J Real-Time Image Proc (2019) 16:1189–1205
DOI 10.1007/s11554-016-0613-z

ORIGINAL RESEARCH PAPER

Real-time smart and standalone vision/IMU navigation sensor

Lounis Chermak[1] · Nabil Aouf[1] · Mark Richardson[1] · Gianfranco Visentin[2]

Received: 18 August 2015 / Accepted: 7 June 2016 / Published online: 22 June 2016

Abstract In this paper, we present a smart, standalone, multi-platform stereo vision/IMU-based navigation system, providing ego-motion estimation. The real-time visual odometry algorithm is run on a nano ITX single-board computer (SBC) of 1.9 GHz CPU and 16-core GPU. High-resolution stereo images of 1.2 megapixel provide high-quality data. Tracking of up to 750 features is made possible at 5 fps thanks to a minimal, but efficient, features detection–stereo matching–feature tracking scheme runs on the GPU. Furthermore, the feature tracking algorithm benefits from assistance of a 100 Hz IMU whose accelerometer and gyroscope data provide inertial features prediction enhancing execution speed and tracking efficiency. In a space mission context, we demonstrate robustness and accuracy of the real-time generated 6-degrees-of-freedom trajectories from our visual odometry algorithm. Performance evaluations are comparable to ground truth measurements from an external motion capture system.

Keywords Real time · Smart multi-platform · Navigation system · Stereo visual odometry · IMU-assisted feature tracking

✉ Lounis Chermak
l.chermak@cranfield.ac.uk

[1] Defence Academy of the United Kingdom, Centre of Electronic Warfare, Cranfield University, Shrivenham SN6 8LA, UK

[2] European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), Keplerlaan 1, 2201, AZ, Noordwijk, The Netherlands

1 Introduction

This paper addresses development of precise vision-based navigation technology for space autonomous robotics missions. Space exploration and operational missions employ a range variety of mobile robots and robotic manipulators, instrumented with different kinds of sensors, on remote planetary surfaces, in orbit, or as assistants to astronauts. Current and future European Space Agency (ESA) missions involving these types of systems are numerous. To mention only a few missions that links directly the aimed result of this study, Mars Sample Return program with its ExoMars Rover Phase program, Lunar rovers with their robot exploration operation, Eurobot robot, and Tian aerobot. The utility of these robots in these types of missions depends on their ability to perform work, and to explore intelligently without frequent contact with the command control station. This requires capabilities for sensing and perception of surrounding unstructured and sometimes occluded environments. It also requires intelligent reasoning about perceptions to perform tasks in a reliable manner in such environments.

The need for intelligent space robots also results from the challenging environmental constraints and planetary surfaces where accurate localisation is critical for applications requiring permanent and precise positioning information. Therefore, the process of conceiving a dedicated standalone navigation system needs to consider environmental and technical constraints. This results not only in design restrictions regarding size and weight, but also in the choice of hardware components and sensors. Consequently, devices with significant power consumption or technologies subject to signal interruptions are thus discarded.

In recent years, many research projects looked at enhancing navigation systems compactness and accuracy.

Progress was made possible with the development of advanced software techniques and algorithms, but also thanks to a wider range of costly affordable off-the-shelve hardware and sensors. In spite of this, building a navigation system remains a challenging task because of the complex trade-off that has to be found between many criteria. Power consumption, autonomy, weight, and size represent the most predominant ones. Consequently, the considerations leading to the right choice of components has to follow a well-defined strategy that needs to take into account the performance compromises and system integration issues raised. Indeed, computational burden can be critical and real-time processing adds a further restriction in the choice of techniques and algorithms. With these constraints in mind, it appears wiser, in terms of autonomy and power consumption, to use passive sensors.

In this context, for perception of robotic platforms, visual sensors such as cameras present many advantages. Cheap, lightweight, smart, and benefiting from a large choice of types, cameras have been extensively utilised in research within numerous domains of applications such as navigation, medical imaging, and surveillance. Visual odometry (which can be described as the process enabling, through the analysis of images, the estimation of a platform relative motion) has yielded great achievements in the domain of navigation and localisation [1–4]; especially, stereo visual odometry which enables recover of 3D feature information via stereoscopy. Hence, less than one per cent relative error, achieved in the estimated trajectories, has been reported in the literature [5, 6]. This makes visual sensors one of the greatest technologies to be equipped to a navigation system, with the condition that the environment provides enough illumination, textured and overlapping static content, between subsequent acquired images.

Another type of sensor which has been often used for navigation tasks is the inertial measurement unit (IMU). This sensing platform measures linear and angular accelerations undergone, to continuously estimate the kinematics of a moving object. Integration of these values gives an estimate of the object's position, velocity, and orientation. Its main backward is that initial measurements are inherently drifty and need an external reference source of information, such as Global Positioning System (GPS) signal for instance, to correct the IMU's absolute position.

Our case study involves a GPS-denied environment, and the IMU sensor is not intended to be used as the main source of information for navigation task. That being said, the IMU sensor can be used as an excellent combination sensor with cameras. Indeed, visual and inertial sensors present different but very complementary information. IMU sensors, with higher data acquisition frequencies than visual sensors, can fill an eventual lack of visual data resulting from environmental conditions, as stated above. On the other hand, visual estimated motion brings the required reference to update IMU's absolute position and consequently preventing it to drift over the time. IMU data can be fused with visual data for pose estimation using filters such as Kalman and its variations [7–9]. It can also be used to assist the visual tracking process [10–12].

Choosing the sensors that suit the context is one side of the problem when developing a standalone visual odometry-based navigation sensor. The other side consists in the use and implementation of these sensors within the framework including algorithms, hardware design, and real-time performance. Therefore, this paper aims to describe the strategy and details to reach the goal of this study. Thus, after review of prior work in Sect. 2, the software architecture and the developed stereo visual odometry pipeline are explained in Sect. 3. Then, the navigation system hardware is described in Sect. 4. Finally, in Sect. 5, results in the experimental phase are demonstrated and discussed.

2 Related work

In this section, we focus principally on real-time stereo feature-based visual odometry approaches in the first instance, emphasising embedded/online contributions and then more specifically on the algorithms that are related to our methodology.

A multitude of works has contributed to improving visual odometry. Reviews and more recent tutorials give a detailed picture of the different visual odometry overall techniques [2, 3] and stereo visual approaches [13]. Despite the fact that a majority of works were not primarily aiming to tackle online performances, several contributions have proposed concrete solutions in this sense. Among the early contributions in visual odometry for instance, real-time performance was achieved in [14] in avoiding use of computationally expensive statistical methods to reject features outlier, using a strong Euclidean constraint combined with dense stereo to select inliers from a set of initial 3D correspondences. Nister's contribution [1] is one of the most influential works in this domain. It has introduced many improvements in the visual odometry pipeline, among which the use of RANSAC in the motion estimation stage for outlier rejection. Nister has also changed the processing of relative motion to 3D points projection into a two-dimensional camera pose problem, while it used to be seen as three-dimensional point registration problem only [1]. In fact, minimising 2D image re-projection errors is more accurate than minimising 3D feature correspondences errors. Following the same methodology of Nister, a successful implementation of visual odometry was made possible for Mars Exploration Rovers [15]. Even if the

entire visual odometry process latency was around 3 min (from image acquisition until the camera pose generation). This can be effectively qualified as real time, considering the relatively slow motion of the rover, and also the limited hardware specifications has to be taken into account. In [16], as part of the DARPA LAGR program, a stereo-based visual odometry approach was fused with IMU and GPS information in a Kalman filter scheme to avoid long-term drifts for outdoor long trajectories. In this work, real-time performance was achieved mainly thanks to a closed-form implementation of SVD computation matrix, which is the most time consuming task of their process. Howard [5] used the same methodology as [14] with an improved inliers selection scheme, based on groups of consistent matches enabling faster point-to-point comparison. Also, this implementation gives impressive position errors which are lower than one per cent on long-term trajectories. More recently, an interesting real-time 3D reconstruction of a trajectory from a stereo video was enabled using an efficient dense stereo matching combined with multi-view images from visual odometry algorithm in [6].

In the works cited above, real-time performance is achieved running on a CPU processor based on desktop, laptop or directly integrated to the robot [1, 14–16]. For fully embedded solutions, a visual odometry implementation for small robots uses an OMAP3530 board, which is composed of a DSP (C64) and an ARM (Cortex A8) [17]. This work takes advantage of the two board components in splitting different tasks between those two components. Dense stereo is done by the DSP, while feature detection, matching and ego-motion are computed by the ARM. Thus, stereo vision and visual odometry are parallelised which enables faster execution of the whole process. The motion estimation algorithm used for this contribution is the one developed in [5]. The whole visual odometry algorithm processes 512 × 384 (0.2 MPixel) at 6 Hz.

A modern implementation [18] presented an independent stereo vision and IMU perception unit equipped with the same OMAP3530 board. It is also equipped with an FPGA board and an Intel Core2Duo 1.86 GHz CPU board. In the same philosophy as [17], each board has been attributed a task. The ARM collects and integrates IMU data, whilst the FPGA board computes the disparity image using Semi-Global Matching. CPU tasks consist of stereo image acquisition, feature detection and matching, and then ego-motion. The visual odometry algorithm used in their work follows the same methodology as in [14]. However, it improves the process by fusing IMU data with visual odometry through an extended Kalman filter (EKF) in order to compensate for the delay of the vision pipeline and to strengthen the state estimation. Processing 1024 × 508 (0.5 MPixel) stereo images, the total visual odometry runs at 5 Hz.

In our work, we present an innovative smart and robust navigation system solution equipped with high-resolution stereo cameras (1.2 MPixel) and also an inertial measurement unit (Fig. 1). The solution is controlled with a single-board computer (SBC) with a 1.9 GHz Dual Core CPU and a NVIDIA chipset-integrated GPU. Visual odometry stages are split between CPU and GPU devices. In contrast to [17, 18], we do not use dense stereo to generate disparity maps as it remains an expensive operation in terms of computation even when running in a dedicated device. In fact, we preferred a sparse approach enabling us to track up to 750 initial features on high-resolution images (1.2 MPixel) in real time. Feature detection and features tracking are well suited to parallelised operations according to the sparse and independent nature of features. Additionally, we present a novel IMU-assisted feature tracking method, based on the KLT (Kanade–Lucas–Tomasi) feature tracker [19], where inertial information is used in combination with 3D geometry and stereoscopic properties in order to predict feature location in subsequent stereo

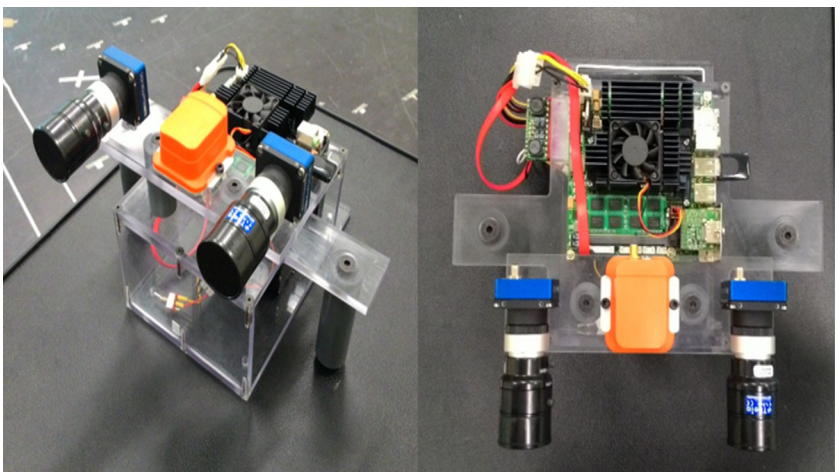

Fig. 1 Standalone stereo ego-motion navigation system

image pairs. This increases feature tracking efficiency while decreasing processing time.

In general, the majority of KLT-based variants modify the warping function using an affine model to adapt the template to the different conditions that might occur between two successive images, such as change in illumination, rotation, and scale [20–22]. Recent contributions have introduced the use of orientation information to assist the KLT feature tracker especially gyroscope data for image stabilisation [12] or GPS/INS [23]. The work of Hwangbo [11], which is one of the pre-eminent work in this sense, presents a robust gyro-aided version of the pyramidal KLT method [19]. It uses instantaneous gyro angles to get inter-image orientation information to help in the computation of the homography matrix between two consecutive images.

The obtained homography matrix is used to update the parameters of an affine photometric model for the warping function. The affine photometric model has 8 parameters allowing robust tracking despite camera rotation and outdoor illumination. However, this model leads to a significant computational cost.

In these two contributions [11, 12], the benefit of gyroscope information is significant allowing the KLT to cope with sharp rotation where it usually fails. However, this remains possible only at the condition of a quasi-pure or a pure camera rotation. Hence, it is assumed a negligible inter-frame translation regarding the scene depth (i.e. very small-scale change). This condition can be fulfilled with a high frame acquisition rate. To do so, the approach used by Hwangbo [11] requires a parallel processing implementation.

In our case, for computation complexity reasons, a translational model is preferred to the affine one for the KLT warping function. Thus, our innovative and computationally efficient IMU-assisted KLT tracker not only uses the gyroscope but also accelerometer data, to get the full IMU information. Consequently, it is robust against rotation changes similarly as [11, 12], but especially, it extends the use of the KLT by handling important scaling between consecutive images. This allows the KLT to be partially released of its spatial constraint, allowing low frame rate processing, which is not the case for gyro-only solutions. To enable a continuous and efficient use of accelerometer measurements, the IMU information has to be updated over time. This is why our IMU-assisted KLT tracker technique is an integral part of a visual odometry algorithm, which is the second contribution of this work. Indeed, at each new image, the inter-frame pose resulting from our visual odometry initialises the IMU.

In this work, instead of using the Levenberg–Marquardt algorithm [24], we decided to adapt the double-dogleg trust region method [25] which is a variant of the dogleg algorithm [26], to solve the bundle adjustment for motion estimation. Like the Levenberg–Marquardt algorithm, this technique combines steepest descent and Gauss–Newton direction. The main difference lies in direct control between the two directions by the means of a trust region which is likely to increase the convergence speed. In [27], it was shown that the use of the dogleg trust region technique presents advantages in terms of computational cost compared to Levenberg–Marquardt methods for full bundle adjustment applied to 3D structure reconstruction only. Regarding the visual odometry algorithm, we demonstrate similarly to [5] and [17] that a two frames approach is enough to achieve accurate egomotion. Finally, the overall solution is independent of any external source (e.g. GPS).

3 Navigation system software development

The software part of our navigation system is the implementation of our visual odometry approach including pipeline structure and algorithms. In order to achieve real-time performance, the efforts were primarily focused on refining the classical structure of the stereo visual odometry pipeline. Additionally, in order to maximise the efficiency of the stereo visual odometry pipeline, the different operations are shared between the CPU and the GPU memories (Fig. 2). Also, the CPU memory is configured in a multi-threading scheme with two threads. The main thread is in charge of image acquisition from the stereo camera and manages the stereo visual odometry algorithm in parallel with the GPU memory. The second thread handles IMU data acquisition.

3.1 Algorithm description

Although there are many approaches to implement stereo visual odometry, the majority of them follow a feature-based pipeline composed of distinct but interdependent stages, summarised here as a reminder:

Image acquisition Previous and current stereo pairs consisting of 4 images are acquired. It generally includes a rectification process which facilitates the stereo matching stage.
Feature detection Detects the remarkable keypoints on the 4 images.
Keypoints description Calculates for each image the descriptor of each keypoint which contains the related surrounding information.
Stereo matching Matches unilaterally or bilaterally the keypoints between the previous stereo pair images and between the current stereo pair images.

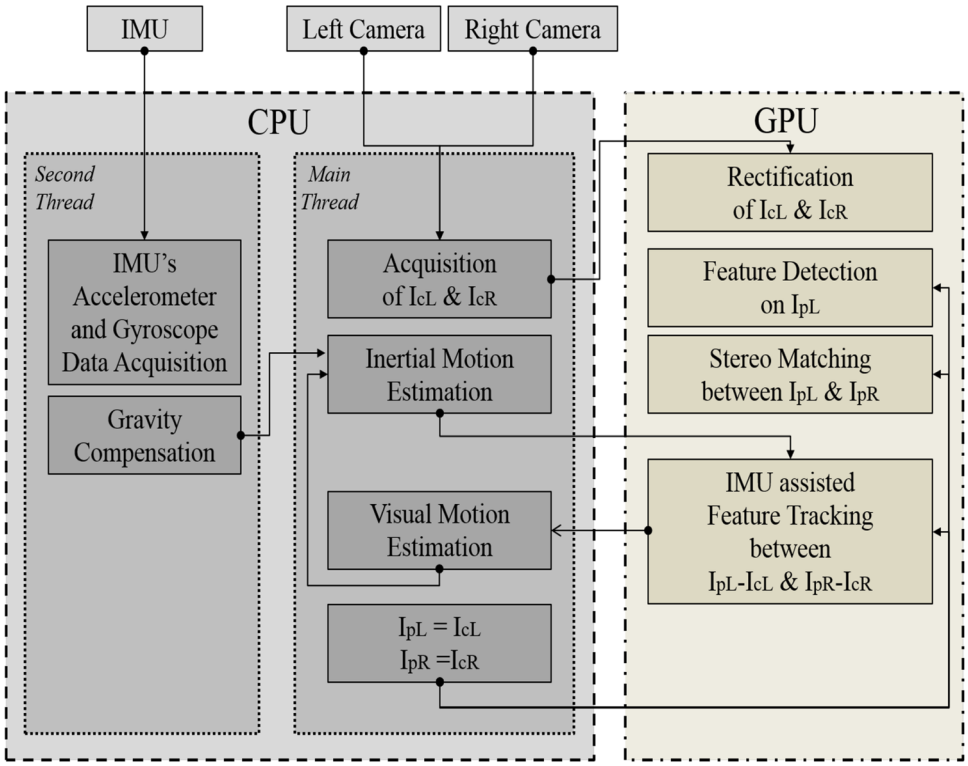

Fig. 2 Software implementation structure

Temporal matching Matches unilaterally or bilaterally keypoints between the previous and current left images and between the previous and current right images.
Motion estimation Calculates the 6-degree-of-freedom inter-frame pose using a least square-based motion estimation algorithm.

We made the decision to use a photometric-based matching approach in selecting the KLT technique, instead of a descriptor-based matching approach. The first reason is that even if using descriptors might be more robust, it is computationally expensive especially when it is based on SURF or SIFT. On the other hand, there are detector/descriptor combinations such as FAST/BRIEF for instance which have lower level of robustness but offer a better computational cost. Nevertheless, descriptor-based approaches are a structurally heavy process as it requires the computation for each keypoint on each image.

The KLT technique considers local information derived from small search windows surrounding each of the interest points. It assumes a certain invariance which constrains template image analysis in time, space, and brightness. These conditions are fairly well guarantied, based on the non-highly dynamic nature of general space mission scenarios. Thus, in our approach after the image acquisition and rectification of the most recent stereo pair I_{cL} and I_{cR}, we only need to run the feature detector technique only on the previous left image I_{pL} rather than on the 4 images of the two stereo pairs ($\{I_{pL}, I_{pR}\}$ and $\{I_{cL}, I_{cR}\}$). For this operation, we use a GPU implementation of the keypoint detector known as "good features to track" (GFTT) [28]. The set of detected features in I_{pL} forms the inter-frame reference keypoints for our stereo visual odometry.

Therefore, starting with this set of features as initial conditions, the KLT will search locally on I_{pR} using a GPU implementation of pyramidal KLT [29]. This allows the combination of feature detection and stereo matching operations at the same time. A small filtering function discards wrong matches which do not validate epipolar and spatial constraints. This results in a consistent set of previous stereo matched feature pairs $s_{\mathrm{p.}}$

In parallel, each time, new images are captured, the main thread gets from the second thread, inertial measurements that were accumulated during the inter-frame. Calibrated accelerometer and gyroscope measurements are acquired at a frequency of 100 Hz and are given a timestamp before the gravity compensation stage. Individual timestamps and known transfer time delays enable us to synchronise inertial and visual data. By integrating inertial data, we obtain an inertial motion estimation matrix composed of $R(q_{imu})$ and t_{imu}. Inertial data are then combined with the stereo pair set of feature s_{p} in a IMU-assisted KLT scheme (detailed in the next section).

As a result, we obtain the set of inertial estimated features s_c^*. These serve as initial conditions in the GPU

implementation of pyramidal KLT algorithm, to find the right candidates in the current stereo pair I_{cL} and I_{cR}. Epipolar and spatial constraints are checked, and the remaining candidates form a final set s_{pc} which consists of previous and inertial current 2D features. Once more, a number of operations are avoided as feature detection and temporal matching are associated. The final set of feature s_{pc} linking the 4 images is transferred to CPU memory as input to the motion estimation function. Visual motion estimation is computed by minimising re-projection errors between consecutive stereo pairs. Velocities are then calculated from the resulting visual motion estimation R_v and t_v, and serve as initialisation for the integration step in the next inertial motion estimation stage. R_v and t_v are accumulated in a global motion matrix Rt_{global} memorising the full trajectory done by the intelligent navigation sensor.

In the presented stereo visual odometry pipeline, we take advantage of the photometric-based matching approach characteristics in order to minimise at best the number of operations. Compared to a classical descriptor-based approach which takes around 15 operations, the proposed structure enables us to significantly reduce the number of operations in the visual odometry pipeline to only 8 stages including the inertial data-related stages.

3.2 IMU-assisted KLT feature tracking

The singularity of our technique resides in the use of stereoscopic properties in order to combine visual and inertial data. Contrary to similar works [11, 12], which are based on homography 2D transform image operation, we use 3D geometry combined with the knowledge of inertial inter-frame pose ($R(q_{imu})$ and t_{imu}) to predict the localisation in the current stereo frames of the previous initially detected and stereo matched features. Figure 3 summarises our idea and highlights five key steps:

Stereo matching The detected features are matched between right and left previous images giving a set s_p=$\{p_{pL(j)}, p_{pR(j)}\}$ of n correct stereo correspondences ($j = 1, \ldots, m$, *m the number of points*).

3D reconstruction Features from the set s_p are reconstructed in 3D using stereo calibration parameters by triangulation [25], resulting into a set S_p=$\{P_{(j)}\}$ of 3D points representing the position of the stereo correspondences in the space.

Inertial motion estimation matrix Acquired IMU information (accelerometer and gyroscope) is calibrated data to which we compensate the gravity. Then, the inertial inter-frame relative motion composed of $R(q_{imu})$ and t_{imu} is obtained after integration of the processed IMU data as follow:

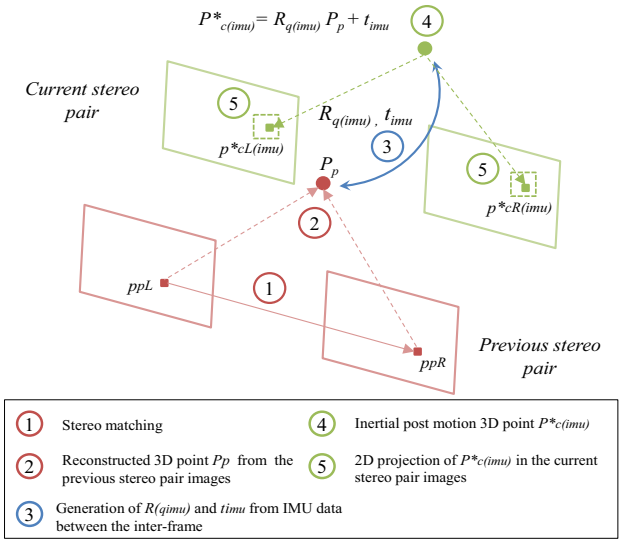

Fig. 3 IMU-assisted feature tracking process

$$\begin{cases} \dfrac{dq_{imu}}{dt} = \dfrac{1}{2} q_{imu} \otimes \omega \\ \dfrac{dv_{imu}}{dt} = R_{q_{imu}} a_{imu} + g \\ \dfrac{dt_{imu}}{dt} = v_{imu} + v_v \end{cases} \tag{1}$$

R_{qimu} represents the rotation matrix corresponding to q_{imu}, $\otimes$ is the quaternion product, g is the gravity vector, and v_v is initial speed resulting previous visual motion estimation.

Calculation of 3D inertial guesses This inertial motion matrix is then combined with S_p in order to obtain the 3D inertial guesses following equation of motion (2) described below:

$$P^*_{c(imu)}(t') = R(q_{imu}) P_p(t) + t_{imu} \tag{2}$$

The set of 3D post-inertial motion features is called $S^*_c = \{P^*_{c(imu)}\}$ with $P_p = [X_p, Y_p, Z_p]^{\mathrm{T}}$ and $P^*_{c(imu)} = [-X_{c(imu)}, Y_{c(imu)}, Z_{ci(imu)}]^{\mathrm{T}}$.

Projection into 2D image plane Components of S^*_c is projected into the current stereo pair images using the stereo camera parameters as described here:

$$\begin{cases} p^*_{cL(imu)} = \begin{bmatrix} u^*_{cL(imu)} \\ v^*_{cL(imu)} \end{bmatrix} = \begin{bmatrix} f \dfrac{X_{c(imu)}}{Z_{c(imu)}} + u_0 \\ f \dfrac{Y_{c(imu)}}{Z_{c(imu)}} + v_0 \end{bmatrix} \\ p^*_{cR(imu)} = \begin{bmatrix} u^*_{cR(imu)} \\ v^*_{cR(imu)} \end{bmatrix} = \begin{bmatrix} f \dfrac{X_{c(imu)} - B}{Z_{c(imu)}} + u_0 \\ f \dfrac{Y_{c(imu)}}{Z_{c(imu)}} + v_0 \end{bmatrix} \end{cases} \tag{3}$$

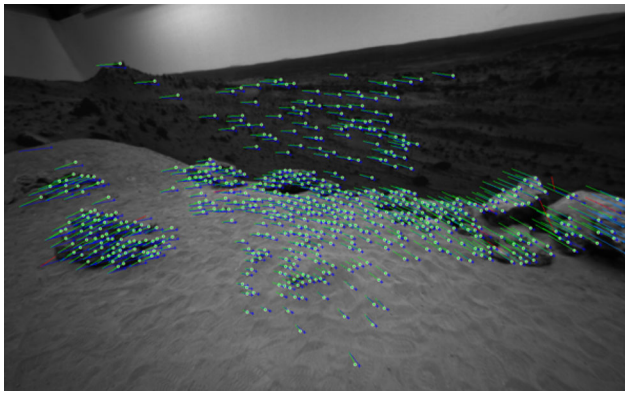

Fig. 4 Illustration of the IMU-assisted feature tracking. *Blue lines*—inertial optical flow, *red lines*—outliers, and *green lines* with *white dotted* end—inliers

where f is the focal length, u_0 and v_0 are the central pixel's coordinates, and B is the stereo baseline.

A set $s_c^* = \{p^*_{cL(imu)}, p^*_{cR(imu)}\}$ of 2D post-inertial motion features is then formed. Thus, inertial guesses are passed to the KLT a relatively fair location estimation of the features to be tracked which results in two main advantages. First, it reduces the probability of tracking wrong features. Second, accurate guess locations indicate a search area which results in a decrease in the dedicated search time to locate the right candidate. Figure 4 illustrates final result of this process.

3.3 Visual motion estimation

Given the group of feature correspondences between consecutive stereo image pairs s_{pc}, the camera motion is computed following the nonlinear objective function f, minimising the feature re-projection error in function of the motion parameter vector κ expressed as follows:

$$\min \sum_{i=1}^{N} \left\|p_{cL(i)} - f(P_{p(i)};\kappa)\right\|^2 + \left\|p_{cR(i)} - f(P_{p(i)} - B;\kappa)\right\|^2 \tag{4}$$

with

$$\kappa = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & t_x & t_y & t_z \end{bmatrix}^T \tag{5}$$

This motion parameter vector κ to be optimised is a 1×7 vector which consists of the four quaternion elements for the orientation and the translational elements on the three axes (x, y, z). The nonlinear re-projection function f takes as input P_p a 3D triangulated feature from the previous stereo pair and the motion parameter κ (also the baseline B for right pair features). The relation between spatial and planar representations is obtained with the help of the rectified camera matrix K_{rect} as described in (3). The objective is to reduce the pixel distance between the tracked features and their relative re-projected features.

The Levenberg–Marquardt algorithm is widely used to solve the bundle adjustment problem [30, 31]. In contrary to line search optimisation methods, trust region approaches set first a maximum distance before choosing a direction. Hence, the model is trusted around a restricted area Δ, which is adjusted along iterations. If the model matches the objective function f, then Δ is increased, whereas it decreases if the approximation is poor. In this work, instead of using the Levenberg–Marquardt algorithm, we decided to adopt the double-dogleg trust region method [25] which is a variant of the dogleg algorithm [26] to solve the bundle adjustment for motion estimation. The dogleg algorithm is delineated by two lines composed of the steepest descent direction and the Newton point direction. The optimal trajectory follows the steepest descent direction until reaching the Cauchy point (C.P) then converges to the Newton point passing by the dogleg step. This latter should be intersecting with the trust region boundary Δ. By introducing an intermediate Newton step N between the C.P and the actual Newton point, the behaviour of the double-dogleg algorithm presents a further improvement. Indeed, the optimal curve trajectory crosses the trust region before the original dogleg. This direct control between these two lines (steepest descent and Newton) by the mean of trust region (characterised by Δ) gives a faster optimisation to the algorithm and is also the main difference with the Levenberg–Marquardt algorithm.

This optimisation algorithm is implemented in a RANSAC scheme and aims to determine a set of inliers. At each iteration, three feature correspondences are randomly chosen. We apply rotation and translation transformation obtained from motion parameters to the set of 3D features in previous stereo image pairs using the equation of motion. The resulting 3D positions are then projected on the current stereo image pair.

Then, we iteratively minimise the sum of errors of features re-projection using our introduced double-dogleg trust region method. If it converges, we obtain the camera motion estimation and update the motion parameters. Then, an inliers selection process is carried out using last the motion parameters. If the norm of the error projection sum of a feature correspondence lies under a certain pixel threshold, it is considered as an inlier. Motion parameters giving the highest rate of inliers are kept and then used in a motion refinement stage using only the inliers. From the local camera motion obtained, we derive relative translations to get local velocities that serve in initialising subsequent inertial motion estimation.

3.4 System framework

The developed and presented stereo visual odometry algorithm is aimed to be embedded into the GPS-denied visual navigation solution. One of the main issues to be solved relates to the software and hardware communication. If hardware manufacturers provide linkers such as drivers, SDK, and API to facilitate interconnection, then each sensor has its own protocols and classes. Generally, these linkers enable a deep control of the sensors helping users to exploit all their functionalities. In addition to sensors communication links, the programming language of the main program and the different libraries to be used are essentials. One of the objectives when designing a software framework is to standardise intercommunication between all the involved components. Having said that there are robotic frameworks which aim to unify all robotic components and to implement links between them, R.O.S (Robot Operating System) is the most notable among others such as Player, Urbi, or Orca for instance. These robotic frameworks are also called robotics middleware as they aim to offer an intermediate platform to connect hardware and software parts of a complex robotic system. The range of robots and hardware (cameras, lasers, audio, etc.) managed by these middleware is continuously increasing. Despite this remarkable standardisation effort for the majority of sensors, only basic functionalities are reachable through such middleware. For our application, we wanted to have a full access to the cameras and IMU functionalities in order to have better control and optimise at best the utilisation of data streams. This is why we have implemented our own linkers using the camera's API and the IMU's SDK.

Our main program is coded in C++. We use computer vision functions from the OpenCV C++ library [29]. POSIX Thread is used for thread management [32] and the C++ Boost POSIX time library for timestamp generation and time-related operations [33]. The solution was also developed to be portable on Linux or Windows operating systems. Our program was tested offline on Windows 7. However, it is an Ubuntu 12.04 LTS version that is installed in our navigation system to run our stereo visual odometry algorithm online.

4 Navigation system hardware

In this section, we present the hardware components that were selected to develop our standalone navigation system and explain the reasons that motivated these choices. As it has been mentioned in the introduction, several constraints were to be considered in order to design an independent and flexible visual navigation system. Using off-the-shelf hardware, we aim to present a smart solution, with a minimum footprint but also powerful enough to manage inertial and vision sensors, while handling the whole visual odometry pipeline in real time.

4.1 Cameras selection

Starting with the camera selection seems logical to us since the nature of our task gives a central role to visual sensors. We opted for two MvBlueFOX-IGC USB 2.0 cameras embedding a 1280 × 690 pixels resolution CMOS Aptina MT9M034 image sensor; this presented an advantageous compact design. Finally, two Theia MY110 M 110° × 94° field of view ultra-wide lenses complete the visual sensor package. Ultra-wide lenses provide very low distortion, which is facilitated in the camera calibration and the stereo rectification processes.

In a visual odometry context, having a wider field of view is really advantageous. It increased the feature keypoints persistence as well as the probability to catch remarkable points, within additional content of the scene, enabled by a wider field of view. The larger shared field of view, between the two cameras, also increases the probability of finding potential matches for stereo correspondences. The provided C++ API enables tight control of the different camera functionalities such as capture, frame rate, exposure, gain, and time stamping.

4.2 IMU selection

The inertial measurement unit is used in our solution, to assist the feature tracking operations. The selected inertial device for this task is an Xsens Mti-G. It is an integrated GPS and IMU with Navigation, Attitude and Heading Reference System (AHRS) processor. It is based on MEMS inertial sensors including also 3D magnetometer and a static pressure sensor as well as a miniature GPS receiver. In terms of dimension, its compact size (60 × 50 mm) suits well with our navigation system design requirements. Xsens Mti-G provides an USB hardware connection via an RS232 to USB converter. Calibrated accelerometer and gyroscope data (no GPS is enabled) are accessed through the provided sensor SDK which is coded in C. It gives us a relative flexibility for handling time stamping and data transfer.

4.3 Board selection

The board is an important part of the visual navigation sensor as it centralises all the input/output. Thus, for several reasons, our preference was given to ITX board types belonging to the single-board computer category (SBC). ITX boards offer a large flexibility regarding size,

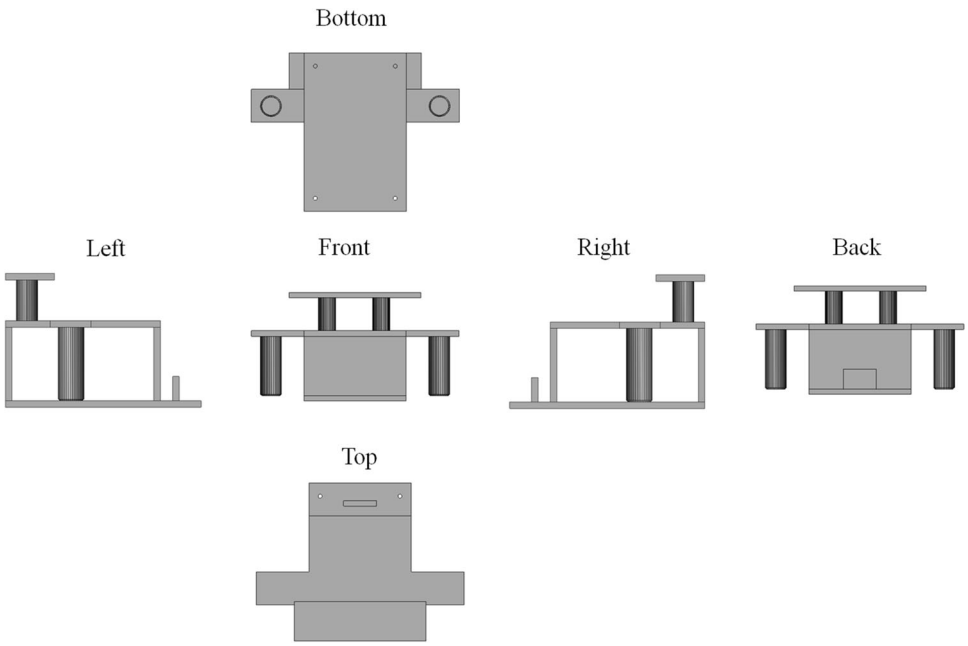

Fig. 5 Detailed views of the navigation system structure designed with SolidWorks software

Fig. 6 *Top left* two views of the final cameras position; *bottom left* final Vicon architecture; and *right* ESA's laboratory pool reproducing a Mars ground-like environment

processors, and peripheral connectors. For example, the ITX SBC range size varies from 45 × 75 mm (mobile) to 170 × 170 mm (mini). The processor type mainly depends on budget. Indeed, the latest ITX can support 4th generation of Intel i5 or i7 processors. Intel Atom or Celeron along with other types of processors such as AMD, ARM, Cortex, and Freescale IMX are also available.

In our case, we choose a nano ITX SBC (SECOnITX-ION) of 170 × 170 mm size from the brand SECO. This SBC has a 1.9 GHz CPU processor Intel Celeron Dual Core T3100 and 16-core GPU-integrated controller NVIDIA® GeForce 9400 M. The selected SBC provides a powerful CPU processor compared to other available commercial SBCs. Although the GPU processor is quite basic compared to the latest graphic cards, it fitted reasonably well to the purpose of our application. We added a 4 GB DDR3 memory which is the limit that can be handled by the dedicated SO-DIMM socket. Ubuntu distribution was installed in a 1 TB DELL PDA1000B portable external hard drive, and we used 4 GB RAM, which is the maximum memory that can be handled by the board.

The four USB connectors were holding, respectively, to the two stereo cameras, IMU sensors, and Wi-Fi dongle. The Wi-Fi dongle was used for SSH communication

between SBC and an external laptop to launch and stop the visual odometry program. The SBC is powered via a +12 V_{DC} AT/ATX connector. We used a DC-DC picopsu-80-WI-32 (pico power supply unit (picopsu), 80 W, wide input 12–32 V), which has the great advantage of being small, silent, fan-less and with a very small footprint. Two modes of operations were designed:

Development mode The picopsu is connected to the sector via an 80 W AC/DC adapter. It was used during development and testing phases.

Experiment mode The picopsu is connected to a Li-ION 14.8 V 5200 mAH battery pack from the brand Tenergy via a P4 connector linked a with power switch to supply the SBC. It is for datasets acquisition and real-time assessment of our sensor navigation system. This battery pack enables the visual navigation sensor to run for slightly longer than an hour.

4.4 Whole structure hardware

The structure carrying all the components has been designed in a cubic form for convenience with a rectangular stereo plate on its front top and two handles on its sides, allowing handheld navigation as illustrated in Fig. 5.

The two cameras are placed on each side of the rectangular stereo plate in a way that the stereo baseline is 16 cm. This chosen distance combined with the ultra-wide angle provided with the Theia lenses gives a good compromise between design compactness and stereo vision properties. The Xsens device reference frame was carefully aligned as accurately as possible in the middle of the baseline structure to facilitate reference frame transformations with the left stereo camera. The latter is taken as the main reference frame. The SECOnITX-ION SBC is mounted on the top of the structure in order for the SBC's fan to have good airflow, allowing the SBC to avoid heating thanks to a good air circulation. However, this is not the only reason. The SBC occupies a central role and needs to be placed in such way that it is accessible to all the other components. The portable external hard drive is placed at the back of the structure, while the battery pack is fixed inside the cubic structure (in between the two

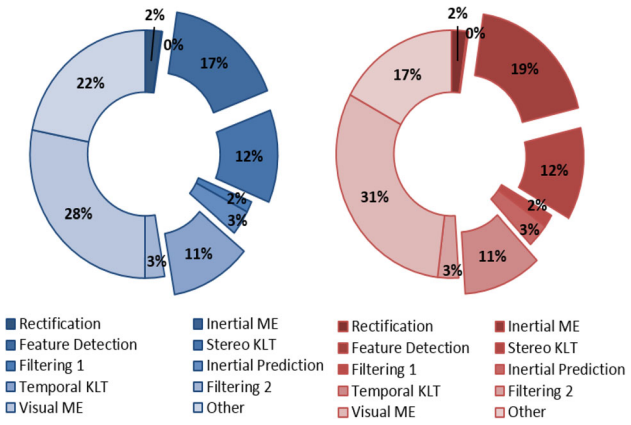

Fig. 7 Run A: runtime breakdown in percentage of visual odometry pipeline using double-dogleg algorithm (*left*) and Levenberg–Marquardt algorithm (*right*) for visual motion estimation stage

Table 2 Run A: average feature-related operations results through visual odometry pipeline on 247 frames

	Detected features	Correctly stereo tracked	Inliers
Using DDL	569	235	208
Using LM	569	235	210

Table 1 Run A: runtime of visual odometry pipeline at 1280 × 960 resolution and starting with 750 initial features

	Using double dogleg		Using Levenberg–Marquardt	
	in (ms)	in (%)	in (ms)	in (%)
Rectification	4.4	2	4.4	2
Inertial ME	0.3	0	0.4	0
Feature detection	33.5	17	38.8	19
Stereo KLT	24.1	12	24.6	12
Filtering 1	2.8	2	3.4	2
Inertial prediction	6.4	3	6.3	3
Temporal KLT	22.5	11	22.6	11
Filtering 2	5.6	3	6	3
Visual ME	55.9	28	63.7	31
Other	43.5	22	35.1	17
Total	199	100	205.3	100
Frame rate	5.02 fps		4.87 fps	

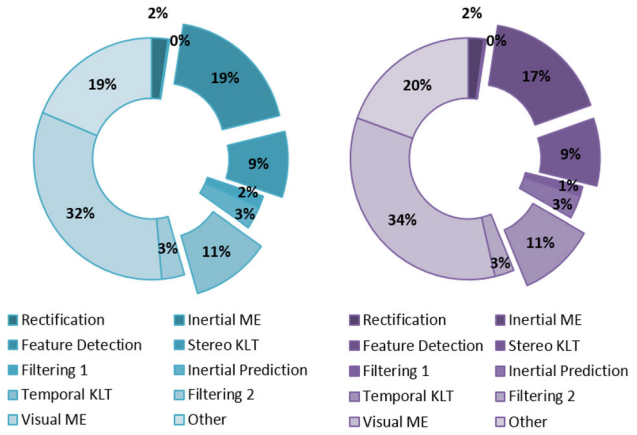

Fig. 8 Run B: runtime breakdown in percentage of visual odometry pipeline using the double-dogleg algorithm (*left*) and the Levenberg–Marquardt algorithm (*right*) for visual motion estimation stage

handles). In order to adapt different robotic platforms, four holes have been added to the bottom of the structure. Dimensions of the whole structure are 16 × 20 cm (approx.) excluding the handle "wings". The total weight is about 2.5 kg.

5 Experiments

The assessment of the designed visual navigation sensor performance is divided into two parts. The first part focuses on runtime analysis of the full visual odometry pipeline. The second part evaluates visual odometry accuracy in trajectory generation. Also, the double-dogleg algorithm is compared to the sparse bundle adjustment version of the Levenberg–Marquardt algorithm in [24] regarding the motion estimation performances.

Table 4 Run B: average feature-related operations results through visual odometry pipeline on 242 frames

	Detected features	Correctly stereo tracked	Inliers
Using DDL	457	221	193
Using LM	457	221	195

5.1 Dataset and experimental conditions

Experiments took place in ESA's laboratory equipped with a 9-m square pool reproducing a Mars ground-like environment (clay, rocks, etc., see Fig. 6). The visual navigation system was handheld, and runs were made walking around the pool following specific paths.

Working in an environment that attempted to reproduce some of the conditions in Mars led us to face some of those challenging aspects. For instance, the non-homogenous pool's ground creates instability and the limited feature environment or textureless zones were plenty. Hence, the navigation system was subject to recurrent and sometimes sudden variation in height due to clay bumps, holes, or slipping surfaces when walking on it.

5.2 Experiment set up and Vicon system

In order to validate our solution in terms of accuracy, we need to have a strong and reliable navigation reference. This is provided by the Vicon motion capture system (Bonita) that equips the ESA laboratory and which consists of 10 networked infrared (IR) cameras, 7 of them are 1.3 megapixel resolution (MX13+) and the three remaining are 2 megapixel resolution (MX20+). They also provide a high frame rate capture capability up to 100 fps. The IR cameras track 50-mm spherical retroflective markers that appear isolated from the scene background because of their high reflectivity. The

Table 3 Run B: runtime of the visual odometry pipeline at 1280 × 960 resolution and starting with 750 initial features

	Using double dogleg		Using Levenberg–Marquardt	
	in (ms)	in (%)	in (ms)	in (%)
Rectification	4.4	2	4.4	2
Inertial ME	0.4	0	0.6	0
Feature detection	36.4	19	35.3	17
Stereo KLT	17	9	18.6	9
Filtering 1	3.2	2	2.7	1
Inertial prediction	5.7	3	6.3	3
Temporal KLT	21	11	21.6	11
Filtering 2	6.1	3	5.5	3
Visual ME	63.2	32	69.9	34
Other	36.4	19	40	20
Total	193.8	100	204.9	100
Frame rate	5.16 fps		4.88 fps	

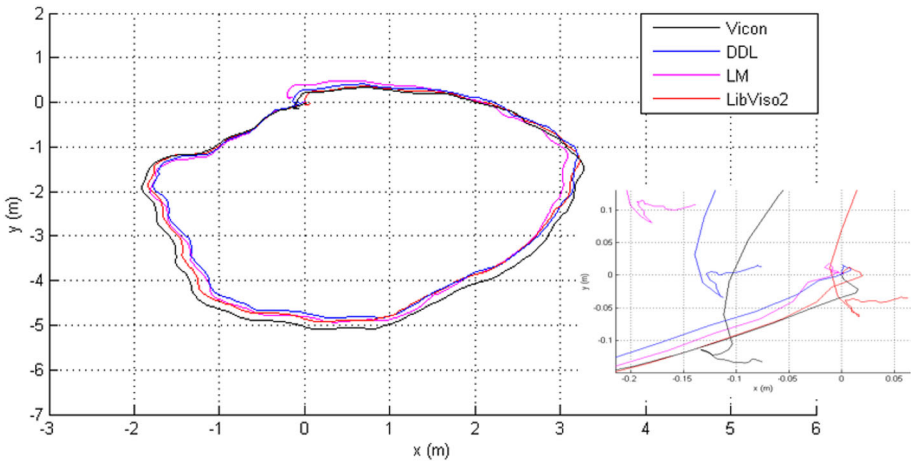

Fig. 9 Run A: 2D plot and zoom on final position of the trajectory generated with the navigation sensor using the double-dogleg algorithm (*blue*) and the Levenberg–Marquardt algorithm (*magenta*), compared to the Vicon reference (*black*) and the LibViso2 library (*red*)

markers' 3D positions can be precisely recovered by triangulation through the Vicon Nexus software. The host desktop running the Vicon Nexus software is linked to the Ultranet, and the Giganet networking devices. The final architecture of the Vicon set up is illustrated in Fig. 6. In order to represent the navigation sensor in the Vicon Nexus software, four markers were placed on the visual navigation system in such a way that only the stereo plate is represented.

5.3 Visual odometry runtime performances

In this section, we give a detailed runtime analysis of each step of the stereo visual odometry pipeline for two representative runs (A and B). The runtime breakdown for run A is given in Table 1 and illustrated in Fig. 7. Feature detection, stereo KLT, and temporal KLT operations which run on the GPU device, represent almost half of the stereo visual odometry total runtime. This highlights the importance of using parallel programming to achieve real-time results. Hence, the adopted features sparse approach suits parallelisation very well. Indeed, the time saved in the visual odometry runtime is proportional to the number of features processed.

In Table 1, we can observe that except for the visual motion estimation stage, runtimes of the visual odometry pipeline using the two different techniques are almost equivalent. Indeed, the double-dogleg algorithm is faster than the Levenberg–Marquardt. Table 2 shows the high rate of inliers from the correctly stereo tracked matches for both techniques which provide a strong basis for the motion estimation stage.

A similar conclusion can be drawn in run B (see Fig. 8). Runtime of the complete visual odometry pipeline is given in Table 3 for run B. Figure 8 shows that runtime operations are proportionally similar to run A.

As a result, the developed visual navigation sensor solution achieves real-time performance for both runs, at a satisfying 5 fps frame rate. This is realised, with 1.2 megapixel stereo images while processing up to 750 initial detected features. Comparable recent works such as [17] or [18] reported an equivalent frame rate with lower resolution images and less initial features. Indeed, in [17], their algorithm which uses a more robust dense stereo algorithm finishes with 140 inliers for 400 initial features, while we obtain 200 inliers on average.

The rate of inliers from the correctly stereo tracked matches is still high, with on average 88 % for the dogleg and the Levenberg–Marquardt on both runs (Tables 2, 4). This is quite a remarkable result regarding the 5 fps acquisition constraint, resulting in a large inter-frame

Table 5 Run A: final position relative error comparison in trajectory generation

	2D relative error		3D relative error	
	in (m)	in (%)	in (m)	in (%)
DDL	**0.147**	**0.87**	**0.207**	**1.21**
LM	0.250	1.48	0.371	2.18
LibViso2	0.169	0.99	0.251	1.48

Bold value indicates the best results of the compared techniques

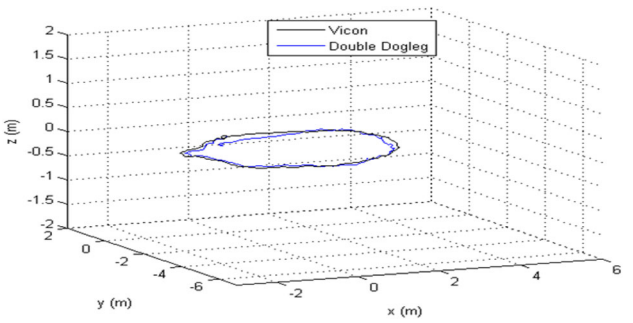

Fig. 10 Run A: 3D plot of the trajectory generated with the navigation sensor using the double-dogleg algorithm (*blue*) compared to the Vicon reference (*black*)

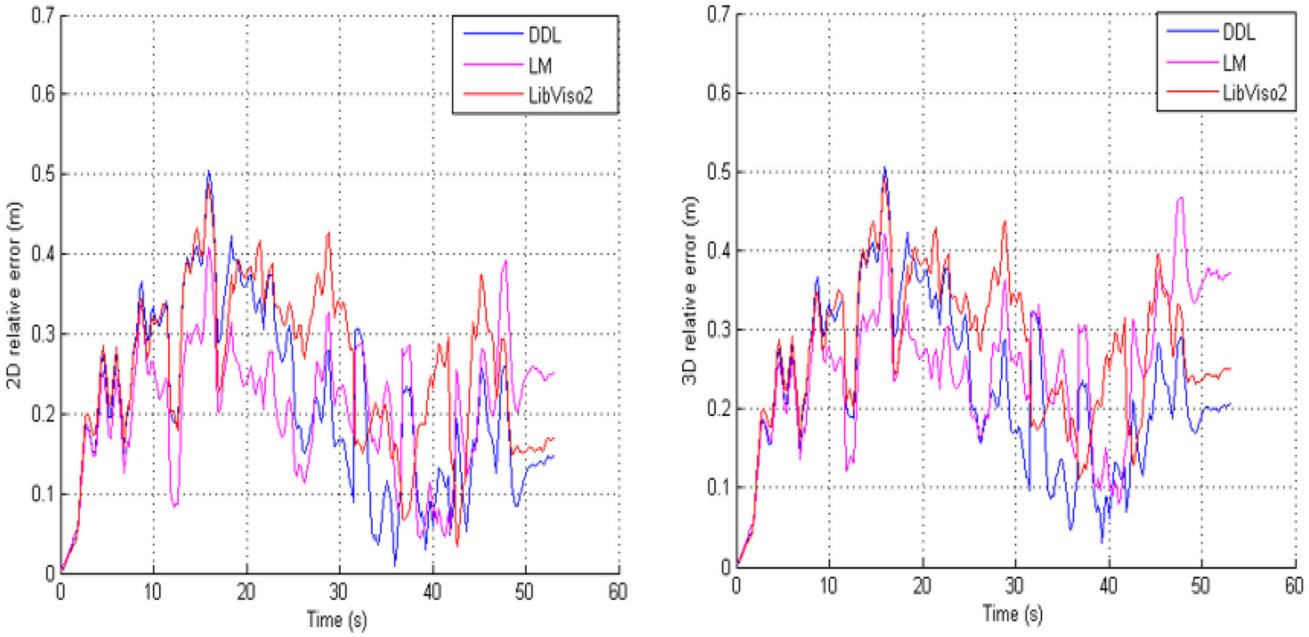

Fig. 11 Run A: relative RMSE 2D (*left*) and 3D (*right*) in metre over the time regarding the Vicon reference trajectory

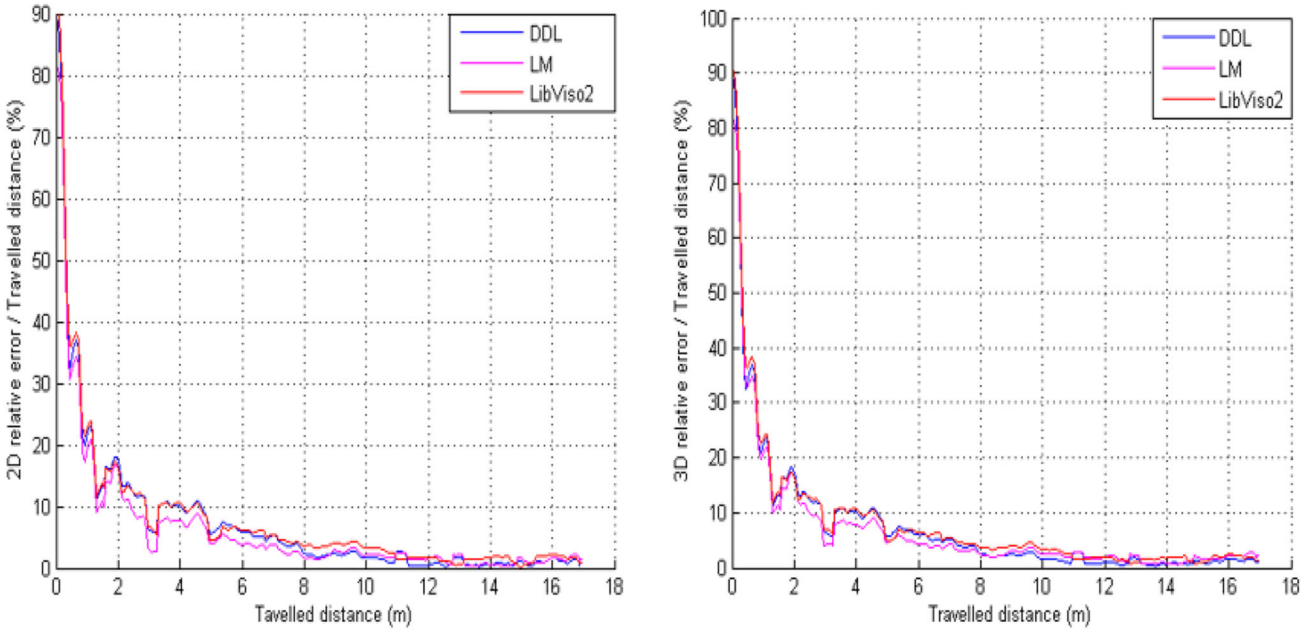

Fig. 12 Run A: relative RMSE 2D (*left*) and 3D (*right*) in per cent over the travelled distance regarding the Vicon reference trajectory

optical flow and especially the space-like environment where remarkable features are not plenty.

5.4 Visual odometry trajectory generation performances

For this assessment, we walked around the space pool for the two runs (A and B) closing the loop. Figure 9, shows the results of our stereo visual navigation system running with the double-dogleg algorithm (in blue) and with the Levenberg–Marquardt (in magenta), as well as with the LibViso2 library, providing a stereo visual odometry approach based on the Gauss–Newton algorithm [6] (code is available at [34]) (in red) compared to the reference motion capture trajectory (in black).

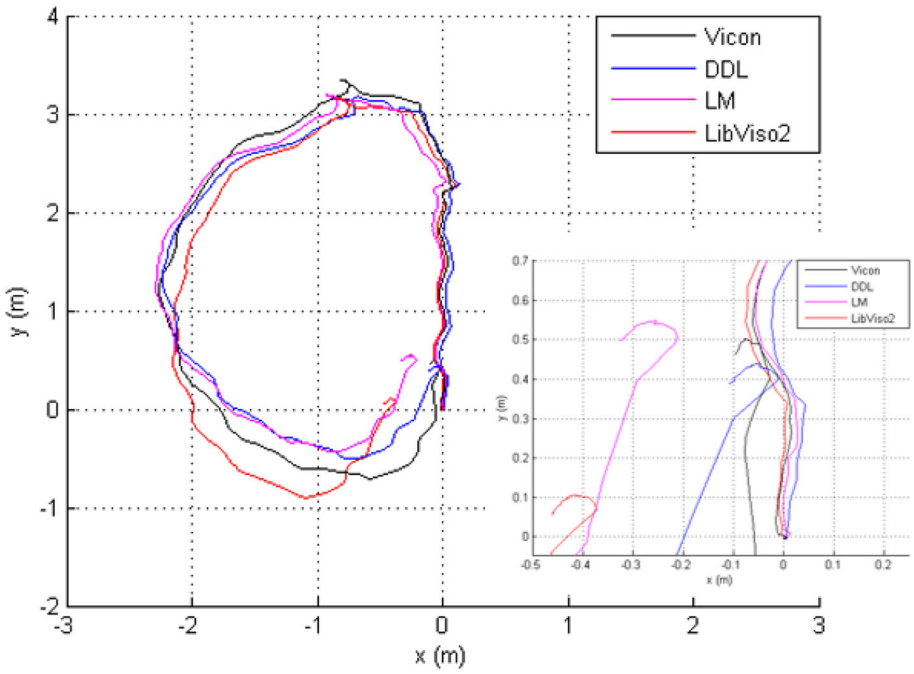

Fig. 13 Run B: 2D plot of the trajectory generated with the navigation sensor using the double-dogleg algorithm (*blue*) and the Levenberg–Marquardt algorithm (*magenta*), compared to the Vicon reference (*black*) and the LibViso2 library (*red*)

All the compared trajectories give almost the same shape as the Vicon reference trajectory. However, our trajectory remains the closest to the ground truth for the major part of the route and lies at the final position only at 14.7 cm from the Vicon trajectory closing point. The total travelled distance for Runa A is 16.95 m (Table 5).

In Fig. 9, a zoom of the starting and final position in 2D is also shown. In Fig. 10, the 3D trajectories of our solution and the Vicon reference are plotted. We can see that height is well estimated which shows the quality of the provided 6-degrees-of-freedom (DoF) solution.

Figure 11 shows that the 2D relative root-mean-square error (RMSE) remains constant over the time with of course some fluctuation similarly, to the relative 3D error in Fig. 12, where the trend gives a monotonic decrease of the error along the travelled distance. As a result, at the end of the route, the proposed solution achieves a remarkable error below 1 % of the travelled distance.

Figure 13 shows the results of our visual odometry algorithm running in our visual navigation system in run B. In this run, all the compared trajectories also give the same shape as the Vicon reference trajectory until a certain point. After this point, only the trajectory generated with our navigation system using double dogleg remains close enough from the ground truth, lying at the final position only 7.6 cm from the Vicon trajectory closing point, for a total travelled distance of 11.17 m (Table 6). Our visual odometry-based navigation system using the Levenberg–Marquardt is not closing the loop but remains not too far from the 2D final position (23.2 cm, see Table 6). On the other hand, the algorithm from LibViso2 library finishes the farthest from the Vicon reference final point. This case (run B) allows us to enlighten the utility of IMU information, which enables our navigation sensor algorithm to cope with this kind of uneven problem, whilst minimising the effect on the final trajectory. This problem was caused

Table 6 Run B: final position relative error comparison in trajectory generation

	2D relative error		3D relative error	
	in (m)	in (%)	in (m)	in (%)
DDL	**0.076**	**0.68**	**0.413**	**3.7**
LM	0.232	2.07	0.518	4.64
LibViso2	0.548	4.91	0.626	5.61

Bold value indicates the best results of the compared techniques

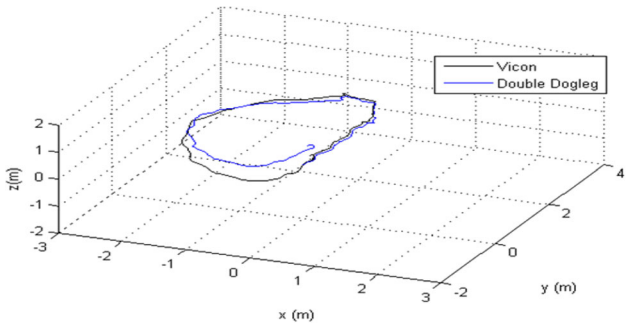

Fig. 14 Run B: 3D plot of the trajectory generated with the navigation sensor using the double-dogleg algorithm (*blue*) compared to the Vicon reference (*black*)

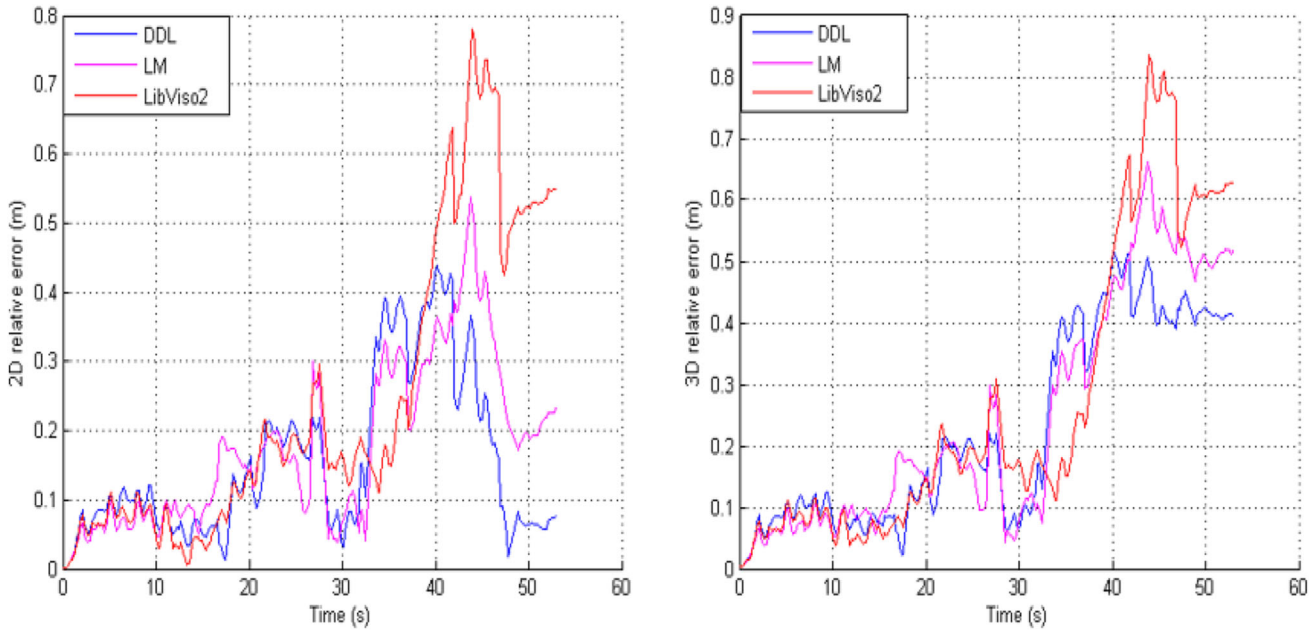

Fig. 15 Run B: relative RMSE 2D (*left*) and 3D (*right*) in metre over the time regarding the Vicon reference trajectory

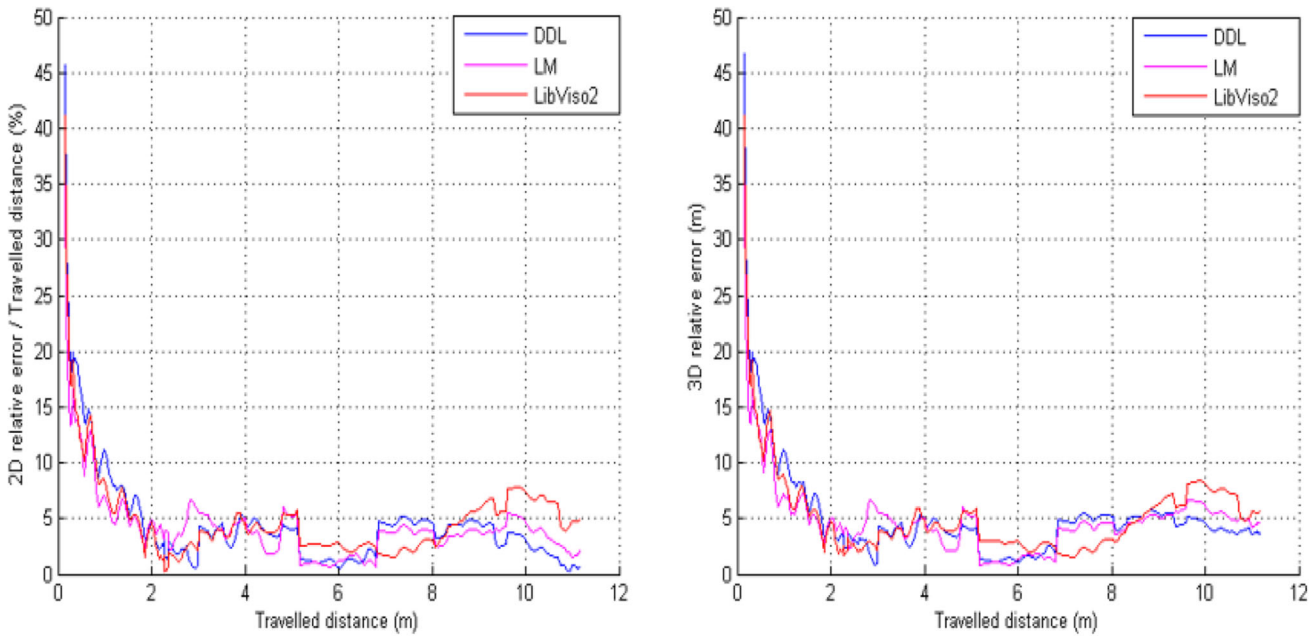

Fig. 16 Run B: relative RMSE 2D (*left*) and 3D (*right*) in per cent over the travelled distance regarding the Vicon reference trajectory

by an image acquisition lag which gave a 1-s delay between two frames.

In Fig. 14, the 3D trajectories of our solution and the Vicon reference are plotted. Here as well, that height is well estimated, expect towards the end where the delay caused by the image acquisition lag had a certain consequence, which slightly shifted the generated trajectory up the Z-axis.

Similarly to run A, Figs. 15 and 16 show that the relative RMSE remains bounded over the time despite the small hump before the end which characterises the encountered lag problem.

The results showed in this section demonstrate that our visual navigation sensor is able to generate accurate visual odometry trajectories in space-like environments, while achieving real-time performances. Our stereo visual

navigation sensor solution performs better and slightly faster with the double-dogleg method than with the Levenberg–Marquardt. It also provides more accurate 6-DoF trajectories than the ones generated using the Lib-Viso2 library.

6 Conclusion and future work

The visual/IMU navigation system presented in this paper is a real-time smart and standalone stereo/IMU ego-motion localisation sensor. We demonstrated through the different sections of this work the great potential of the strategy described and the choices of components and techniques used in our visualisation pipeline.

We developed an efficient strategy for our visual odometry algorithm that consists of optimising the usual pipeline, but also running feature detection stereo tracking and temporal tracking into the GPU device present in the nano ITX single-board computer. The use of IMU data to predict features for the next acquired stereo images improves the quality of the selected features. This also has a significant influence on the accuracy of the generated trajectories. We also showed that the use of the double-dogleg algorithm is well suited to a visual motion estimation application and has a faster implementation than the Levenberg–Marquardt algorithm.

As a result of a balanced combination of hardware and software implementations, the proposed solution achieved a 5 fps frame rate processing with up to 750 initial features at a resolution of 1280 $\times$ 960. This is the highest reached resolution in real time for visual odometry applications to our knowledge. Additionally, the visual odometry accuracy of our algorithm achieves less than 1 % relative error in the estimated trajectories.

The work described has great potential. Possible enhancements in the software and physical setup may yield further performance enhancement. The physical design was not optimised in terms of space utilisation. This may prove to be the easiest improvement to make in future iterations of this design.

Acknowledgments The research leading to these results has received funding from Thales and ESA (European Space Agency). We would like to thank them for their help and support.

References

1. Nister, D., Naroditsky, O., Bergen, J.: Visual odometry. IEEE Conf. Comput. V. Pattern Recognit. **1**, 652–659 (2004)
2. Scaramuzza, D., Fraundorfer, F.: Visual odometry [Tutorial]. IEEE Robot. Autom. Mag. **18**(4), 80–92 (2011)
3. Fraundorfer, F., Scaramuzza, D.: Visual odometry: Part II: matching, robustness, optimization, and applications. IEEE Robot. Autom. Mag. **19**(2), 78–90 (2012)
4. Konolige, K., Agrawal, M., Solà, J.: Large-scale visual odometry for rough terrain. In: Robotics Research, The 13th International Symposium ISRR, pp. 201–212. Springer (2011)
5. Howard, A.: Real-time stereo visual odometry for autonomous ground vehicles. In: IEEE International Conference on Intelligent Robots and Systems, pp. 3946–3952 (2008)
6. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3d reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium, pp. 963–968 (2011)
7. Chen, S.Y.: Kalman filter for robot vision: a survey. IEEE Trans. Industr. Electron. **59**(11), 4409–4420 (2012)
8. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: IEEE International Conference on Robotics and Automation, pp. 3565–3572 (2007)
9. Tian, Y., Hamel, W.R., Tan, J.: Accurate human navigation using wearable monocular visual and inertial sensors. IEEE Trans. Instrum. Meas. **63**(1), 203–213 (2014)
10. Voigt, R., Nikolic, J., Hürzeler, C., Weiss, S., Kneip, L., Siegwart, R.: Robust embedded egomotion estimation. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2694–2699 (2011)
11. Hwangbo, M., Kim, J.-S., Kanade, T.: Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation. Int. J. Robot. Res. **30**(14), 1755–1774 (2011)
12. Ryu, Y.-G., Roh, H.-C., Chung, M.-Y.: Video stabilization for robot eye using IMU-aided feature tracker. In: IEEE International Conference on Control Automation and Systems, pp. 1875–1878 (2010)
13. Sünderhauf, N., Protzel, P.: Stereo odometry–a review of approaches. Chemnitz University of Technology, Technical Report (2007)
14. Hirschmuller, H., Innocent, P.R., Garibaldi, J.M.: Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In: IEEE 7th International Conference on Control, Automation, Robotics and Vision, vol. 2, pp. 1099–1104 (2002)
15. Maimone, M., Cheng, Y., Matthies, L.: Two years of visual odometry on the mars exploration rovers. J. Field Robot. **24**(3), 169–186 (2007)
16. Agrawal, M., Konolige, K.: Real-time localization in outdoor environments using stereo vision and inexpensive GPS. In: IEEE International Conference on Pattern Recognition, pp. 1063–1068 (2006)
17. Goldberg, S.B., Matthies, L.: Stereo and imu assisted visual odometry on an omap3530 for small robots. In: IEEE Computer Vision and Pattern Recognition Workshops, pp. 169–176 (2011)
18. Schmid, K., Hirschmuller, H.: Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device. In IEEE International Conference on Robotics and Automation, pp. 4671–4678 (2013)
19. Bouguet, J.-Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. OpenCV Documentation, Intel Corporation, Microprocessor Research Labs (1999)

20. Zinßer, T., Gräßl, C., Niemann, H.: Efficient feature tracking for long video sequences. In: Pattern Recognition, Proceedings of the 26th DAGM Symposium, pp. 326–333. Springer (2004)
21. Sinha, S.N., Frahm, J.-M., Pollefeys, M., Genc, Y.: Feature tracking and matching in video using programmable graphics hardware. Mach. Vis. Appl. **22**(1), 207–217 (2007)
22. Lee, H.-K., Choi, K.-W., Kong, D., Won, J.: Improved Kanade–Lucas–Tomasi tracker for images with scale changes. In: IEEE International Conference on Consumer Electronics, pp. 33–34 (2013)
23. Tanathong, S., Lee, I.: Translation-based KLT tracker under severe camera rotation using GPS/INS data. IEEE Geosci. Remote Sens. Lett. **11**(1), 64–68 (2013)
24. Lourakis, M.I., Argyros, A.A.: SBA: a software package for generic sparse bundle adjustment. ACM Trans. Math. Softw. (TOMS) **36**(1), 2 (2009)
25. Dennis Jr., J.E., Mei, H.H.W.: Two new unconstrained optimization algorithms which use function and gradient values. J. Optim. Theory Appl. **28**(4), 453–482 (1979)
26. Powell, M.J.: A new algorithm for unconstrained optimization. Nonlinear Program. 31–65 (1970)
27. Lourakis, M.L.A., Argyros, A.A.: Is Levenberg–Marquardt the most efficient optimization algorithm for implementing bundle adjustment? IEEE Int. Conf. Computer Vision **2**, 1526–1531 (2005)
28. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600 (1994)
29. http://opencv.org/
30. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment — a modern synthesis. In: Vision Algorithms: Theory and Practice, Proceedings of the International Workshop on Vision Algorithms, pp. 298–372. Springer (2000)
31. Konolige, K., Agrawal, M.: FrameSLAM: from bundle adjustment to real-time visual mapping. IEEE Trans. Rob. **24**(5), 1066–1077 (2008)
32. https://computing.llnl.gov/tutorials/pthread
33. http://www.boost.org/
34. http://www.cvlibs.net/software/libviso/

Lounis Chermak received his MSc degree in image processing in 2011 and the Ph.D degree in computer vision in 2014 from Cranfield University, UK. He is currently Research Fellow at the Centre of Electronic Warfare in Cranfield University, UK. His research interests include, computer vision, robotics, navigation systems, and embedded sensor fusion.

Nabil Aouf is a Reader at the Centre of Electronic Warfare, Cranfield University, UK. His research interests are aerospace and defence systems, information fusion and vision systems, guidance and navigation, and tracking and control and autonomy of systems. He is an Associate Editor of the International Journal of Computational Intelligence in Control. He has more than 100 publications of high calibre in his domains of interest.

Mark Richardson has over 30-year experience of electro-optics and infrared systems and countermeasures in the defence industry and UK academia and has written well over 200 classified and unclassified papers on these subjects. He is currently head of the Centre for Electronic Warfare and Director of Research at the Defence Academy of the UK.

Gianfranco Visentin received is M. Eng., Turin Polytechnic, 1987. He worked from 1988 to 1990 on advanced control systems for automotive applications at Fiat Research Centre (Turin). He has been employed since 1991 at the European Space Technology Research Centre working on the development of technology for space robotics. Among the recent activities, whose focus is on the applications of robotics to in-orbit servicing, he has participated to the Geostationary Servicing Vehicle study.