\lambda\,B)s\psi \\ (D+\lambda\,B)c\psi + (C+\lambda\,A)s\psi \end{bmatrix}^{(k)} \tag{8}$$

A least squares framework can be devised now from equation 8 to determine Δx_v, Δy_v, $c\Delta\psi$ and $s\Delta\psi$ if for a number of ground points $(x_i, y_i, z_i = 0), i = 1, \ldots, n$, their image projections $(x_{p,i}, z_{p,i})$ have been matched between frames k and $k+1$ [3]:

$$\min_{\substack{\Delta x_v \\ \Delta y_v \\ c\Delta\psi,\, s\Delta\psi}} \left(U \begin{bmatrix} \Delta x_v \\ \Delta y_v \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} - V \right)^2 \tag{9}$$

where:

$$U = \begin{bmatrix} 1 & 0 & P_1 & -Q_1 \\ 0 & 1 & Q_1 & P_1 \\ 1 & 0 & P_2 & -Q_2 \\ 0 & 1 & Q_2 & P_2 \\ \multicolumn{4}{c}{\cdots\cdots\cdots\cdots} \\ 1 & 0 & P_n & -Q_n \\ 0 & 1 & Q_n & P_n \end{bmatrix}^{(k+1)}, \quad V = \begin{bmatrix} P_1 \\ Q_1 \\ P_2 \\ Q_2 \\ \cdots \\ P_n \\ Q_n \end{bmatrix}^{(k)}, \quad \begin{aligned} P_i^{(k+1)} &= (C_i + \lambda_i\,A_i)^{(k+1)}c\psi^{(k)} - (D_i + \lambda_i\,B_i)^{(k+1)}s\psi^{(k)} \\ Q_i^{(k+1)} &= (D_i + \lambda_i\,B_i)^{(k+1)}c\psi^{(k)} + (C_i + \lambda_i\,A_i)^{(k+1)}s\psi^{(k)} \\ P_i^{(k)} &= (C_i + \lambda_i\,A_i)^{(k)}c\psi^{(k)} - (D_i + \lambda_i\,B_i)^{(k)}s\psi^{(k)} \\ Q_i^{(k)} &= (D_i + \lambda_i\,B_i)^{(k)}c\psi^{(k)} + (C_i + \lambda_i\,A_i)^{(k)}s\psi^{(k)} \end{aligned} \tag{10}$$

By means of the pseudo-inverse $U^+ = (U^TU)^{-1}U^T$, the solution to the least squares problem is:

$$\begin{bmatrix} \Delta x_v \\ \Delta y_v \\ c\Delta\psi \\ s\Delta\psi \end{bmatrix} = \begin{bmatrix} \mathcal{AF} - \mathcal{CD} - \mathcal{BG} \\ \mathcal{AG} + \mathcal{BF} - \mathcal{CE} \\ \mathcal{AD} - n\mathcal{F} + \mathcal{BE} \\ \mathcal{AE} - n\mathcal{G} - \mathcal{BD} \end{bmatrix} / (\mathcal{A}^2 + \mathcal{B}^2 - n\mathcal{C}) \quad \left| \quad \begin{aligned} \mathcal{A} &= \sum P_i^{(k+1)} \\ \mathcal{B} &= \sum Q_i^{(k+1)} \\ \mathcal{C} &= \sum \left(P_i^{(k+1)}\right)^2 + \left(Q_i^{(k+1)}\right)^2 \end{aligned} \quad \right| \quad \begin{aligned} \mathcal{D} &= \sum P_i^{(k)} \\ \mathcal{E} &= \sum Q_i^{(k)} \\ \mathcal{F} &= \sum P_i^{(k+1)}P_i^{(k)} + Q_i^{(k+1)}Q_i^{(k)} \\ \mathcal{G} &= \sum P_i^{(k+1)}Q_i^{(k)} - Q_i^{(k+1)}P_i^{(k)} \end{aligned} \tag{11}$$

A further non-linear refinement step follows next to ensure $c\Delta\psi^2 + s\Delta\psi^2 = 1$, which is not guaranteed by the least squares solution. To this end, equation 9 is re-arranged as a non-linear optimization problem aiming at minimizing for $\Delta x_v, \Delta y_v, \Delta\Psi$ and the Levenberg-Marquardt algorithm is used [8]. The odometer can be globally regarded as an *almost-closed-form* solution because experiments have shown that a few iterations (2-5) are enough to attain the minimum.

A standard Kalman filter assuming constant acceleration constitutes the final stage of the odometer. The state is given by $(\dot{\mathbf{x}}, \ddot{\mathbf{x}})^T = (\dot{x}_v, \dot{y}_v, \dot{\psi}, \ddot{x}_v, \ddot{y}_v, \ddot{\psi})^T$ and the measure by $\Delta\mathbf{x} = (\Delta x_v, \Delta y_v, \Delta\psi)^T$:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^t = \overbrace{\begin{bmatrix} I & \Delta_t I \\ 0 & I \end{bmatrix}}^{A} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^{t-1} + Q\,, \qquad \begin{bmatrix} \Delta\mathbf{x} \end{bmatrix}^t = \overbrace{\begin{bmatrix} \Delta_t I & 0 \end{bmatrix}}^{H} \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}^t + R \tag{12}$$

[3] Observe that two matchings are enough to estimate Δx_v, Δy_v, $c\Delta\psi$ and $s\Delta\psi$.

Table 1: Average absolute error in vehicle pose: x_v (2nd col), y_v (3rd col), ψ (4th col), execution time (5th col) and frames processed per second (6th col).

	# frames	ϵ_{x_v} (cm)	ϵ_{y_v} (cm)	ϵ_{ψ} (^{o})	ms	fps
DS1	641	6.17	8.43	0.66	11.61	86.15
		12.10	5.91	2.75	10.82	92.38
DS2	1681	8.56	8.61	0.56	10.99	90.98
		11.32	19.22	7.11	10.12	98.78
DS3	2401	15.37	7.58	0.57	10.77	92.82
		11.65	30.09	10.39	9.98	100.21
DS4	2041	6.50	4.41	0.25	12.16	82.21
		6.91	7.84	3.94	9.30	107.56

To finish, figure 4 summarizes the structure of the visual odometer.

5 EXPERIMENTAL RESULTS AND CONCLUSIONS

This section reports the performance results obtained for the visual odometry approach that has been described. To this end, the publicly available datasets *1LoopDown* (DS1), *2LoopsDown* (DS2), *3Loops-Down* (DS3) and *hoveringDown* (DS4) made public by the EU-FP7 project sFly [4] have been used. The datasets were collected from a quadrotor flying 1, 2 and 3 loop sequences and hovering within a space of approximately 1m $\times$ 1m $\times$ 1m. These datasets comprise ground truth from a Vicon system and images from a ground-looking camera, as well as height and IMU data (accelerations, attitude rates, absolute angles and absolute headings). See [9] for a detailed description of the datasets.

Figure 5 and table 1 provide quantitative data from the performance evaluation. For every dataset DSi, the true height provided by the Vicon was used, and the experiment was run twice: first feeding the odometer with the roll, pitch and yaw angles from the ground truth (each first row of the table), and second emulating more realistic conditions, in which the pitch and roll angles came from the IMU, while the yaw angle, although available, was not used (each second row of the table). As can be observed, best results are in general obtained if the odometer is provided with the yaw angle, although the rule is

[4]http://projects.asl.ethz.ch/sfly/doku.php?id=mav_datasets

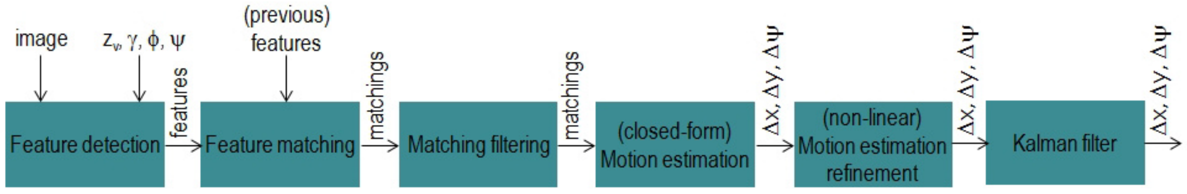
image
z_v, γ, ϕ, ψ
(previous) features
features
matchings
matchings
$\Delta x, \Delta y, \Delta\psi$
$\Delta x, \Delta y, \Delta\psi$
$\Delta x, \Delta y, \Delta\psi$
Feature detection
Feature matching
Matching filtering
(closed-form) Motion estimation
(non-linear) Motion estimation refinement
Kalman filter

Figure 4: Summary of the visual odometer.

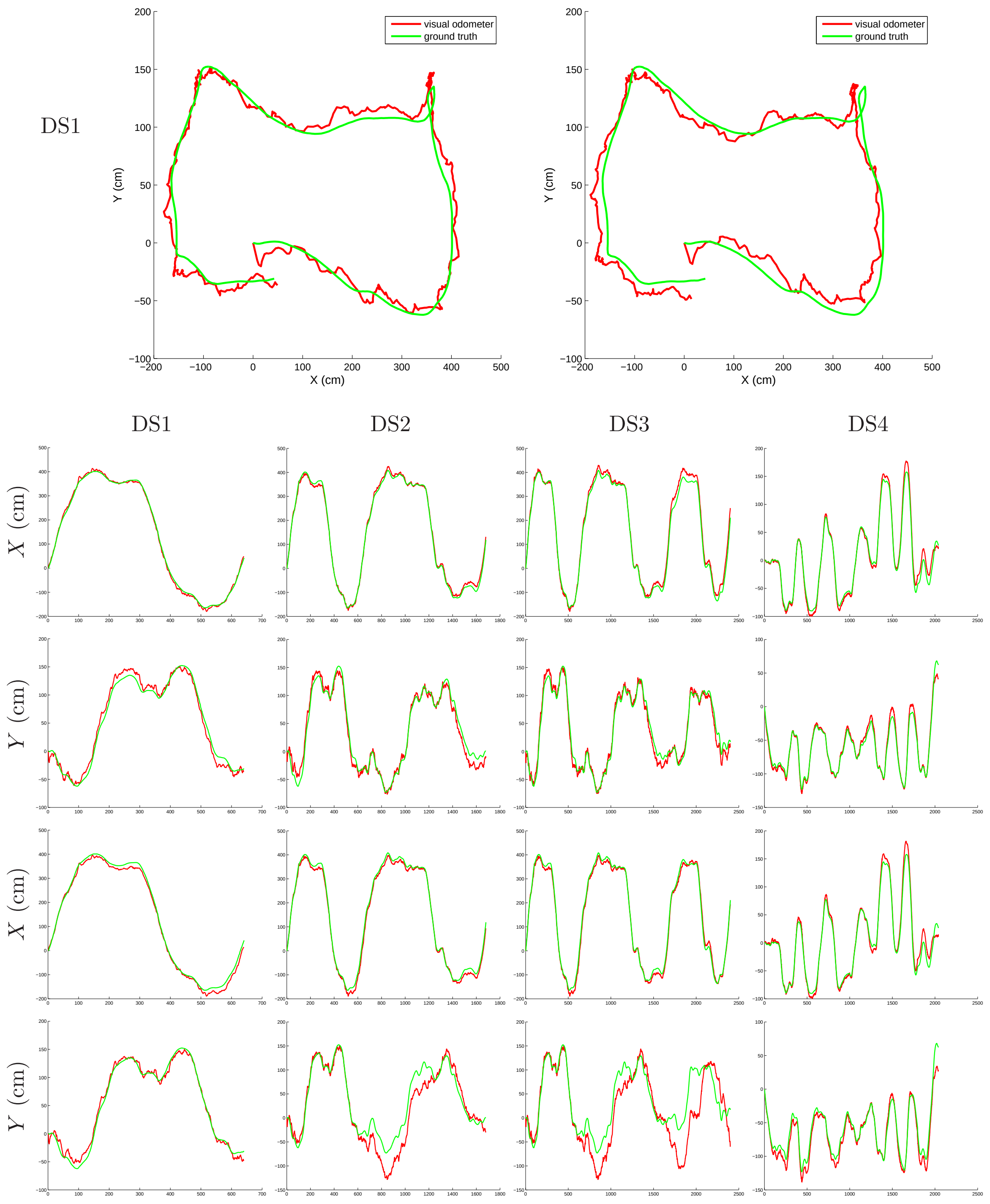

Figure 5: [top] Plots of xy estimations for DS1, feeding (left) and not feeding (right) the VO with true yaw. [bottom] Plots of the x and y estimations against the respective ground truth for the different datasets, feeding (upper) and not feeding (lower) the VO with true yaw. The green lines correspond to the ground truth and the red lines to the estimation.

not always met. In any case, the error in x is never larger than 15 cm, the error in y is never larger than 30 cm and, finally, the error in ψ is less than $10°$. Regarding the execution time, on average it is around 10.5 ms per frame, or approximately 95 frames per second. These times correspond to a dual-core Intel Core i5 @2.53GHz processor.

6 ACKNOWLEDGEMENTS

The authors are very thankful to Kontron and Quality By Measurement (QBM) for supplying the onboard Intel Atom board.

REFERENCES

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

[2] P. Pounds, R. Mahony, and P. Cork. Modelling and control of a quad-rotor robot. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2006.

[3] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proc. of IEEE International Conference on Robotcis and Automation*, pages 361–366, 2007.

[4] Markus Eich and Thomas Vögele. Design and control of a lightweight magnetic climbing robot for vessel inspection. In *Proc. of IEEE Mediterranean Conference on Control and Automation*, pages 1200–1205, 2011.

[5] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J.P. Beltran. A control software architecture for autonomous unmanned vehicles inspired in generic components. In *Proc. of IEEE Mediterranean Conference on Control and Automation*, pages 1217–1222, 2011.

[6] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[7] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. Description of the algorithm, 2000.

[8] M. Lourakis. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004. [Accessed on 24 July 2011.].

[9] G.H. Lee, M. Achtelik, F. Fraundorfer, M. Pollefeys, and R. Siegwart. A benchmarking tool for mav visual pose estimation. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, 2010.