Implementation of Pure and Hybrid Visual Odometries on electric self-driving all-terrain vehicles

Alessandro Scarciglia

School of Electrical Engineering

Thesis submitted for examination for the degree of
Master of Science in Technology.
Lecce 29.05.2020

Supervisor

Professor Arto Visala,
Professor Marcello Chiaberge

Advisor

M.Sc. Tabish Badar

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Alessandro Scarciglia

Title Implementation of Pure and Hybrid Visual Odometries on electric self-driving all-terrain vehicles

Degree programme Electronics and electrical engineering

Major Autonomous Systems **Code of major** ELEC0007

Supervisor Professor Arto Visala,
Professor Marcello Chiaberge

Advisor M.Sc. Tabish Badar

Date 29.05.2020 **Number of pages** 80 **Language** English

Abstract

The design of a fully autonomous mobile platform is tightly related to the development of a reliable navigation system. In such context, the problem of the *self-localisation* covers one of the most important challenges. Nowadays both autonomous and non-autonomous vehicles mainly rely on GPS when dealing with pose estimation. As a drawback, the GPS system is likely to return a non-informative estimate for short-range displacements (i.e. indoor mobile robots operations), as well as in some places the signal can be either unreliable or totally absent.

For the reasons mentioned above, a good *self-contained localisation* system needs to be integrated, in order to fulfill the localisation issue in a wider range of environments. The purpose of this research work is to develop a real-time application in order to get a *vision-based* pose estimate of a Polaris electric all-terrain vehicle equipped with an omnidirectional catadioptric camera. After a general overview of the state-of-the-art about *visual odometry* (VO), two approaches are compared: the former is a pure VO technique based on the hypothesis of ground planarity, whereas the latter is a hybrid approach which makes use of the speed information to offset the camera non-idealities.

In the end, performances are evaluated and the best solution undergoes to *code generation*. The goal to pursue is to obtain an execution time at least lower than the camera frame rate, in order to obtain a feasible solution for a real-time application.

Keywords visual odometry, omnidirectional camera, autonomous mobile robots, self-contained localisation, GPS-denied navigation

Contents

Preface

Resilience. It is the ability to spring back into shape and move on besides difficulties. I strongly feel to use this word to describe my experience abroad in a year which probably was not the best moment to stay far from home. In spite of all the unpredictabilities I coped with, I managed to conclude my master thesis with the exceptional support of a group of people that I would like to thank.

At first, I would like to thank my supervisor Prof. Arto Visala for the trust he showed in assigning me this interesting topic, as well as my advisor Tabish Badar who constantly followed my progresses even from far away. Likewise, I thank Mika Vainio and Andrei Sandru for the precious suggestions they gave me in the final review of my work.

A special thanks is addressed to my home supervisor Prof. Marcello Chiaberge from Politecnico di Torino, who always replied promptly to all my doubts and requests all over the academic year.

A remarkable thanks goes to my family who supported by any means my education in these long twenty-four years and who shortened the distance from Italy to Finland with their long calls and unconditional love.

Last but not the least, I want to say thank you to all my friends, mainly to those who always have been by my side despite the time and distances.

Italy, 29th May 2020

Alessandro Scarciglia

1 Introduction

1.1 Background

In the recent years, a steep increase of autonomous functionalities in systems has been observed, especially as far as the *automotive industry* is regarded. In order to direct such progress to the development of fully autonomous platforms, an outstanding improvement in the perception capability is needed. As an example, a reliable knowledge of the mobile robot position is a necessary information for computing a control action, in order to correctly interact with the external environment. In this regard, the problem of *localisation* represents a cornerstone for an adequate operation of the autonomous functionalities.

With the word *Odometry* ("*odos metron*", namely "*route measure*"), the use of data from the motion sensors is meant, in order to obtain an estimate of the robot pose (position and configuration) with respect to a known initial position. According to the type of sensors involved, it is possible to classify several odometry techniques which have been designed and validated through the years. In this regard, a clear overview is provided in Figure 1.

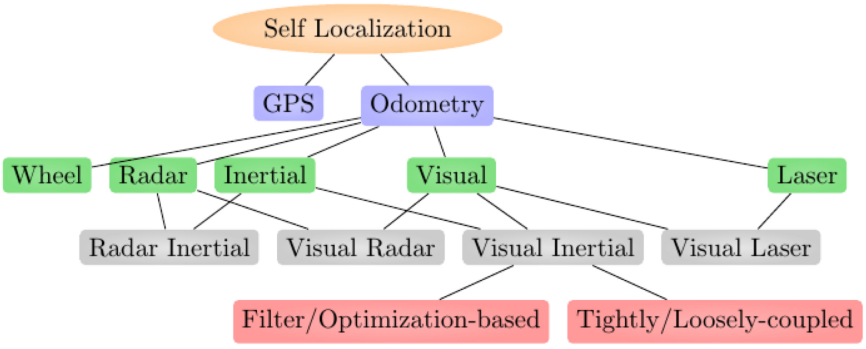


Figure 1: Overview of the possible solutions to the *self-localisation* problem. The picture is a courtesy of [1].

In 1973 US developed the Global Positioning System (GPS) for military purposes, but only in the early 1980s it has become one of the main source to accomplish civilian localisation tasks on the Earth. Though GPS demonstrates to provide a position estimate with a maximum precision in the order of centimetres, it is not able to return the configuration of the detected object. Moreover, the satellites signal can be easily compromised when passing through or being reflected from structures (such as walls, water surfaces, ground etc.), so it is

not reliable to adopt GPS as the only mean to recover the absolute pose of an autonomous platform.
According to [1], it is possible to group five different approaches of odometry, in order to side or replace the GPS in the self-localisation problem. Among those categories, the following research will focus mainly on *visual odometry* (VO) techniques.

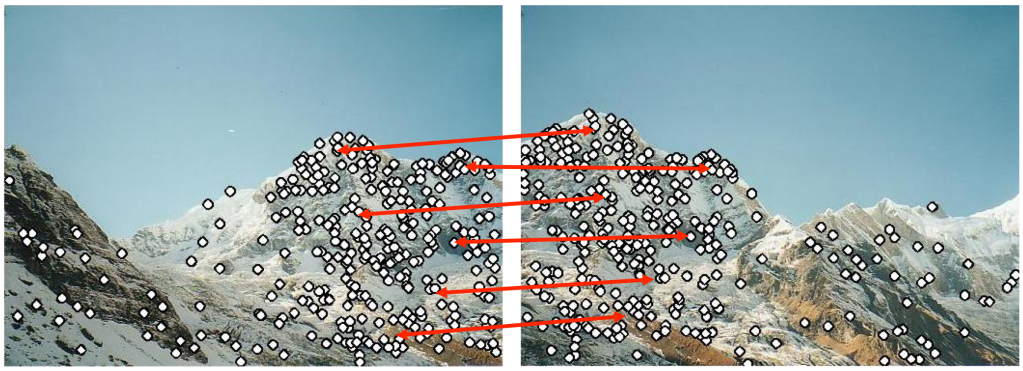


Figure 2: Example of features matching between two images [3].

Visual odometry estimates the pose of a mobile platform from a sequence of images. More generally, it can be seen as a derivation of a wider computer vision technique named after *Structure from Motion* (SfM), which is used to recover a 3D scene and the camera pose from a series of images.
The needed information is extracted from each image at the pixel level as a set of key points (e.g. corner points) by using a proper *feature detector*. Once the coordinates of such key points have been collected, it is then required to store the characteristics of each point (i.e. the gradients information about its neighbourhood levels of intensity) in a vector called *descriptor*. At the end of the process, the set of descriptors represents somehow the identity of the picture itself and it can be used for making comparisons between different images [2, 3], as it is shown in Figure 2. Once the features of a first image have been correctly matched with the features found in the next image, it is possible to estimate the homography matrix $\mathbf{H}$ which best transforms the first set of coordinates into the second one.
The shape of such transformation matrix depends on the dimension of the geometrical space from where the features have been extracted.

3D-3D correspondences If a stereo camera is employed and if the relative calibration parameters are known, the 3D absolute position of a feature can be obtained. Starting out from two stereo images, once the features have been detected, two sets of 3D points can be stored.

The absolute scale can be then computed solving a minimisation problem on the L_2 distance between the two sets.

3D-2D correspondences The minimisation of the re-projection error on the image plane demonstrates to be an improvement of the 3D-3D matching. Again, the calibration parameters are strictly necessary to back-project a 3D point onto the 2D image plane.
The general problem can be formally stated as it follows:

$$H_t^k = \underset{H_t^k}{\operatorname{argmin}} \sum_i \| \, p_k^i - P_{t-1}^i \, \|^2 \tag{1}$$

where p_k^i is the image measurement and P_{t-1}^i is the back-projection of the feature X_{t-1}^i onto the image I^t [1]. The problem is widely known as *PnP*, which stands for *Perspective-n-Points*. In this regard, to compute the camera pose a minimum of three 3D-2D correspondences are required (thus, P3P).

2D-2D correspondences Most of the times, no information about the third dimension is known and it is required to reconstruct the camera pose only working on the 2D image plane. Given two subsequent images, the homography matrix can be computed passing through the *essential matrix* $\mathbf{E}$, thus exploiting the epipolar constraint [4] as described in Figure 3.

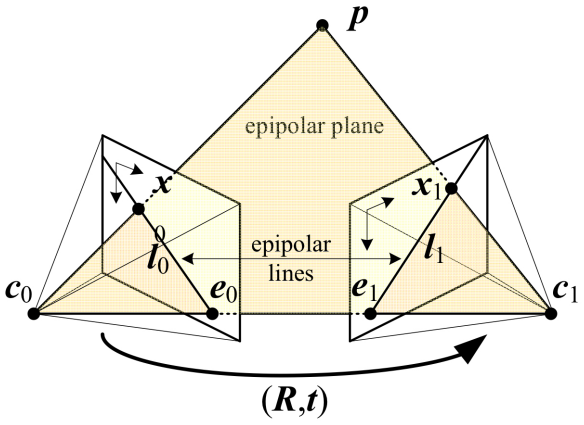


Figure 3: Epipolar geometry representation. The picture is a courtesy of [2].

Given $\mathbf{p}$, $\mathbf{x}$ and $\mathbf{x_1}$ laying on the same plane and thus imposing a null mixed product, the essential matrix can be derived with the Nister five-point algorithm [21]. Once $\mathbf{E} = [\mathbf{t}]_x\mathbf{R}$ is computed, the motion parameters are obtained by singular value decomposition (SVD). As a drawback, a pure 2-dimensional

approach does not return the absolute scale of the pose estimates. This problem can be solved by adopting hybrid odometry techniques in order to recover the real world scale from other sensors.

Apart from a feature-based method, a *direct approach* can be adopted for estimating a sequence of camera poses. In direct approaches, raw measurements regarding pixels (i.e. the spatial and temporal changes in intensity) are employed for computing the update in position and configuration of the camera. The apparent motion field obtained with such procedure is called *optical flow* (OF).

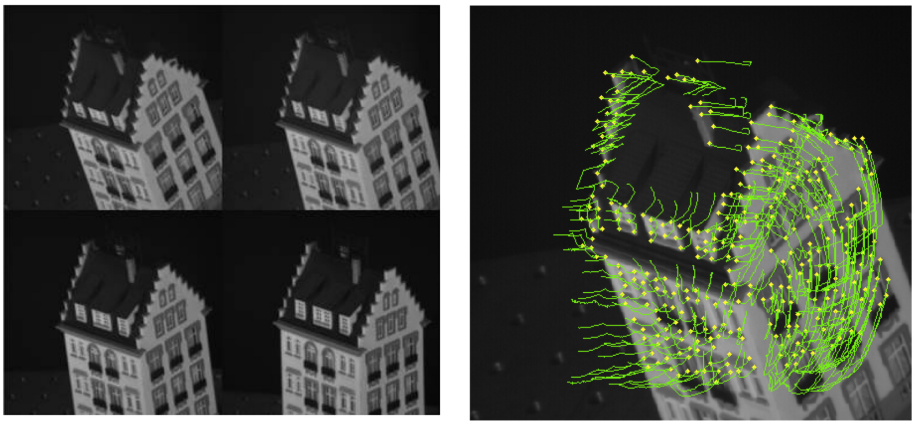


Figure 4: An example of optical flow. Some key points are tracked on four subsequent images of a building.

Among all the algorithms which are based on the computation of an OF, the Lukas-Kanade-Tomasi (LKT) method has a certain notoriety [5]. LKT takes for granted some hypotheses on a scanned $n \times m$ pixel neighbourhood, namely the brightness constancy and the local smoothness. On one hand, the smaller the neighbourhood (i.e. the size of the window employed to scan the image) the stronger the hypotheses. On the other hand, the smaller the window, the higher the risk of not detecting any motion (e.g. the barber pole illusion). Such a problem is named after *aperture problem* and it can be solved by custom-made optimisation of the neighbourhood size, according to the entity of the motion which is expected to be detected. Supposing to hold the hypotheses in a $n \times n$ window, the problem is formally posed as:

$$\begin{bmatrix} I_x(\mathbf{x_1}) & I_y(\mathbf{x_1}) \\ I_x(\mathbf{x_2}) & I_y(\mathbf{x_2}) \\ . & . \\ . & . \\ . & . \\ I_x(\mathbf{x_{n^2}}) & I_y(\mathbf{x_{n^2}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{x_1}) \\ I_t(\mathbf{x_2}) \\ . \\ . \\ . \\ I_t(\mathbf{x_{n^2}}) \end{bmatrix} \tag{2}$$

where each row is the equation obtained from the intensity gradient of a single pixel of the considered neighbourhood and the unknown vector $\begin{bmatrix} u & v \end{bmatrix}^T$ contains the components of the apparent motion along the image axes (in pixel units). As a drawback, the solution is sub-optimal since it makes use of n^2 equations to return a LS estimate of two components. Moreover, the computation involved in OFs are generally rather demanding for real-time odometry applications, thus a motion integration on an optical flow will not be taken into account in the present research work.

1.2 Objectives

The purpose of this thesis work is to design, validate and generate the code of a VO application to be deployed and tested real-time on an *electric all-terrain vehicle* (e-ATV). The subject platform is a *Polaris Ranger* [6] and it is exploited by the Autonomous System Research Group at Aalto University, in order to test and validate autonomous functionalities. Such category of vehicles have recently presented a significant use in both surveillance and forestry. They are particularly suitable for driving over irregular and unfamiliar terrains, also in non-optimal weather conditions. Moreover, an indisputable advantage of such platforms comes from the zero-emissions as well as their economical operation.

Figure 5: Polaris Ranger vehicle equipped with sensors for some autonomous functionalities.

As it is specified in previous works [7], the Polaris vehicle is already equipped with a wide range of sensors, actuators and electric modules which are necessary for implementing the autonomous functionalities. These include Programmable Logic Controller (PLC), Light Detection and Ranging (LiDAR), Automatic

Braking System (ABS), Synchronous Position, Attitude and Navigation (SPAN), embedded computers, wheel encoders and also a catadioptric omnidirectional camera for vision tasks.

The usage of an omnidirectional camera is one of the most incisive topic of the following research work. With respect to a conventional perspective planar camera model, a full FoV camera allows to detect a greater number of features, which are tracked on the image plane for longer periods. These details bring about an increase in the frame-level dataset size, which is then employed for estimating the frame-to-frame camera pose. Conversely, omnidirectional cameras introduce a consistent distortion of the image and are particularly hard to be assembled. Indeed, a small error on the geometric tolerances could bring to a failure in the calibration process. As far as the calibration is regarded, a well known disadvantage of large FoV cameras consists in a non-linear calibration function. A first attempt to calibrate the mounted camera has been lead by a group of students [8], but the computed calibration parameters came out to be lacking. In this research work, an attempt of non-linear calibration is made and results are presented by testing the calibration function directly in the developed VO application. Finally, a hybrid VO approach is adopted in order to cross over the calibration issue, since intrinsic camera parameters show a high sensitivity to geometric tolerances, mechanical vibrations as well as to changes in the working temperature [9].

1.3 Procedure

Following the general trend adopted for the validation of applications in the *automotive industry*, the current research work is structured as a V-shaped development flow.

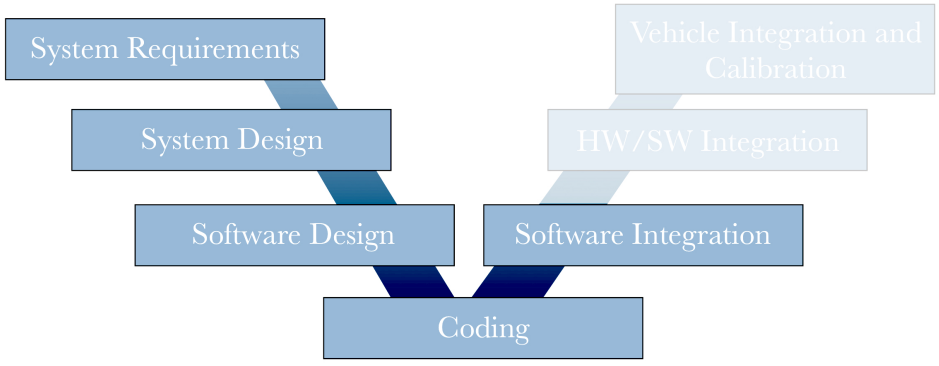


Figure 6: V-Shaped development flow truncated at the Software Integration.

After a deep analysis of the State-of-the-Art on VO algorithms in Chapter 2, a software design proposal will follow. As it is illustrated in Figure 6, the first step consists in the definition of the *system requirements*, namely all the constraints which are necessary to bound the acceptable solutions. As the Polaris Ranger is concerned, it is remarkable to fix the available hardware (i.e. the type of camera and eventual auxiliary sensors for hybrid configurations) as well as the fixed camera position (forward-facing). Finally, in order to direct the application to a possible real-time usage, it is necessary to guarantee that the execution time is at least lower than the camera frame rate of about 10 Hz.

In Chapter 3, an attempt of camera calibration is carried out as well as the acquisition and some pre-processing steps of the video frames. In Chapter 4 and Chapter 5 two possible *system designs* will be presented. At this stage, the application model is described with a platform-independent domain specific language and the main purpose consists in testing the validity of the results.

Once performances are evaluated, the best solution is pushed towards the *software design* process. Chapter 6 starts out with a Matlab script which contains the whole VO process (including some camera specific functions and the motion integration). The pure VO application to be deployed is collected in a Matlab function file to be then tested and turned into a C++ source. In the final steps, performances are evaluated in a Software-In-the-Loop validation process and the trend of the execution time is reported and discussed.

In the end, conclusions are presented in Chapter 7 along with possible sources of error that are highlighted in order to sustain and improve future works on the topic.

2 State-of-the-Art

In this chapter, a brief introduction to a camera mathematical model is provided, with a particular focus on catadioptric omnidirectional cameras. A good knowledge of a camera optical configuration is of primary importance when dealing with the projection and re-projection of image features and also it helps to understand eventual non-idealities.
In the second paragraph, the issue of the camera calibration is under the limelight. For perspective cameras, the calibration process can be posed as a linear problem which is easy to handle, whereas omnidirectional cameras need a solution to a non-linear estimate of the calibration parameters.
Finally, the last two paragraphs introduce the State-of-the-Art of some VO algorithms, which are taken into account for the subsequent application design.

2.1 Camera model

Pure VO techniques make use of images as the only source to compute the camera pose estimate. With this in mind, an accurate knowledge of the camera mathematical model is strongly required, in order to understand the logic behind the widespread VO algorithms. An outstanding overview of the most recent mathematical models employed to describe the camera projection topic is provided in [10].

Single viewpoint property From an optical point of view, cameras can be classified as *central* or *non-central*. Central cameras are characterised by a single effective viewpoint. The single viewpoint property is of primary importance in vision systems, because it allows to define one and only function to project and back-project points from the real world to the camera image plane.
In case of perspective cameras, such function is a linear application $\mathbf{P} \in \mathbb{R}^{3\times4}$, whereas for omnidirectional cameras it is a non-linear function $f \in \mathbb{R}^{3\times1}$ which varies according to the geometry of the lens and/or mirror employed.
Looking at Figure 8, it is straightforward to understand why central camera models are preferred over non-central ones as far as visual odometry applications are regarded. In non-central models, the optical rays intercepting the image plane do not intersect in a point. Indeed, a *caustic region* is defined, namely a geometrical locus of the points within which optical rays pass through. Such configuration does not allow to properly define a calibration function, thus non-central models are not easy to handle. In the following research work, only central vision systems are taken into account.

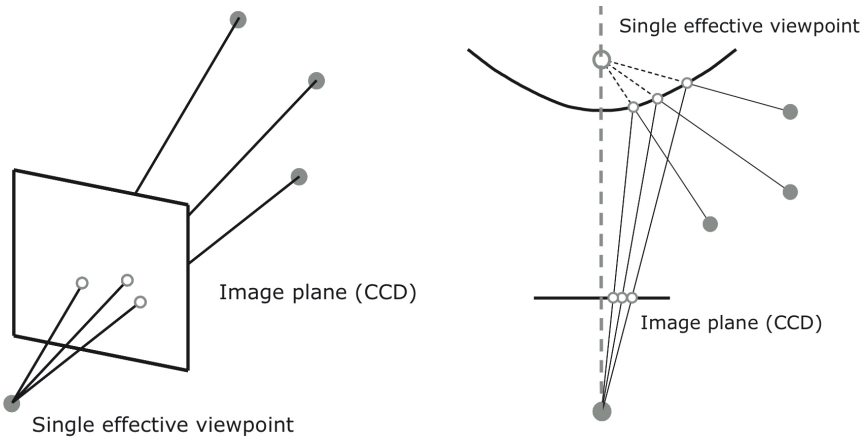


Figure 7: Single viewpoint property for perspective cameras (on the left) and omnidirectional cameras (on the right). The picture is a courtesy of [10].

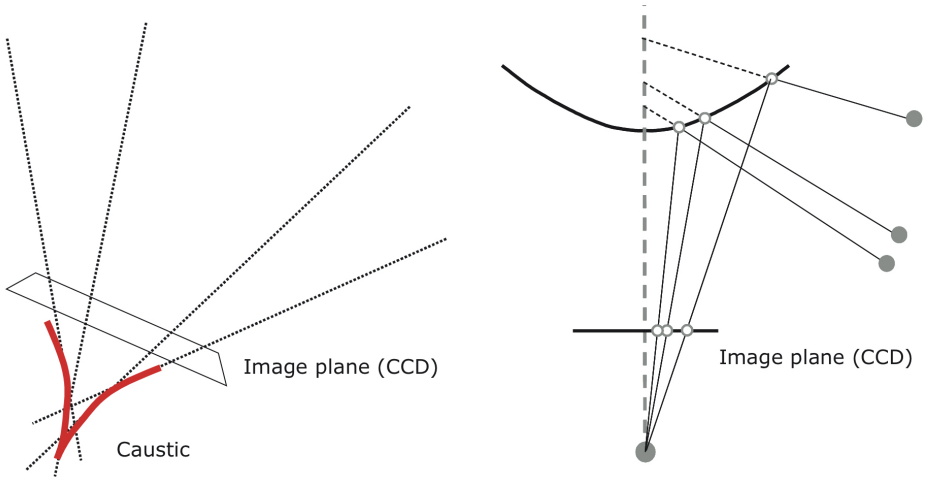


Figure 8: Caustic effect for perspective cameras (a) and catadioptric omnidirectional cameras (b). The picture is a courtesy of [10].

Omnidirectional camera model The study of central catadioptric cameras dates back to 1637 with René Descartes, but the most recent formulation is to be addressed to Baker and Nayar in 1998 [22], who translated the theory in a modern language and introduced it to the computer vision community. One of the main key point of their research is the identification of six catadioptric configurations which consist in a conventional lens (pin-hole camera model) associated to a mirror, whose shape is the class of swept conic sections.

- If an orthographic camera is associated to a paraboloidal mirror or if a perspective camera is coupled with planar, ellipsoidal or hyperboloidal mirror, a *non-degenerate* configuration is obtained.

- If a perspective camera is coupled with either a spherical or a conical mirror, a *degenerate* configuration is obtained.

In order to adopt a camera model with the single effective view point assumption, the assembly of a non-degenerate configuration is needed. From a mathematical perspective, a first unifying theory on catadioptric non-degenerated cameras was provided by Geyer and Dandiilidis in 2000 [12].
According to their research, every catadioptric system transformation is isomorphic to a perspective mapping from a sphere centred in the effective view point onto a plane, with the projection center laying on the perpendicular to the plane itself. Later, a more compact derivation to such approach was provided by Micusik [11], who derived two functions, h and g, which describe the whole mapping process starting out from a 3D sphere vector and ending up to the 2D image plane.

The great advantage of employing the *unit sphere image coordinates* consists in the generalisation of some computer vision concepts to whatever central perspective or omnidirectional camera. A standard perspective camera model is a linear application $\mathbf{P} \in \mathbb{R}^{3\times4}$ which maps a world point $\mathbf{X} = (X, Y, Z)$ into a 2D point on the image plane $\mathbf{x} = (x, y, 1)$ in homogeneous coordinates and up to a scale factor λ.

$$\lambda\mathbf{x} = \mathbf{P} \cdot \mathbf{X} \tag{3}$$

As shown in Figure 9, a standard perspective model does not allow to represent a large FoV, because each line passing through the view point intersect one and only one point. So it is not possible to make a distinction between two diametral positions.
With this in mind, the model proposed by [12] and detailed by [11] overcomes the stated problem by representing a world point as a unit vector with origin in the sphere centre (coincident with the effective view point). This way, the camera model turns into:

$$\lambda\mathbf{q} = \mathbf{P} \cdot \mathbf{X} \tag{4}$$

where $\mathbf{q} : \|\mathbf{q}\| = 1$ is a unit vector contained in the unit sphere and it is representative of the associated image point. Before diving deep into the explanation of the camera model, two hypotheses must be fixed a priori:

1. The mirror has been crafted with very high-precision geometrical tolerances about symmetry.

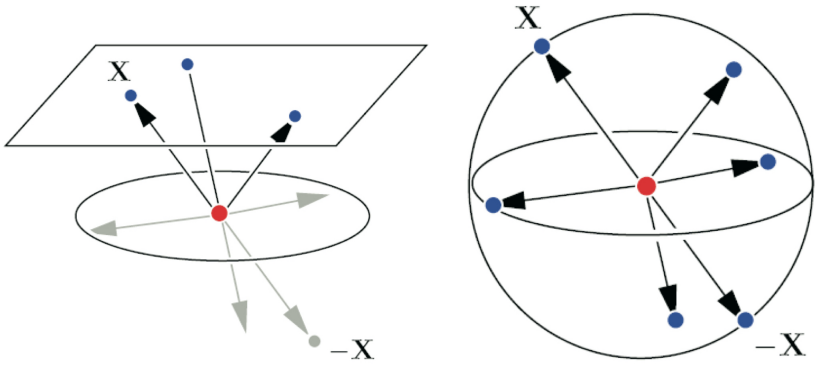


Figure 9: The limits of a perspective model (on the left) is overcome by employing the unit sphere coordinates (on the right). The picture is a courtesy of [11].

2. The perpendicularity of the mirror symmetry axis with respect to the image plane is guaranteed.

Taking for granted by manufacturing the stated conditions, it is possible to define the non-linear camera projection function starting from whatever world point $\mathbf{X} \in \mathbb{R}^3$. According to [11], there always exists a vector $\mathbf{p}'' = \begin{bmatrix}\mathbf{x}'' & z''\end{bmatrix}^T$ directed to the world point $\mathbf{X}$ such that:

$$\mathbf{p}'' = \begin{bmatrix} h(\|\mathbf{u}''\|)\mathbf{u}'' \\ g(\|\mathbf{u}''\|) \end{bmatrix} \tag{5}$$

where $\mathbf{u}'' \in \mathbb{R}^2$ is the two-dimensional vector and it is representative of the associated point on the sensor plane according to a reference system centred in the image itself. In order to improve the clarity of the geometric relationships, Figure 10 is provided.
The functions $h,\ g$ are $\mathbb{R} \to \mathbb{R}$ relationships whose dependency on $\mathbf{u}''$ is ensured by the hypotheses on symmetry stated above. The entire definition of such functions consists also of two parameters l and m which depend on the mirror geometry and whose possible values are specified in [12].

$$h(\|\mathbf{u}''\|) = \frac{l(l+m) + \sqrt{\|\mathbf{u}''\|^2(1-l^2) + (l+m)^2}}{\|\mathbf{u}''\|^2 + (l+m)^2} \tag{6}$$

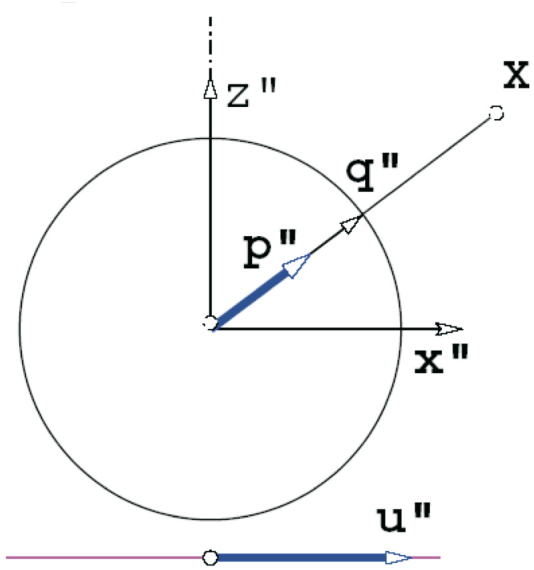


Figure 10: Micusik [11] catadioptric omnidirectional camera model.

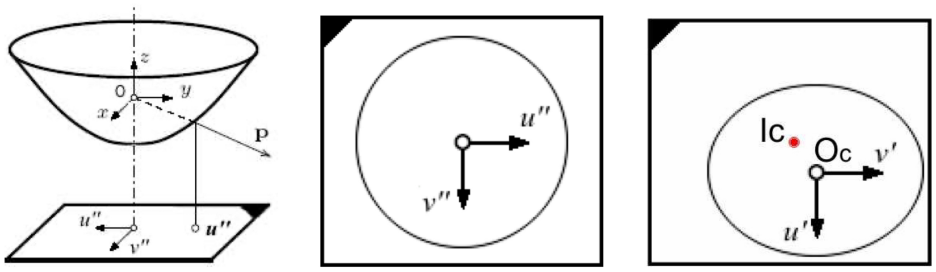
$$g(\|\mathbf{u}''\|) = \frac{l\|\mathbf{u}''\|^2 + (l+m)\sqrt{\|\mathbf{u}''\|^2(1-l^2) + (l+m)^2}}{\|\mathbf{u}''\|^2 + (l+m)^2} \tag{7}$$

From the sensor plane to the image plane After the reasoning made so far, it is time to clarify a distinction between the *sensor plane* and the *image plane*. The former is an auxiliary 2D surface, ideally perpendicular to the optical axis of the mirror and whose reference system is centred exactly on the intersection between the axis and the plane itself. The latter is the actual image plane which is compromised by the digitalisation error (e.g. pixel are not perfectly square) and is affected by manufacturing and/or assembly defects (e.g. perpendicularity with the optical axis is defined within a tolerance range). Moreover, the reference system is place on the up-left corner of the image.
If the distortion induced from the digitalisation is negligible and the tolerance intervals are tight enough, the step which brings a point from the sensor plane to the image plane can be represented by a 6DOF *affine transformation* $\mathbf{T} = [\mathbf{A}|\mathbf{t}]$,

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{u}'' = \mathbf{A}\mathbf{u}' + \mathbf{t} \tag{8}$$

where the vector $\mathbf{u}' \in \mathbb{R}^2$ represents the image plane coordinates and the vector $\mathbf{u}'' \in \mathbb{R}^2$ contains the sensor plane coordinates in pixels. The importance of defining the projection in terms of $\mathbf{u}'$ is considerable, since it is the only information available to the user.



a b c

Figure 11: The complete projection process from a world point to an image point is illustrated. a) From world coordinates in unit sphere metric coordinates. b) Sensor plane metric coordinates. c) Image plane pixels coordinates. The picture is a courtesy of [10].

To sum up, the non-linear function which project an image point $\mathbf{u}'$ to the corresponding $\mathbf{p}''$ vector can be defined as follows:

$$\lambda \mathbf{p}'' = \lambda \begin{bmatrix} h(\|\mathbf{A}\mathbf{u}' + \mathbf{t}\|)(\mathbf{A}\mathbf{u}' + \mathbf{t}) \\ g(\|\mathbf{A}\mathbf{u}' + \mathbf{t}\|) \end{bmatrix} = \mathbf{P} \cdot \mathbf{X} \tag{9}$$

2.2 Camera calibration

The Micusik model for catadioptric omnidirectional central cameras is not easy to handle when dealing with camera calibration issues. A complete definition of the non-linear $f(\mathbf{u}')$ function would require the computation of the functions h and g in order to guarantee the projection relationship.
As a first step, since the transformation is defined up to a scale factor, it is possible to divide the entries of f by the function h in order to isolate the dependence on $\mathbf{u}''$, that is to say $\mathbf{A}\mathbf{u}' + \mathbf{t}$. For instance, setting $h = 1$, the calibration problem consists in finding the function g which best suits the projection transformation. Formally,

$$\lambda \mathbf{p}'' = \lambda \begin{bmatrix} \mathbf{u}'' \\ g(\|\mathbf{u}''\|) \end{bmatrix} = \mathbf{P} \cdot \mathbf{X} \tag{10}$$

The Taylor model The procedure introduced by [10] proposes a N-degree polynomial for defining the function g, such that

$$g(\|\mathbf{u}''\|) = a_0 + a_1\|\mathbf{u}''\| + a_2\|\mathbf{u}''\|^2 + ... + a_N\|\mathbf{u}''\|^N \tag{11}$$

this way, the calibration problem aims at finding those N+1 coefficients $(a_0, a_1, ..., a_N)$ which best approximate the original shape of g in the neighbourhood of $\|\mathbf{u}''\| = 0$. However, it is remarkable to highlight that the function g admits a zero first derivative computed on $\|\mathbf{u}''\|$

$$\left.\frac{dg}{d\|\mathbf{u}''\|}\right|_{\|\mathbf{u}''\|=0} = 0 \tag{12}$$

thus, it is possible to reduce the calibration coefficients from N+1 to N. This way, the expression (5) can be turned into the simplified polynomial shape

$$\mathbf{p}'' = \begin{bmatrix} \mathbf{u}'' \\ a_0 + a_2\|\mathbf{u}''\|^2 + ... + a_N\|\mathbf{u}''\|^N \end{bmatrix} \tag{13}$$

and then the projection relationship (10) can be written as

$$\lambda\mathbf{p}'' = \lambda \begin{bmatrix} \mathbf{u}'' \\ a_0 + a_2\rho^2 + ... + a_N\rho^N \end{bmatrix} = \mathbf{P} \cdot \mathbf{X} \tag{14}$$

From this moment on, for simplicity with the term ρ the norm $\|\mathbf{u}''\|$ is intended.

Calibration process With the projection function given by equation (14), the calibration process consists in finding a suitable polynomial degree and the respective coefficients which best satisfy the transformation, as well as the affine transformation $[\mathbf{A}|\mathbf{t}]$ to map the image plane onto the sensor plane.
At the beginning, it is taken for granted that no distortion occurs at the pixels level, such that $\mathbf{u}'' = \mathbf{u}'$, which implies $\mathbf{A} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$. The correct transformation will be detected later by non-linear refinement. With this in

mind, it is possible to make use of equation (14) directly with the image points such that:

$$\lambda \mathbf{p}'' = \lambda \begin{bmatrix} u' \\ v' \\ a_0 + a_2 \rho'^2 + ... + a_N \rho'^N \end{bmatrix} = \mathbf{P} \cdot \mathbf{X} \tag{15}$$

For the camera calibration, an external pattern of known dimensions is needed (usually a checkboard is employed). If the external references are chosen on the same plane it is possible to set the third dimension of the j-th observation of the i-th pattern $Z_j^i = 0$.

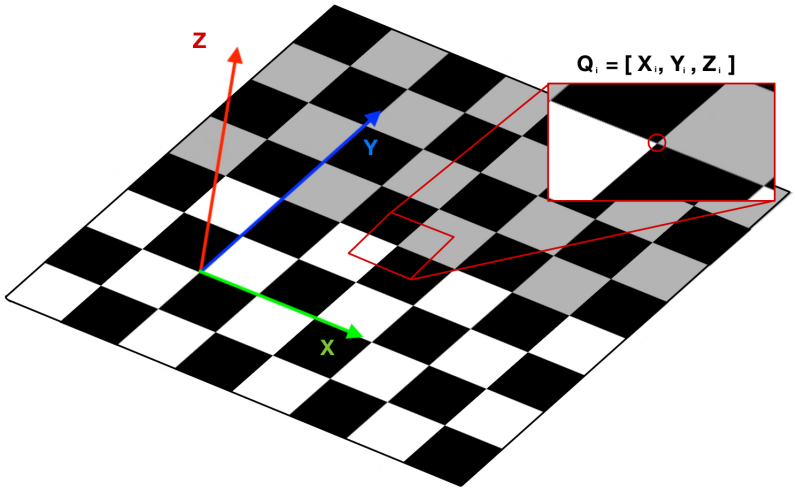


Figure 12: Checkerboard pattern for calibration with detail of an observation.

The following procedure refers to a single pattern, so the index i will be omitted for simplicity. Taking into account the expression (15), the first step is to estimate the *extrinsic parameters*, namely the rotation matrix and the translation vector which relate the camera pose to the checkerboard reference frame. Substituting in (15) a generic observation, one obtains:

$$\lambda_j \mathbf{p}''_j = \lambda_j \begin{bmatrix} u_j \\ v_j \\ a_0 + a_2 \rho_j^2 + ... + a_N \rho_j^N \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ 1 \end{bmatrix} \tag{16}$$

In order to get rid of the multiplier λ_j, $\mathbf{p}''_j$ is multiplied vectorially on both sides of the equation. This way, with equation (17) three conditions are obtained

but only one is linear in the parameters to be estimated. The linear equation (18) is obtained by extending the terms $\mathbf{r}_1$ and $\mathbf{r}_2$.

$$\lambda_j \cdot \mathbf{p''}_j \times \mathbf{p''}_j = \mathbf{p''}_j \times \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ 1 \end{bmatrix} = 0 \tag{17}$$

$$u_j(r_{21}X_j + r_{22}Y_j + t_2) - v_j(r_{11}X_j + r_{12}Y_j + t_1) = 0 \tag{18}$$

Once the condition from a single observation of a single pattern is obtained, it is possible to design a block matrix $\mathbf{M}$ which contains all the observations of the current pattern employed for the calibration. Likewise, the parameters to be estimated can be stacked in an unknown vector $\mathbf{\Pi} = \begin{bmatrix} r_{11} & r_{12} & r_{21} & r_{22} & t_1 & t_2 \end{bmatrix}^T$, and the problem

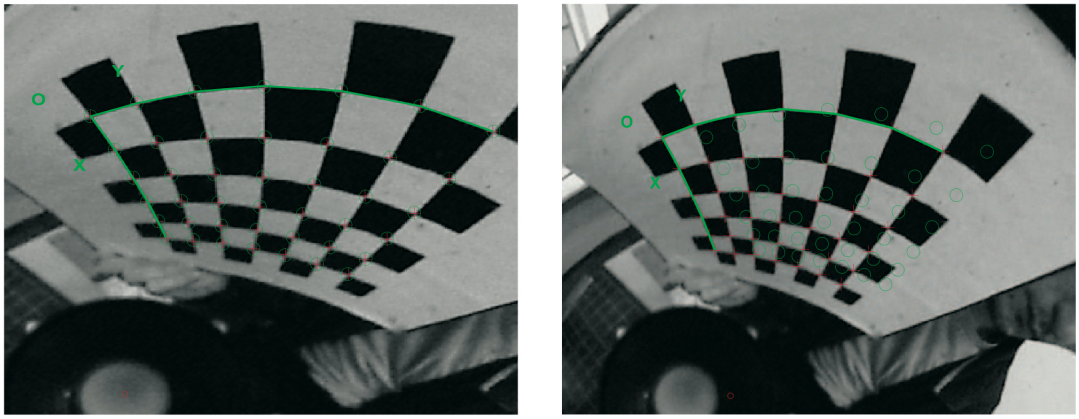
$$\mathbf{M} \cdot \mathbf{\Pi} = \mathbf{0} \tag{19}$$

can be solved with a least-squares criterion. The orthonormality of the solution is guaranteed by SVD. So far, all the extrinsic parameters but t_3 have been defined. In the next step of the calibration process, both t_3 and the *intrinsic parameters* are computed.

After the first bunch of parameters has been estimated, the first two conditions provided by (17) can be solved linearly by following a procedure analogue to the one already presented. This time, instead of using only L observations of the i-th pattern, all the K patterns are employed. Finally, in order to choose an appropriate N-degree polynomial, such procedure is iterated starting from $N = 2$ and terminates when the re-projection error is lower than a threshold ϵ. Once again, the re-projection is employed in order to refine recursively the initial estimate of $[\mathbf{A}|\mathbf{t}]$ [10].
Such procedure is the one implemented in the *Ocam Calib Toolbox* [13], which will be used in Chapter 3 to attempt to a calibration of the Polaris omnidirectional camera.

2.3 Pure VO approaches

Once the camera has been correctly calibrated, it can be employed on vehicles in order to accomplish the self-localisation of the platform. In *pure VO*



a b

Figure 13: Re-projection after a) and before b) the non-linear refinement of the affine transformation parameters. The picture is a courtesy of [10].

approaches, the camera is the only sensor which is used to collect the necessary data for solving the pose estimation problem, thus an accurate calibration is of primary importance. A recent increase in the research on omnidirectional visual odometry can be observed in the *space industry*, especially for planetary rover localisation. An outstanding overview of the algorithms employed for such purpose is presented in [14].

Carnegie Mellon's *Hyperion*, is a solar-powered rover which is employed for research on autonomous navigation in the Chile's Atacama Desert. Such platform is supposed to simulate the operations of a rover on Mars, thus the capability of self-localisation cannot be addressed to any GPS. Because of error integration, it has been demonstrated that classic wheel-odometry accumulates an error which is around the 5% of the covered path. Such error can be thought even greater if the terrain is slippy and rough like the martian soil. For such reasons, a VO module is implemented in order to guarantee a higher precision on the position estimate. The great advantage of using VO is that the error brought about the digitalisation is uncorrelated with respect to the one carried by other types of odometry. Moreover, the pose estimate recovered from a camera view is not affected by eventual slippage of the wheels on the ground. On this subject, the authors of [14] proposed two VO approaches: the former makes use of a *Robust Optical Flow* from salient features tracked in each pair of subsequent frames, while the latter implements an *Iterated Extended Kalman Filter* to estimate at each step both the camera pose and the 3D position of the features in the environment. The first method is based on the hypothesis of planarity of the ground, whereas the IEKF implementation does not require

any a priori assumption about the external environment.

Robust optical flow method Starting out from a tracked feature of image coordinates (u, v), it is possible to obtain the real world coordinates (x, y, z) up to an unknown scale factor λ.

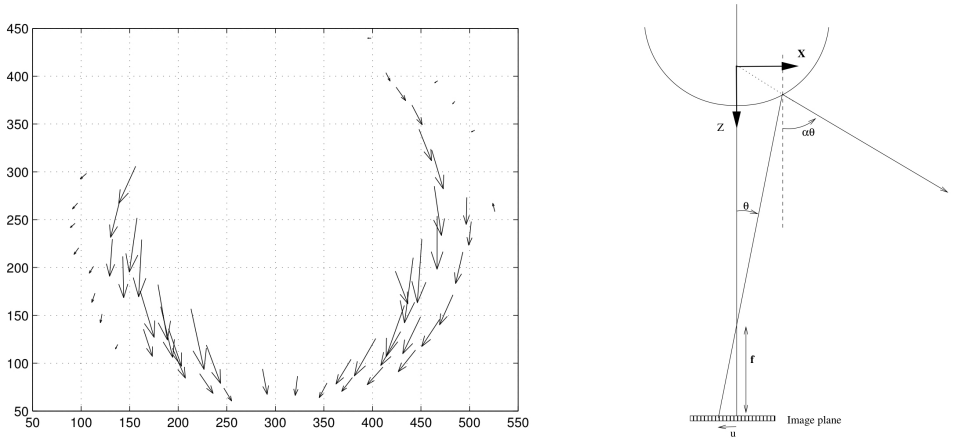


Figure 14: Left: the motion field of some tracked features between two frames. Right: the camera notation [14].

If one makes use of the similarity property between triangles, it is straightforward to get the angle $\theta = tan^{-1}\left(\frac{u}{f}\right)$ where f is the focal length. The same reasoning holds for computing the angle associated to the v coordinate. According to such notation, it is possible to define the corresponding points on the mirror reference frame:

$$a = tan\left[\alpha tan^{-1}\left(\frac{u}{f}\right)\right] sin\beta \tag{20}$$

$$b = tan\left[\alpha tan^{-1}\left(\frac{v}{f}\right)\right] cos\beta \tag{21}$$

It is further assumed that the greatest part of the tracked features belong to the ground as well as the ground itself can be approximated with an arbitrary geometrical plane of the kind $Ax + By + Cz = 1$, thus:

$$\lambda \begin{bmatrix} A & B & C \end{bmatrix} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = 1 \tag{22}$$

In a first moment, the mobile platform tracks a given feature of coordinates (u, v) which is projected onto the hypothetical ground plane with coordinates (x, y). When the vehicle moves, it computes an external displacement of $(\Delta x, \Delta y, \Delta\theta)$ so that the feature shifts from (x, y) to (x', y').
Finally, the back-projection of the feature returns the image point (u', v') and the optical flow can be computed like:

$$(d\hat{u}, d\hat{v}) = P(u, v, \{u_0, v_0, f, \alpha\}, \{\Delta x, \Delta y, \Delta\theta\}) \tag{23}$$

as a function of the initial conditions (u, v), the principal point (u_0, v_0), the focal length f, the elevation angle α, the displacement $(\Delta x, \Delta y)$ and the yaw $\Delta\theta$. The only parameter which is assumed to be known is the height h of the camera reference frame with respect to the inertial reference frame (i.e. the ground plane).

Once the displacement function is known, a cost function to be optimised is chosen. Such cost function e_1 is based on the median of the norm between the observed and the estimated displacement:

$$e_1 = median\Big[\sqrt{(du_i - d\hat{u}_i)^2 + (dv_i - d\hat{v}_i)^2}\Big] \tag{24}$$

The presented problem is a multivariate constrained nonlinear optimisation problem, which can be solved with some algorithms based on interior-point techniques. The use of constraints improves the robustness of the returned solution, as the magnitude of the displacement can be contained in a region of acceptable values in order to weaken the outliers effect. One of the main reasons which might bring failure to this method is that the *brightness constancy* is not guaranteed in every environment condition. Ideally, the transformation and the subsequent integration of the optical flow finally returns the actual motion in the world reference frame.

In one hand, the estimate of the intrinsic camera parameters is ideal, with a minimum precision of about 0.4 pixels. In the other hand, the odometry is rather failing with articulated paths and it looses precision very early.

Structure from Motion method The SfM proposed in [14] implies the use of an Iterated Extended Kalman Filter (IEKF) to estimate both the camera pose and the 3D position of the tracked features. The computational demand of such approach depends on how many features are identified on each frame. Indeed, the state vector is made up of $6 + 3p$ entries, which are the six-parameters

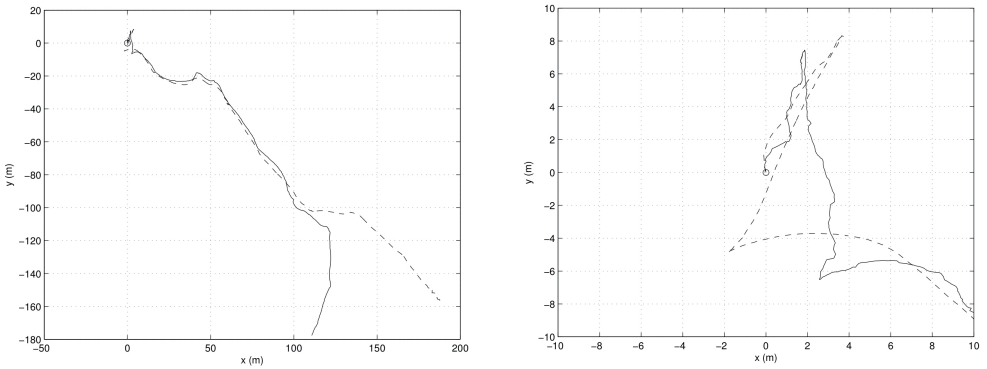


Figure 15: Results obtained from the robust optical flow method. The plots show the ground truth based on a very precise GPS measure (dashed line) and the estimated position (continuous line). The pictures are a courtesy of [14].

to define a 6DOF model of the camera, plus a set of world coordinates for a number p of features which are tracked on the current frame.

The propagation step uses a model to estimate the change in the camera pose and features position according to the information obtained with the previous step. With respect to a common EKF, this time no direct estimate is provided, but a large uncertainty α is added to the measures. Such an operation is carried out under the assumption of a small frame-to-frame movement and it gets weaker as α increases.

In the measurement step, the new observations are employed to return a new state estimate which best fits the old observations as well. In this model, the still visible tracked features are assumed to be equal to the re-projection of the 3D points with an additive Gaussian noise. Such re-projection function can be written like:

$$\mathbf{x}_j = \Pi(R(\rho)^T \mathbf{X}_j + \mathbf{t}) \tag{25}$$

where ρ and $\mathbf{t}$ represent respectively the set of Euler angles and the translation vector which transform the 3D point $\mathbf{X}_j$ into the re-projection $\mathbf{x}_j$ on the image plane. The refinement of the parameters estimate improves the precision with successive iterations, but an increasing number of features might bring about a consistent delay.
In the end, it is clearly observable that the second approach offers a consistent improvement with respect to the robust optical flow method, but at the expense of a larger computational demand.

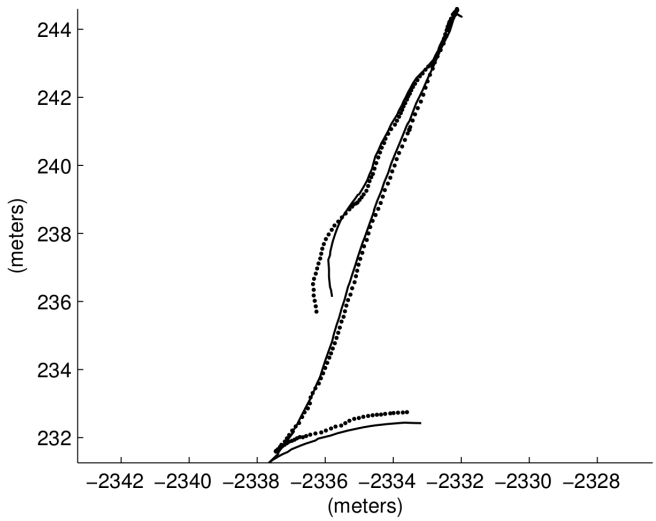


Figure 16: Results obtained with the IEKF implementation. The ground truth is obtained by a precise GPS measure (dashed line), whereas the position is estimated with SfM (continuous line). The picture is a courtesy of [14].

A faster pure VO algorithm is proposed by [15]. It consists in a feature-based approach which addresses the computation of the camera pose to the estimate of a transformation matrix. Moreover, the assumption of ground planarity holds, thus it is expected to collect a consistent number of features which belong to the ground.

Homography-based ground plane navigation A rigid body transformation can be thought as a combination of a rigid rotation described by a matrix $\mathbf{R}$ and a translation $\mathbf{T}$ in the three-dimensional world. When the vehicle moves, an arbitrary point of the scene $\mathbf{X}_1$ is transformed into a point $\mathbf{X}_2$ according to the relationship:

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{T} \tag{26}$$

Assuming that the ground can be modelled as a geometric plane, the point $\mathbf{X}_1$ must satisfy the geometrical constraint, formally posed like

$$\mathbf{n}^T\mathbf{X}_1 = h \tag{27}$$

where the vector $\mathbf{n} \in \mathbb{R}^3$ is the plane normal and the scale h is the distance from the camera reference frame to the ground. Given this relationship, it is possible to rewrite the transformation (26) in order to collect $\mathbf{X}_1$ on the left-side.

$$\mathbf{X}_2 = \Big(\mathbf{R} + \frac{\mathbf{T}\mathbf{n}^T}{h}\Big)\mathbf{X}_1 \tag{28}$$

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{T}\mathbf{n}^T}{h} \tag{29}$$

The matrix $\mathbf{H} \in \mathbb{R}^{3\times3}$ is called *homography matrix* and it represents the rigid rotation of a point $\mathbf{X}_1$ into a point $\mathbf{X}_2$ with the constraint that both $\mathbf{X}_1$ and $\mathbf{X}_2$ belong to the plane $\mathbf{n}^T\mathbf{X} = h$. The next step of the model definition consists in transforming the 3D world coordinates in the respective unit sphere normalised coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$. Such a step requires knowledge of the camera calibration function, in order to back-project the points from the image plane to the unit sphere.

$$\mathbf{x}_2 = \frac{\lambda_1}{\lambda_2}\mathbf{H}\mathbf{x}_1 = \lambda\mathbf{H}\mathbf{x}_1 = \mathbf{H}_L\mathbf{x}_1 \tag{30}$$

Actually, when the estimate of the homography matrix $\mathbf{H}_L$ is obtained, it is defined up to the depth factor λ. In order to obtain the actual transformation, the matrix $\mathbf{H}_L = \lambda\mathbf{H}$ should be divided by the second largest singular value $\sigma_2(\mathbf{H}_L)$. Once the homography is univocally defined, it needs to be decomposed in order to obtain the extrinsic motion parameters.
The decomposition can follow two paths: the former follows a 8DOF function of the points on the image and it is particularly suitable when features cover uniformly the frame, while the latter simplifies the motion with a 3DOF model (Euclidean transformation).
The decomposition of the 8DOF model can be carried out with the *Triggs algorithm* [20]. Such procedure returns two possible solutions and the respective plane normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$. The best solution is the one whose plane normal is closer to the ground model $\mathbf{n}^T = \begin{bmatrix}0 & 0 & -1\end{bmatrix}$. The selected translation vector is then projected onto the ground plane.

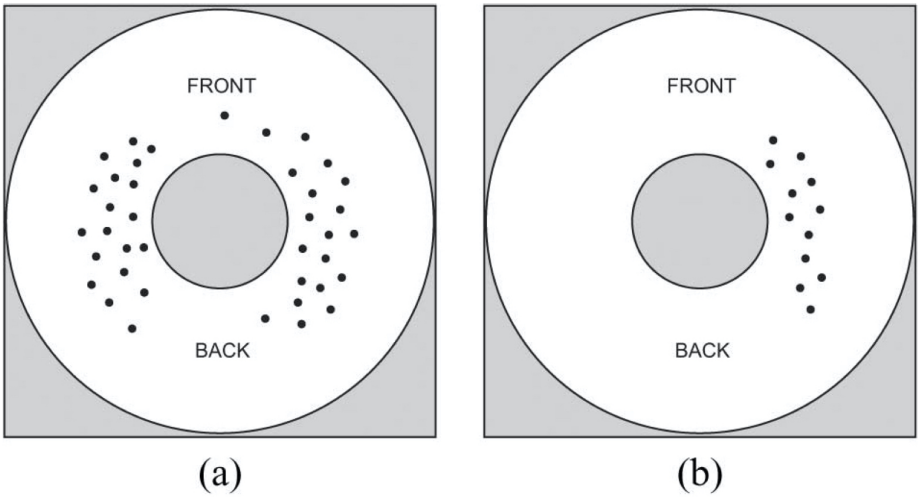


Figure 17: a) Uniform distribution of features. b) Non-uniform distribution.

If the features are tracked similarly to Figure 17.b, then the matrix $\mathbf{H}$ is assumed to be an Euclidean transformation of the shape

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{T}\mathbf{n}^T}{h} = \begin{bmatrix} \cos\theta & -\sin\theta & -t_1/h \\ \sin\theta & \cos\theta & -t_2/h \\ 0 & 0 & 1 \end{bmatrix} \tag{31}$$

This way, the motion parameters can be estimated in a least-square sense computing the pseudoinverse of the matrix of the matched features coordinates. The translation vector $\mathbf{t}$ can be solved simply by multiplying the third column by the absolute scale factor h, whereas the rotation is obtained by SVD, since the solution provided by the sub-matrix $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ may not be orthonormal.
The robust estimate of the homography matrix starting out from the features coordinates is made with a *Random Sample Consensus* (RANSAC) approach. Such procedure can be thought of as an improvement of the least-squares method. Indeed, the LS estimate suffers from outliers which are isolated with respect to a uniform noise distribution (i.e. non-white noise). Vice versa, RANSAC offers a statistical method to reject outliers at the expense of a given number of iterations on the dataset. Such estimation technique is employed in computer vision because in a random subset of features a full accuracy in the matching process is not guaranteed. According to [15], the procedure to be followed in order to come up with a correct estimate of the transformation is the following:

1 - From the set $\mathbf{A}$ of all the features matches, a random subset of at least

four pairs is selected to initialise the homography model in a least-squared sense. Since the matrix $\mathbf{H}$ presents 8DOF, at least four pairs are needed to univocally estimate a model.

2 - Applying the instantiated transformation $\mathbf{H}$ to all the set of pairs, it is possible to collect in the set $\mathbf{S}_1$ all those features whose transformation is close to the corresponding matching within an error tolerance d. The set $\mathbf{S}_1$ is the first *consensus set*.

3 - If the new consensus set is larger than a threshold value t, which is function of the expected outliers in $\mathbf{A}$, it is used to update the current model of the homography.

4 - If the consensus set $\mathbf{S}_1$ is smaller than the fixed threshold, another random set $\mathbf{S}_2$ is chosen from $\mathbf{A}$ and the process restarts from 1.

Apart from the homography, another important model to be chosen is the error function to be adopted in order to asset if a features pair is allowed to enter the consensus set. In [15], this function is the sum of the squared norm of the re-projection differences, namely:

$$err^i = \|\mathbf{x}_2^i - \mathbf{H}\mathbf{x}_1^i\|^2 + \|\mathbf{x}_1^i - \mathbf{H}^{-1}\mathbf{x}_2^i\|^2 \tag{32}$$

and the corresponding threshold d is computed statistically on the error itself by mean of *Median Absolute Deviation* (MAD). Such function is a robust estimate of the statistical dispersion. In the cited paper, the definition of the error threshold is chosen to be

$$d = 5.2MAD \tag{33}$$

$$MAD(err) = median_i\{|err_i - median_j(err_j)|\} \tag{34}$$

Because of the image distortion and the sensitivity of the camera intrinsic parameters to external factors, the heading estimate obtained by orthonormalisation of the sub-matrix $\mathbf{Q} \in \mathbb{R}^{2\times2}$ is not so accurate [15]. For such reason, another method to estimate the frame-to-frame heading is proposed by the author.

The *visual compass* is a direct VO approach which makes use of the pixel intensity value in order to detect an optical flow, thus a possible lateral shift. In a first moment, this movement is measured in pixels, then it can be translated in degrees according to a constant which relates the number of pixels to the

camera FoV. The visual compass practically consists in scanning an image portion with a fixed size window. A pure rotation would cause a shift of the pixels columns in the window area, thus a minimisation problem on the pixels intensity can be set in order to obtain the rotation information under the shape of number of columns.

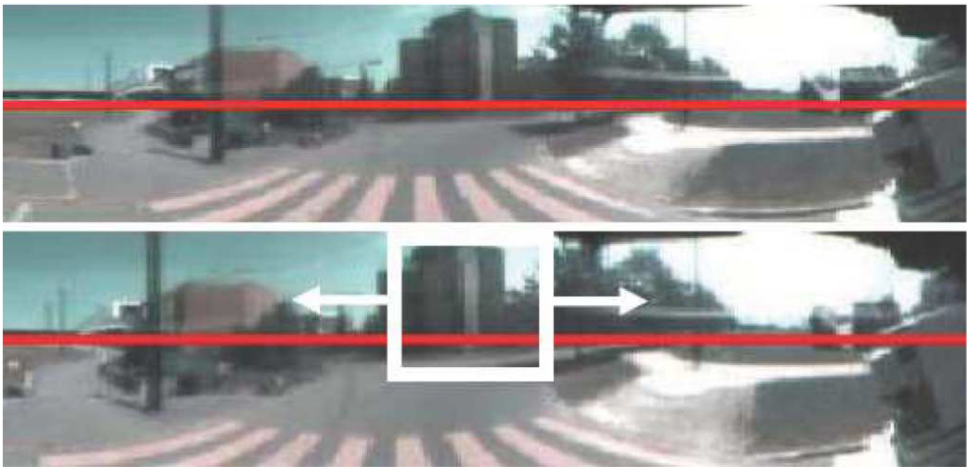


Figure 18: In white, the window employed in [15] to scan an area on the red horizon.

The use of the visual compass is not much robust in case of brightness changes or in environment with a flat background. Conversely, it results rather accurate in urban environments which present an articulated background because of buildings, trees etc. The choice of the window size and position is of crucial importance if a reliable heading is needed. The authors of [15] found out that the narrower the FoV, the better the heading estimate. In this regard, practical results are shown in Figure 19.

Though some general guidelines in order to choose the scanning window have been provided, in the later implementation it is found that the correct dimensioning strongly depends on many settings parameters (camera position, image resolution, image pre-processing etc.), thus the right size and position must be defined case by case. Another limit of the visual compass consists in the vehicle steering speed. If the platform moves slowly, it is more probable to detect correctly the frame-to-frame columns shift. Vice versa, when the yawing of the vehicle is too fast, the heading estimate lacks in reliability.

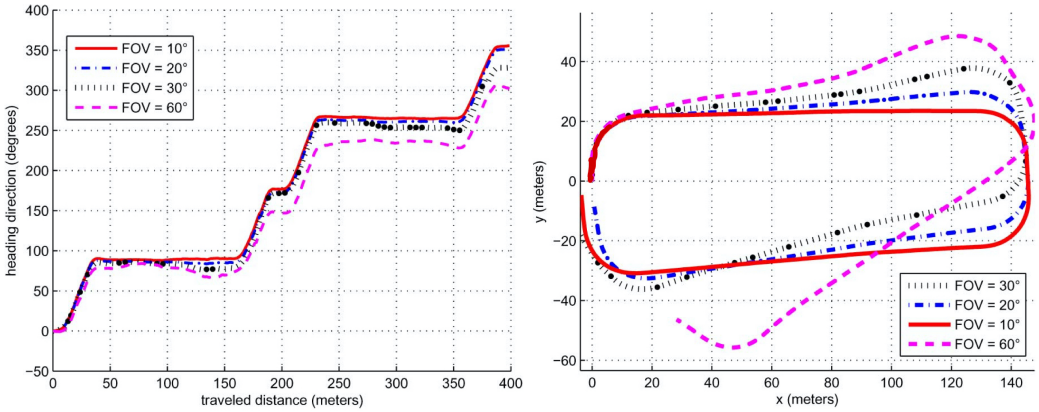


Figure 19: Path and heading estimates with different values of FoV. The plots are a courtesy of [15].

2.4 Hybrid VO approaches

If no information regarding the relationship between the mobile platform and the external world is known in advance (e.g. the distance h from the camera to the ground), it is impossible to recover the absolute scale of the path by only using two images of the same camera. For such reason and more, it is often convenient to employ a VO approach which also makes use of some data estimated by sensors other than cameras.
In the next paragraph, the scientific paper [16] by D. Scaramuzza is presented. In his research, he parametrised a minimal vehicle model in order to speed-up the execution time of the application. As a drawback, this simplification does not let a proper recovery of the absolute scale, which can be then corrected with the vehicle ground speed detected from other on-board sensors.

1-Point-RANSAC SfM method The computational cost of VO algorithms is mainly addressed to the extraction of the features and their matching. However in the matching process it is not guaranteed a full success in features association, thus a robust estimation technique like RANSAC is employed. The number of iterations introduced by the RANSAC algorithm is a function of the *minimal seed set*, namely the smaller number of point pairs which are needed to univocally establish the model. In this regard, a very detailed model of the platform takes longer with respect to a minimal model parametrisation.
In the last few years, a very established method for removing outliers from the main dataset is the *5-Points-RANSAC* algorithm, which needs at least 5 point-to-point matchings to verify the hypothesis.
In order to reduce the minimal seed set, it is possible to exploit the non-holonomic constraint of the platform according to the general *Ackermann*

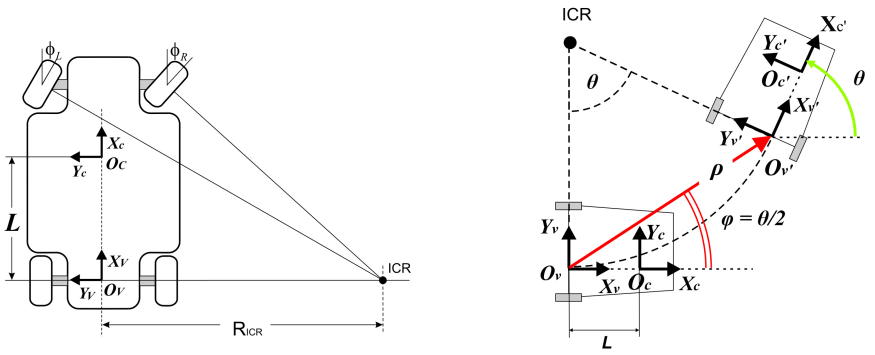


Figure 20: On the left, the Ackermann steering principle. On the right, the illustration of the platform motion model. The pictures are a courtesy of [16].

steering principle shown in Figure 20. According to such principle, when the platform can be modelled as a bicycle, all the wheels rotation axes converge in a point named after *instantaneous centre of rotation*, in short ICR. Given this assumption, it is possible to get rid of some extra-parameters and a general speed-up in the outliers removal is observed.

Let $\mathbf{p} = (x, y, z)$ and $\mathbf{p}' = (x', y', z')$ be the matched features coordinates between two successive frames. In order not to loose generality on the type of camera employed, the point coordinates are projected onto the unit sphere reference frame, such that $\|\mathbf{p}\| = \|\mathbf{p}'\| = 1$. If one makes use of a 2D-2D points correspondence, the epipolar constraint can be written as:

$$\mathbf{p}'^{T}\mathbf{E}\mathbf{p} = 0 \tag{35}$$

with $\mathbf{E}$ the essential matrix defined as $\mathbf{E} = [\mathbf{t}]_x\mathbf{R}$ and $[\mathbf{t}]_x$ the skew symmetric matrix of the translation components

$$[\mathbf{t}]_x = \begin{bmatrix} 0 & -t_x & t_y \\ t_x & 0 & -t_z \\ -t_y & t_z & 0 \end{bmatrix} \tag{36}$$

With reference to Figure 20, the camera and the vehicle reference frames are linked by a rigid relationship (forward-facing and distance L), thus the rotation matrix which describes the platform change in pose is also valid for the camera

model. With this in mind, it is possible to explicitly write the essential matrix:

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & \rho\sin(\frac{\theta}{2}) - L\sin(\theta) \\ 0 & 0 & L + \rho\cos(\frac{\theta}{2}) - L\cos(\theta) \\ L\sin(\theta) + \rho\sin(\frac{\theta}{2}) & L - \rho\cos(\frac{\theta}{2}) - L\cos(\theta) & 0 \end{bmatrix} \tag{37}$$

An interesting simplification occurs when the camera is mounted on the same vertical axis of the platform reference frame (i.e. the centre of the rear axis). This way it is possible to set $L = 0$ and the essential matrix can be written in the shape:

$$\mathbf{E} = \rho \cdot \begin{bmatrix} 0 & 0 & \sin(\frac{\theta}{2}) \\ 0 & 0 & \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & -\cos(\frac{\theta}{2}) & 0 \end{bmatrix} \tag{38}$$

It suddenly comes up that the term ρ is a multiplicative factor which is lost when substituting the essential matrix in the epipolar constraint (35). For such a reason, the presented method does not allow to recover the absolute scale of the path. The implementation of the RANSAC algorithm consists in initialising the model with just one features match:

$$\theta = -2\tan^{-1}\left(\frac{y'z - z'y}{x'z + z'x}\right) \tag{39}$$

and then using the first obtained hypothesis of $\mathbf{E}$ to build the consensus set. Once the iterations are completed, it is possible to refine the essential matrix model with a LS estimate on the pairs contained in the largest consensus set found:

$$\begin{bmatrix} x'_1z_1 + z'_1x_1 & y'_1z_1 - z'_1y_1 \\ . & . \\ . & . \\ . & . \\ x'_nz_n + z'_nx_n & y'_nz_n - z'_ny_n \end{bmatrix} \begin{bmatrix} \sin(\frac{\theta}{2}) \\ \\ \cos(\frac{\theta}{2}) \end{bmatrix} = \begin{bmatrix} 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \tag{40}$$

Finally, the matrix of the coordinates $\mathbf{D}$ is decomposed by SVD and the solution is the eigenvector associated to the smallest eigenvalue. Such vector $\mathbf{e}^*$ is subject to $\|\mathbf{e}^*\| = 1$, so the solution is already orthonormal. This way, the rotation angle θ can be computed directly from $\mathbf{e}^*$. The presented procedure is strongly

based on the hypothesis of $L = 0$, however it has been demonstrated that it can be employed also for small camera-vehicle distances. Moreover, as stated in [16], such parametrisation is rather reliable especially when the yaw angle is relatively small ($\theta < 10$ deg).

An even simpler method for removing the outliers consists in adopting a voting system based on histograms. Such a method allows the number of iterations to be reduced to zero, thus removing the delays introduced by the outlier removal process.

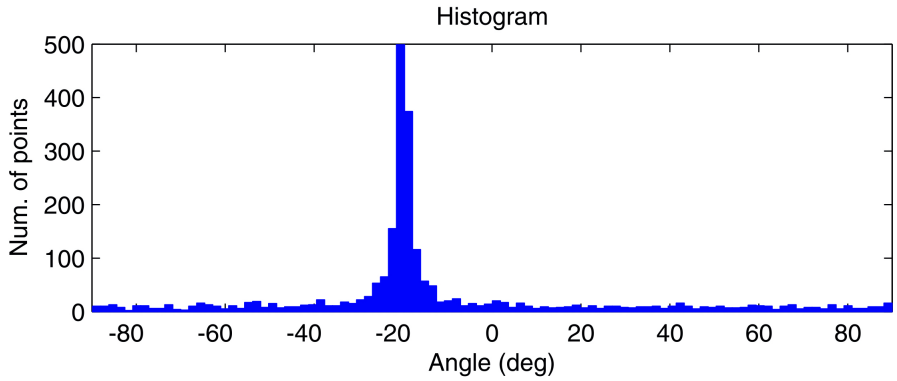


Figure 21: Voting system on the frame-to-frame yaw angle. The picture is a courtesy of [16].

With reference to Figure 21, each angle has been computed with only one feature matching, according to the the equation (39). Once a discretisation has been fixed, it is possible to count the number of pairs which return the same angle estimate. A robust estimate consists in selecting the median of the distribution. When the distribution variance is too high, a RANSAC implementation provides a more accurate result.

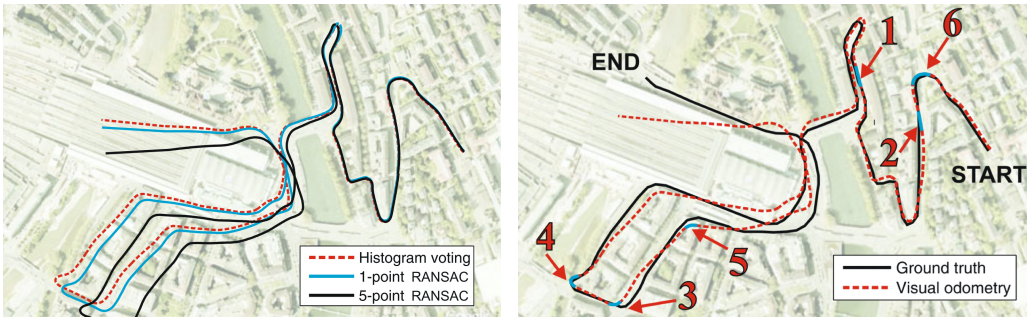


Figure 22: On the left, the odometry obtained with different outliers removal approaches. On the right, the comparison bewteen the histogram voting estimate and the ground truth. The pictures are a courtesy of [16].

It is clearly shown in Figure 22 that 1-Point-RANSAC and histogram voting techniques both improve the precision of the localisation and allows a real-time execution. The recovery of the absolute scale can be obtained by measuring the speed information from other sensors (e.g. by means of wheel odometry). Such speed is then mapped into a displacement according to the camera frame rate and is projected along the heading direction. For the following reason, the presented procedure is not properly a pure VO method.

3 Camera calibration & Image pre-processing

In this chapter, an attempt of camera calibration is carried out by mean of the OCam Calib Toolbox by D. Scaramuzza. Once the polynomial degree is fixed and the coefficients have been computed, two custom-made Matlab functions are designed to both adjust the centre on the image plane and back-project the image points onto the unit sphere. In a second moment, some necessary pre-processing steps about the video frames are highlighted.

3.1 Camera calibration

In order to correctly estimate the camera pose, it is required to know the function which allows to project a world point $\mathbf{X} \in \mathbb{R}^3$ into a point on the image plane $\mathbf{x} \in \mathbb{R}^2$ and vice versa. The theory behind the omnidirectional camera calibration has already been discussed in Chapter 3 and here it is applied thanks to the *OCam Calib Toolbox* by D. Scaramuzza [13]. In order to get a good estimate, a set of 15 pictures taken in a previous research work [8] is employed. The checkerboard is a 6×4 grid and it is shown at different positions on the camera field of view. The camera configuration consists of a Basler ace aca 1600-20gm coupled with a parabolic omnidirectional mirror VS-C450MR-TK.

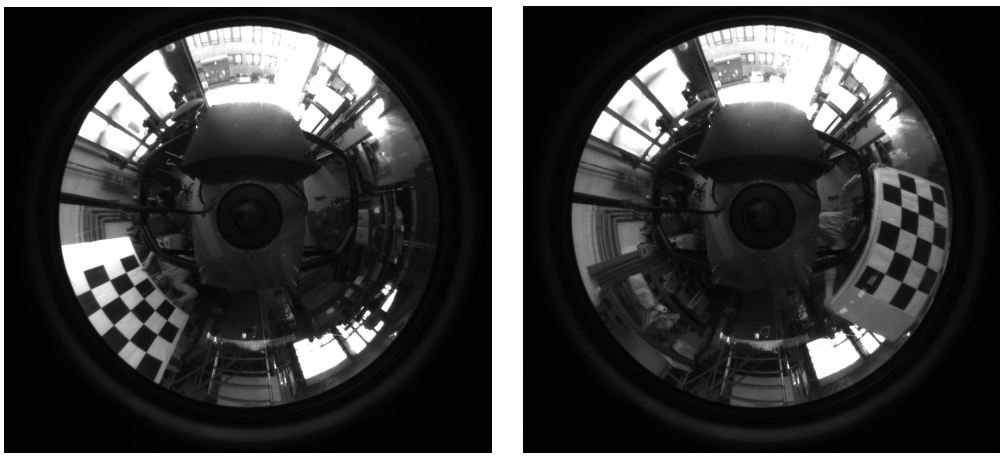


Figure 23: Two samples of the pictures employed for the calibration purpose.

The toolbox also implements an automatic corner detector in order to autonomously track the observations on each pattern. However, such functionality may be not so reliable if the checkerboard is not so close to the camera, thus some pictures have been tracked manually.
It is important to keep in mind that the order of the observations must be kept

unvaried among the patterns, this way the toolbox is able also to return the extrinsic camera parameters. For clarity, the origin and the axes direction is imposed like it is shown in Figure 24.

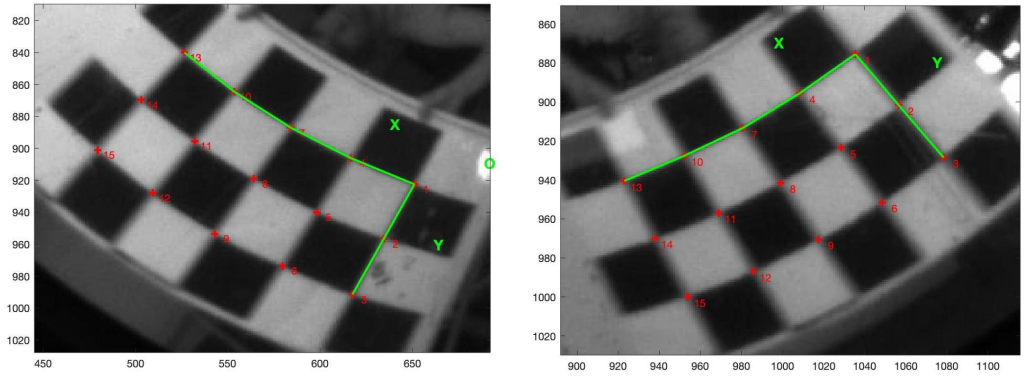


Figure 24: Corners detection and axes orientation among patterns.

Once enough patterns have been analysed and a consistent number of observations have been found, it is possible to recover the polynomial according to the Taylor model. Generally, if one selects a 4-th order polynomial a good model of the camera is returned. Fixing the degree at $N = 4$, it is asked to compute the four coefficients to univocally determine the polynomial shape.

$\mathbf{a_0}$	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$
$-3.627 \cdot 10^2$	0.0000	0.0013	$-8.938 \cdot 10^{-7}$	$4.139 \cdot 10^{-10}$

At the end of the calibration, all the parameters needed for univocally determine the projection function are collected by default in the structure titled *calib_data.ocam_model*. Here it is possible to find the polynomial coefficients *ocam_model.ss*, the affine transformation parameters *ocam_model.c*, *ocam_model.d* and *ocam_model.e* and also the image centre coordinates in pixels *ocam_model.xc* and *ocam_model.yc*. Such arguments are necessary in order to feed the functions which operate the affine transformation as well as the normalisation of the coordinates onto a unit sphere vector.

Affine transformation An affine transformation is an automorphism of an affine space. Such geometrical function maps affine spaces onto themselves (point to point, line to line, plane to plane) and does not preserve angles between entities. The only thing that is kept unmodified is the ratio of distances of the points lying on a straight line. Such transformation brings about a shift and a shear effect in order to transform the sensor plane into the image plane. The

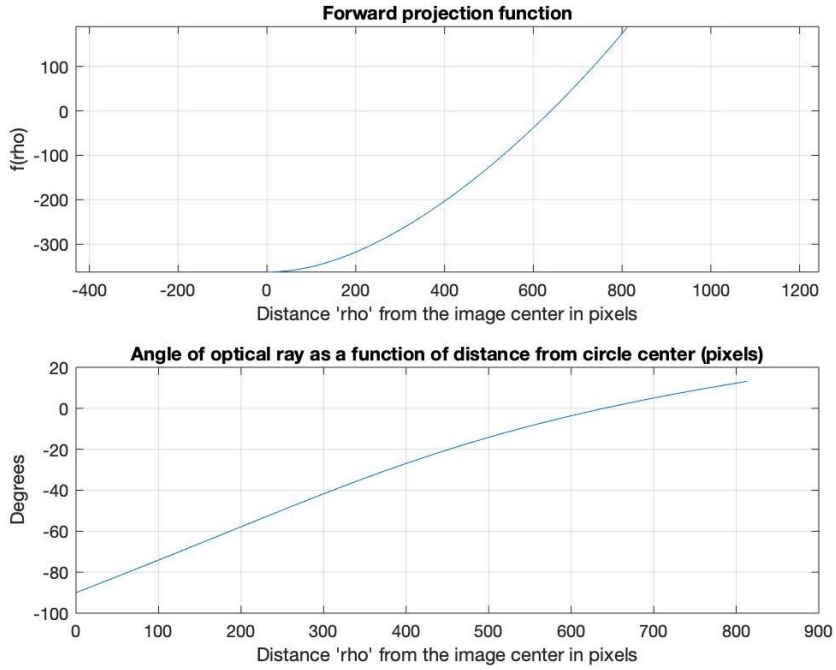


Figure 25: Calibration results returned by the OCam Calib Toolbox.

parameters returned by the toolbox are to be employed like it is shown:

$$\begin{bmatrix} u'' \\ v'' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} + \begin{bmatrix} -x_c \\ -y_c \end{bmatrix} \tag{41}$$

This way it is possible to map a feature from the image plane (u', v') onto the sensor plane (u'', v''). Such procedure has been implemented in the Matlab function *AffineTrans.m*, to be used as it shown here:

$$\mathbf{p}_{sp} = \mathit{AffineTrans}(\mathbf{p}_{ip},\ ac) \tag{42}$$

where $\mathbf{p}_{sp}$ and $\mathbf{p}_{ip}$ are the feature coordinates respectively on the sensor plane and the image plane and ac is an array which contains the transformation parameters in the order:

$$ac = \begin{bmatrix} c, & d, & e, & x_c, & y_c \end{bmatrix} \tag{43}$$

Projection onto the unit sphere Once the image coordinates have been changed in sensor plane coordinates, it is possible to make use of the computed Taylor model to formally write the projection function and to transform the 2D points in 3D unitary vectors on the sphere. At first, the vector $\mathbf{p}''$ is recovered

from the transformation

$$\mathbf{p}'' = \begin{bmatrix} u'' \\ v'' \\ -3.627 \cdot 10^2 + 0.0013\rho^2 - 8.938 \cdot 10^{-7}\rho^3 + 4.139 \cdot 10^{-10}\rho^4 \end{bmatrix} \tag{44}$$

keeping in mind that $\rho = \sqrt{u''^2 + v''^2}$. As specified in Chapter 2, such vector is parallel to the unitary vector $\mathbf{q}''$ which is employed in the omnidirectional camera model. In this regard, to obtain the feature coordinates in the unit sphere reference frame it is needed to normalise the already computed vector, such that $\mathbf{q}'' = \mathbf{p}''/\|\mathbf{p}''\|$. Such procedure has been implemented in the Matlab function *ToUnitSphere.m*, to used like it is shown:

$$\mathbf{q}'' = ToUnitSphere(\mathbf{p}'',\ pc) \tag{45}$$

where pc is an array which contains the polynomial coefficients

$$pc = \begin{bmatrix} a_0, & a_1, & a_2, & a_3, & a_4 \end{bmatrix} \tag{46}$$

3.2 Image pre-processing

RGB to Grey Each image coming from the omnidirectional camera can be thought as a function from $\mathbb{R}^3$ (R, G and B matrices) to $\mathbb{R}$ (actual pixel intensity). However, the information about the color repartition is unnecessary for odometry purposes, so it is preferred to work on grey-scaled images. This problem can be easily solved by the Matlab function *rgb2grey.m* which adopts the following formula to convert the domain of the image from separated RGB indexes to a single grey scale intensity:

$$I_{grey} = 0.2989 \cdot I_{red} + 0.5870 \cdot I_{green} + 0.1140 \cdot I_{blu} \tag{47}$$

The effects on the image histogram are shown in Figure 26.
Once the conversion has been made, the image can be simply represented by a unique matrix whose entries range from 0 to 2^{n-1}, where n is the number of bits employed to discretise the grey scale. In the present case, the image level are represented in *uint8* format (unsigned integer 8-bit), thus 256 grey levels are available.

Image resolution The employed omnidirectional camera returns images with a resolution of 1236×1626. On one hand, a great amount of pixels improves the accuracy of the features tracking, since a more refined grid of

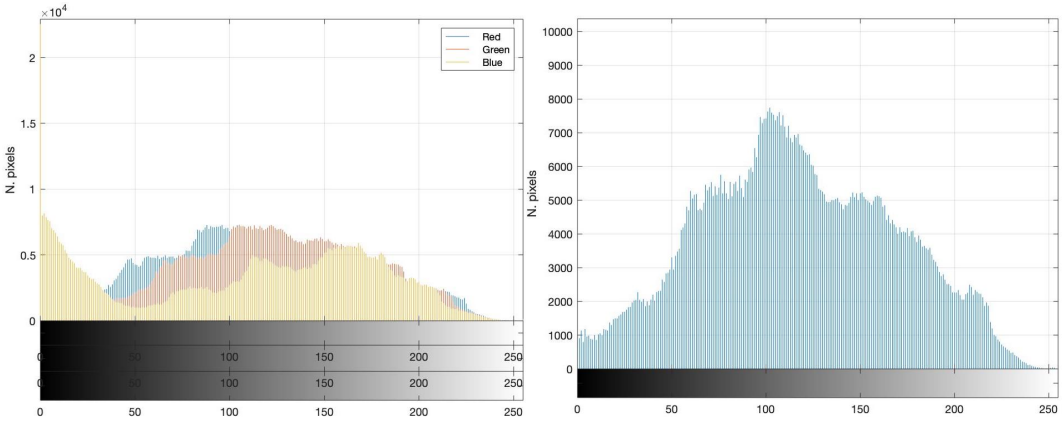


Figure 26: RGB to Grey scale conversion effect on the image histogram.

coordinates is available. On the other hand, high resolution images occupy a considerable amount of memory and causes a slow down of the application execution. For such reason, it is generally convenient to downsize the image resolution in order to reach a compromise between precision of the estimate and speed of the algorithm.
In this regard, a known computer vision technique which is used to obtain several image resolutions is the *Laplacian Pyramid*. Such method consists in saving in a structure both the low and high frequency of an image at different scales. The presence of such parallel structure lets the user recover any scale of the image limiting the loss of information.

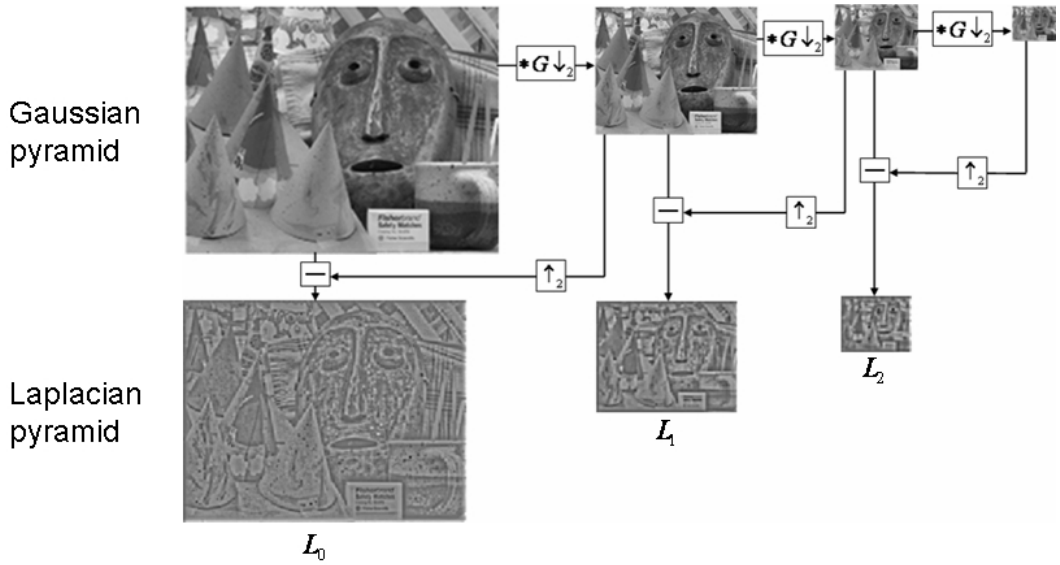


Figure 27: Representation of three levels of transition on a Laplacian Pyramid.

With reference to Figure 27, the original image is filtered with a Gaussian filter (i.e. a low-pass image filter). The difference between the original image and its low-frequency version returns the high-frequency image. Once such procedure has been followed, it is possible to downsample the low-frequency picture such that its linear dimension is halved. In essence, the downsampling consists in deleting alternatively rows and columns, then a gaussian filter is applied again (blurring) and it is possible to proceed towards lower levels of the pyramidal decomposition. At the end of the levels definition, it is possible to go back and forth through different image resolutions, since the backward image reconstruction is guaranteed by the presence of the high-frequency version. Such technique is useful for making research at different scales without loosing information of the original image. For instance, detecting a features at several resolution makes it scale-invariant, thus more reliable.
As far as visual odometry is regarded, the creation of a pyramid would be a useless exploitation of volatile memory, since once the best scale has been chosen there is no need to keep the image spectra. In this regard, only the *Gaussian Pyramid* is taken into account, namely the top side of Figure 27.
The resolution definition is directly solved by the Matlab function *imresize.m* which requires as arguments both the target image and a numerical factor which is then used to downsample the linear dimension.
The best resolution compromise can be defined only with several trials once the body of the VO application is defined.

Image mask Since the camera in mounted on top of the vehicle, not the full nominal field of view can be exploited to track features in a feature-based VO approach. For instance, the centre of the camera shows the vehicle external frame which is fixed with respect to the camera itself. This situation would cause several problem of robustness when dealing with RANSAC for the estimate of the homography. At first, the larger the number of outliers, the lower the quality of the estimate. Moreover, a consistent number of invalid features brings about an increase in the number of iterations, thus a longer execution time.
If one gets used to think at images as matrices, it is straightforward to understand that a mask filter can be designed simply as a binary matrix with the same dimension of the target picture and which contains 0 in the region to be masked and 1 in the other entries. This way, the application of the filter can be resumed in a fast element-by-element matrix multiplication. For clarity, a numerical example is provided:

$$imfilt\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} a & b & c & d & e \\ f & g & h & i & l \\ m & n & o & p & q \\ r & s & t & u & v \end{bmatrix} \right) = \begin{bmatrix} a & b & c & d & e \\ f & 0 & 0 & 0 & l \\ m & 0 & 0 & 0 & q \\ r & s & t & u & v \end{bmatrix}$$

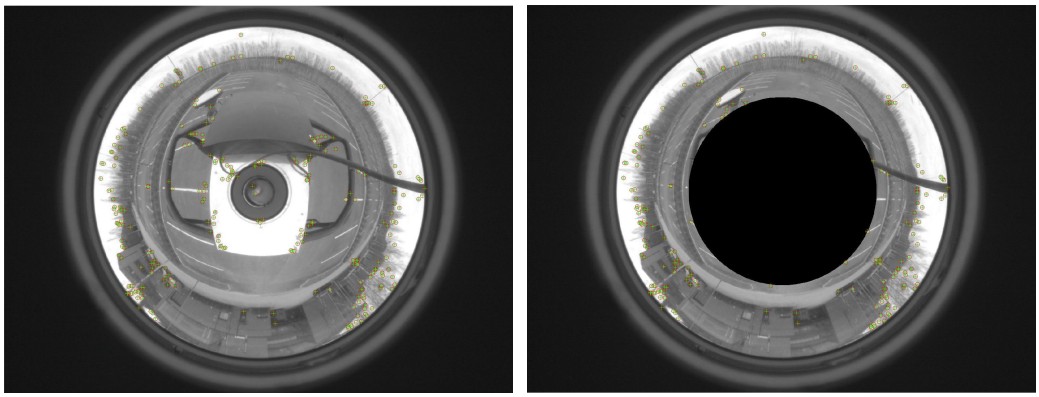


Figure 28: Mask filter applied to the original image. This way no features are detected around the region limited by a minimum radius R_{min}.

The presented functionality has been implemented in a Matlab function named after *PicMask.m*. The function is addressed to the creation of a circular mask in order to prevent the detector from tracking features in the forbidden region. The arguments required are:

$$\textbf{masked\_img} = \mathit{PicMask}(\textbf{img}, [x_c, y_c], R_{min}, R_{max}) \tag{48}$$

where $[x_c, y_c]$ are the centre coordinates of the mask disk, R_{min} and R_{max} are respectively the minimum and maximum radius such that the originale image is kept unvaried in a region $\mathbf{I}(r)$ such that $R_{min} < r < R_{max}$, with r computed from the indicated centre.

4 Pure Visual Odometry

In this chapter a first proposal of a pure VO algorithm is presented. The code is mainly subdivided in five parts: features extraction and matching, visual compass implementation, model estimate by RANSAC, homography decomposition and finally motion integration. Each of these steps is treated in a subsection and it is accompanied by the corresponding Matlab code.

4.1 Features extraction and matching

```
% Detection of keypoints

points1 = detectSURFFeatures(img1);
points2 = detectSURFFeatures(img2);

% Descriptors extraction - ValidPoints are the one
% computed regularly which are not on the border
% of the image.

[features1,valid_points1] = extractFeatures(img1, ...
... points1,'featureSize',128);
[features2,valid_points2] = extractFeatures(img2, ...
... points2,'featureSize',128);

% Matching descriptors

indexPairs = matchFeatures(features1,features2, ...
... 'MaxRatio',0.5);

matchedPoints1 = valid_points1(indexPairs(:,1),:);
matchedPoints2 = valid_points2(indexPairs(:,2),:);

% Visualisation

showMatchedFeatures(img1,img2,matchedPoints1, ...
... matchedPoints2)
```

The following application proposal makes use of a feature-based approach for the estimate of the frame-to-frame displacement. The search of the key points to be tracked is carried out by a *Speed Up Robust Features* (SURF) detector

[17]. A SURF detector is a fast and robust algorithm which makes use of box filters to allow the solution of both recognition and tracking real-time problems. The main steps behind a SURF detector are the feature extraction and the feature description.

Features extraction The extraction process is based on the computation of the Hessian matrix of the pixel intensities. Rather than using different measures at several scales (Hessian-Laplace detector), SURF only relies on the Hessian determinant. Such choice conveys to SURF the know execution speed.
In order to find points of interest on the image, the second derivative is applied to a sub-region of the whole image represented by a filter box of dimension $w \times w$. The appearance of the first two derivative orders along one direction are shown in Figure 29.

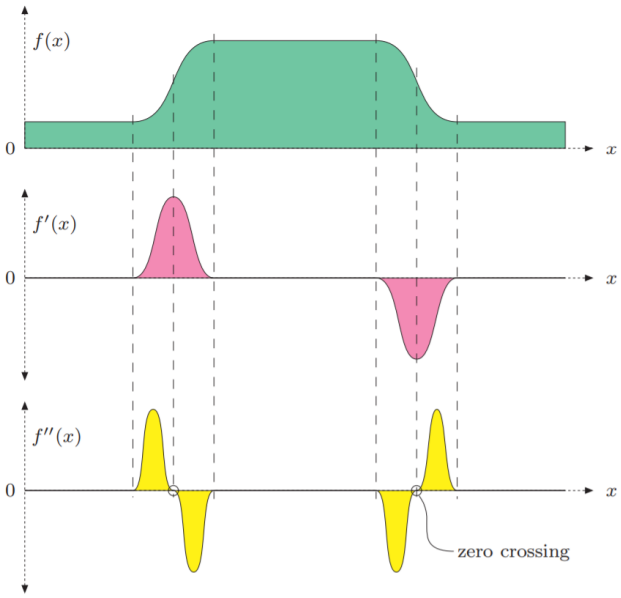


Figure 29: First and second order derivative of pixels intensity along the horizontal direction. The picture is a courtesy of [18].

However, the intensity distribution on an image is not properly as represented in Figure 29 by $f(\mathbf{x})$, but it is affected by high-frequency noise. With this in mind, the derivation would return a non informative result on the zero-crossing point of the Laplacian, thus a Gaussian filter $g(u, v)$ (low-pass image filter) is applied to the picture before the derivative operator. According to the *derivative theorem of convolution*, it is possible to save one operation and apply only a filter of the shape:

$$\frac{d^2}{dx^2}(f * g) = f * \frac{d^2 g}{dx^2} \tag{49}$$

such that the second derivative of the gaussian filter can be represented with only a kernel to be applied each time at every window $w \times w$ which scans the image domain. At the end of computation, each scanned area can be represented by the Hessian

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{yx}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{50}$$

where L_{ij} represents the second order derivative of the Gaussian along the ordered directions i and j, whereas the scalar σ defines the standard deviation of the gaussian distribution. In proximity of a point of interest, the entries of $H(\mathbf{x}, \sigma)$ assume a defined set of values which can be resumed in the Hessian determinant. When such determinant crosses a user-defined threshold value, the current pixel position is marked as a feature and its coordinates are saved in a variable (on the code, *points1* and *points2*).

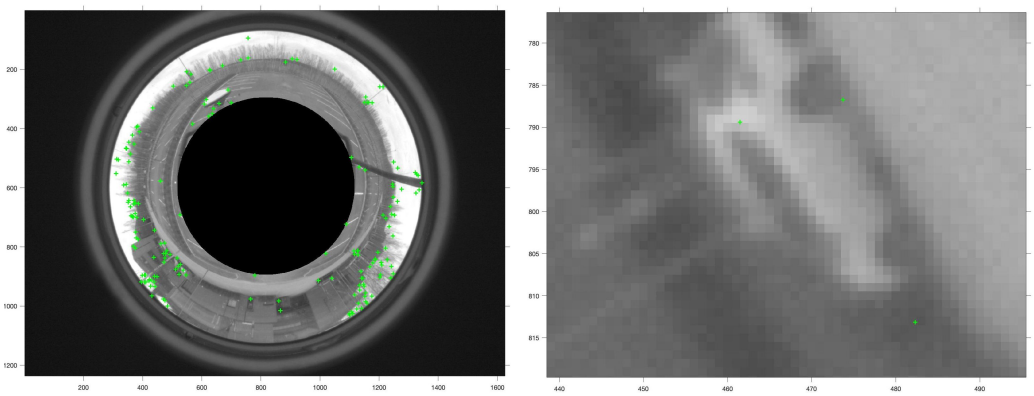


Figure 30: Features extraction process and some of the found key points in detail.

Features description Once the coordinates of the feature have been tracked, if one wants to recognise the same feature in a subsequent frame, it is necessary to build a sort of identity card of the feature itself. Such feature identity is called *descriptor* and it is based on the intensity information of its neighbourhood. The design of a descriptor takes place in two steps: the orientation assignment and the neighbourhood definition.
In order to be rotational invariant, the descriptor is built in the reference frame of a fixed orientation. Such orientation is computed by scanning a circular window around the feature with a radius of $6s$, with s the scale at which the

feature has been detected. Calculating the x and y-direction responses to the Haar-wavelet, an optimum is found. The denser the orientation discretisation, the more accurate the returned result. Conversely, the more accurate the discretisation, the bigger the descriptor dimension, thus the process slows down and requires much more memory. An example of computation of the orientation is provided in Figure 31.

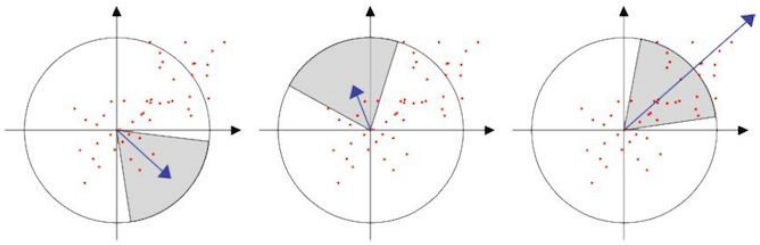


Figure 31: Orientation of the feature according to a defined neighbourhood. The picture is a courtesy of [17].

The choice of the neighbourhood consists in a rectangular region oriented according the previous criteria. The size of the window is generally of $20s$. Such region is then subdivided in a 4×4 squares grid and each single square returns the information of 5 equally spaced pixels. At this level, the Haar-wavelet response is computed again and the results are stored in a $4 \times 4 \times n$ vector, where n is the number of responses which varies according to the orientation discretisation. If only 4 directions are taken into account, a descriptor of size 64 is obtained. For higher precision, if 8 directions are considered, a descriptor of length 128 is returned. In our application, the last choice has been made.

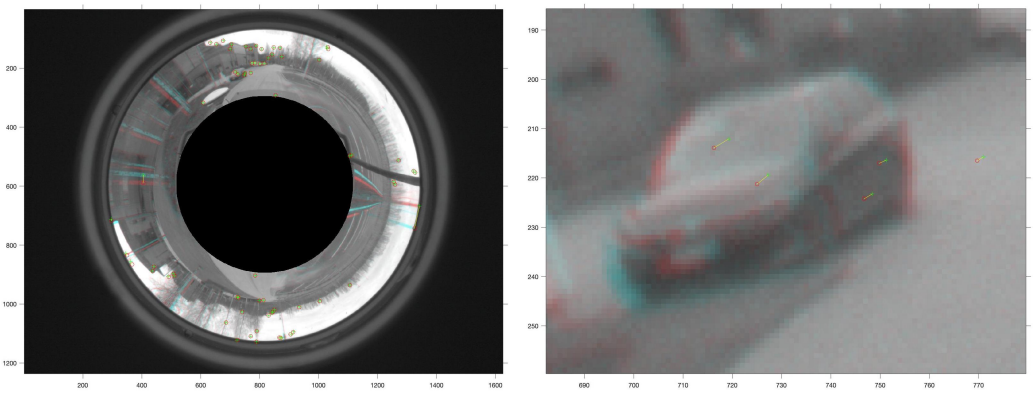


Figure 32: Features matching on two subsequent frames. The two images are overlapped with red and blue shades for distinction.

Features matching As it is clearly shown in the code at the beginning of the section, two frames are acquired from the camera and then both the processes of features extraction (*lines 3-4*) and description (*lines 10-13*) have been carried out. Then, according to a SSD (*Sum of Squared Differences*) criterion, the best feature-to-feature matching is returned (*lines 17-18*). For clarity, the example provided in Figure 32 shows the matching process between two successive frames.

4.2 Visual compass

```
1  % I select a limited FOV to estimate the angular shift
2  % from a frame to the next one. The frontal portion of
3  % the image is ideal because it is the least affected
4  % by the lens distortion. The resulting angle theta
5  % will be used to initialise RANSAC.
6
7  alfa = -10:10; % Possible shift in pixel.
8  diff = zeros(length(alfa), 1); % Euclidean distance.
9  % Define here x_low, x_up, y_low, y_up
10
11 for a = 1:length(alfa)
12
13 diff(a) = sum(sum(img1(x_low:x_up,y_low:y_up) - ...
14 ... img2(x_low:x_up,y_low+alfa(a):y_up+alfa(a))));
15
16 end
17
18 sqrdiff = sqrt(double(diff));
19 [val, ind] = min(sqrdiff);
20
21 % The heading expressed in degrees is the number of
22 % columns shifted multiplied by the angle density in
23 % pixel (since the camera is a 360deg, then the angle
24 % sensitivity is 360deg/Circumference in pixels)
25 % r_mean, C = 2*pi*r_mean, dens_linear = 360/C
26 % C must be determined here by the user according to
27 % the image resolution.
28
29 % Define C here
30 heading(ii) = heading(ii-1) + alfa(ind)*(360/C);
```

Visual compass is a direct visual odometry method to estimate the heading of

a mobile platform. The general steps of such algorithm are provided in [15], however a more personal implementation is provided above.
At first, it is asked to choose a region of the image to scan with a box. Such choice is of primary importance to obtain a reliable estimate, thus a section with a low distortion is preferred. The size of the box is another important aspect to take into account. A big window is more affected by the image distortion and also a consistent amount of pixels brings about a large generalisation of the intensity measurements, thus small variations are not detected properly. Conversely, a very small window is affected by the image noise and so some unreliable estimates are returned.
Though the best window size and ratio will be defined experimentally, in [15] it is suggested to limit the FoV as much as possible.

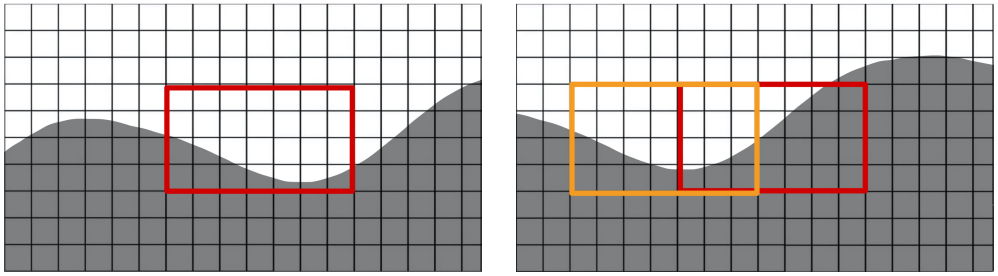


Figure 33: Simplified example of the main idea behind the visual compass. The background on the second grid is the one of the first image shifted to the left of about 4 columns.

Problem set-up At first, it is necessary to make a hypothesis on the amount of pixels which can move horizontally in the frame-to-frame period. Such choice will be made later according to the trend of the yaw rate, while at the moment a default range of $[-10, 10]$ pixels (*line 7*) is fixed.
Given a first frame (Figure 33 on the left), the window is used as an *integral image* such that the sum of all the scanned pixels intensities is computed. When the successive frame is analysed (Figure 33 on the right), the platform has already moved laterally and the scanned area returns a different integral of the intensities (red box). In order to estimate of how many pixels the image plane shifted and so which portion of the image has been scanned in the previous frame (orange box), the integral image is computed several times by shifting the red box centre from -10 to 10 pixels from the original position. When the window covers exactly the portion of the image of the previous frame, the intensity integral is supposed to be almost the same, thus it is possible to know of how many pixels the image moved. Such procedure is implemented as a problem of minimum on the integral images differences (*lines 11-19*).

Pixel-degree conversion When the pixels shift has been correctly estimated, it is necessary to find a conversion factor which links together the change in pixels to a rotation in degrees. This procedure can be done in two ways according to the kind of image available. If a panorama is employed, taking into account that the camera horizontal FoV is a complete 2π, the conversion factor is simply the total 360 *deg* divided by the horizontal length of the image in pixels.
However the choice of unwrapping the image plane in a panorama is not the best solution for eventual feature-based estimates of the motion, because the tracked features coordinates undergo to another transformation before being back-projected onto the unit sphere. With this in mind, the composition of many coordinates transformations may lead to a decrease in the accuracy, especially when the camera works at high frequencies. For the reasons mentioned so far, the proposed VO application makes use of the circular image plane directly recovered from the camera.

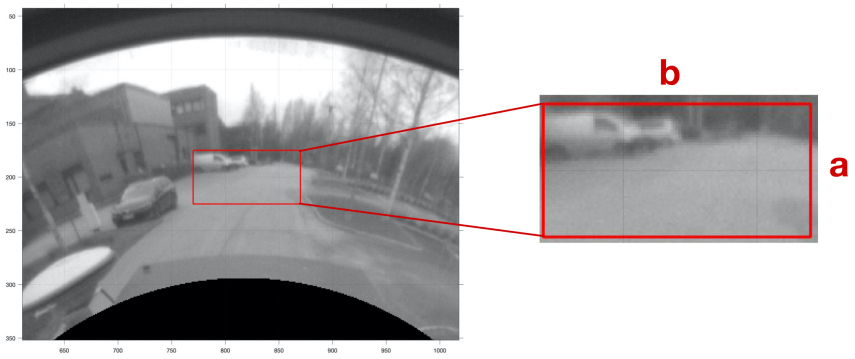


Figure 34: Visual compass box geometry and position on the image plane.

Given a box $a \times b$ with a the shorter side and centred in $C(u, v)$, it is possible to choose a wide range of radii which start from the image centre and end up inside the rectangular region. In order to compute the mean circumference passing through the box, a mean radius is chosen whose length is R_{mean}. Since the box dimension is little in comparison to the full image domain, it is reasonable to represent each pixel inside the rectangular region as if it is distant R_{mean} from the image centre. Holding such hypothesis, the circumference passing through the box centre is $C = 2\pi R_{mean}$, thus the conversion factor can me computed as $\sigma = 360/C \; deg \cdot pixel^{-1}$ (*lines 29-30*). Experimental results show that the pixels at greater radii compensate the error of the ones at lower radii, thus the hypothesis is well satisfied.

Non-idealities attenuation A consistent problem of such pixel-to-degree conversion comes from the fact that the camera centre is not properly the image

centre. At first, because of manufacturing tolerances, the perpendicularity of the mirror with respect to the camera sensor plane is not guaranteed. Secondly, the discretisation of the image in pixels does not keep a perfect circular image, thus the rotation of the pixels takes place around a point which is not actually the one which is used to measure R_{mean}. For that reason, it is reasonable to amortise such error by taking measurements on two different sides of the image diameter. Keeping unvaried the box size, ratio and the length of R_{mean} with respect to a hypothetical centre, the heading estimate is then evaluated as a mean value of the two measurements. This way, if one of the two boxes overestimates the platform steering, the opposite side is more likely to underestimate the measure. In the end, the mean value smoothes the error weight and returns a better estimate. Results are shown in Figure 35.

Effect of the image resolution When high image resolutions are used in VO computations, there are two main disadvantages: the former basically consists in the slow down of the execution speed, while the latter is that every high frequency noise at the pixel scale weights more on the estimate of the platform displacement. With this in mind, three different resolutions have been tested for the definition of the heading. Results are shown in Figure 35.

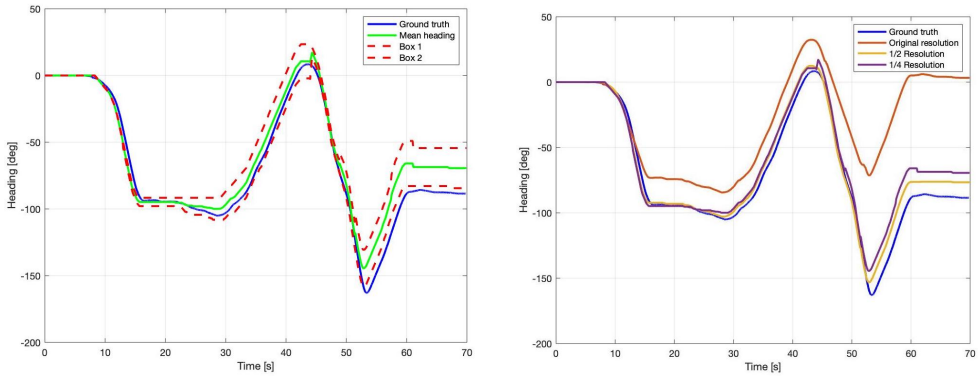


Figure 35: On the left, the effect of averaging two boxes estimates. On the right, a heading estimation at different image resolutions.

Box ratio Through several tests made on real datasets, it has been discovered that the ratio of the box dimensions a/b has somehow an influence on the accuracy of the estimate. The comparison is made with the data coming out from a filtered odometry which is supposed to the the ground truth.

Box scale Another issue to be tested regards the dimension of the box itself. At the moment, it has been realised that the 1/2 resolution and a box ratio of 0.7 are the best choices for an accurate estimate.

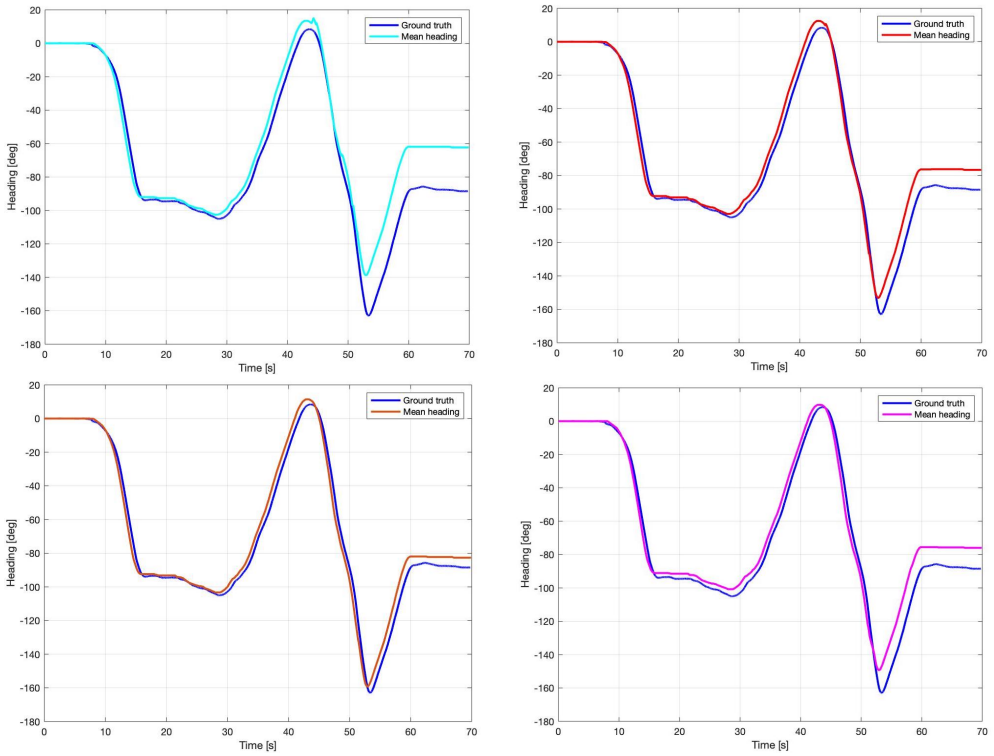


Figure 36: Effect of the box ratio on the heading estimate. The tested ratios a/b are 0.3 (top, left), 0.5 (top, left), 0.7 (bottom, left) and 1 (bottom, right).

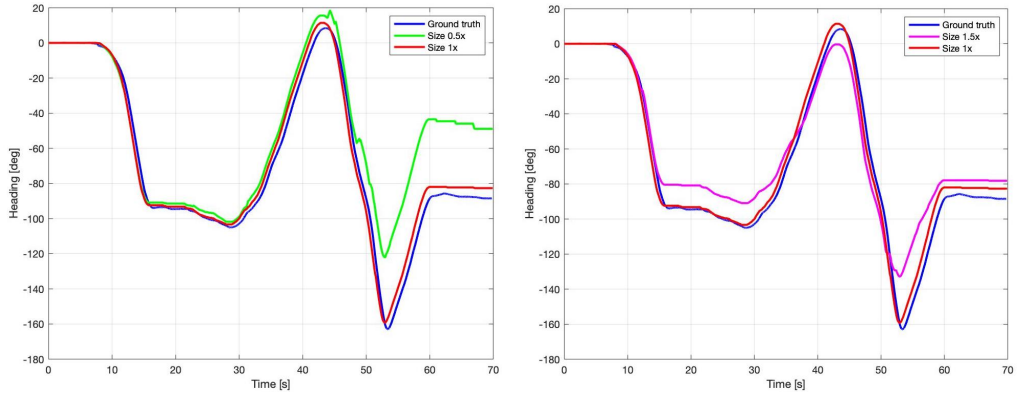


Figure 37: Effect of the box scale on the heading estimate. The tested scales are 0.5× (on the left) and 1.5× (on the right) the original size.

Keeping fixed those two parameters, some tests are made by changing the box scale, in order to clarify which is the best empirical relationship between the image and the window size.
As it is clarified in Figure 37, the original size of the box, that is 200×140 pixels, is the best choice for the heading estimate. Neither a shrinking nor a magnification of the box dimension returns any improvement in accuracy.
Moreover, the usage of a box 150% larger than the initial size highlights the smoothing effect of direct methods based on a considerable number of pixels. When the integral image is computed on a big window, the differences between successive scannings is coarse and not so effective.

Columns shift domain In accordance with *line 11* of the piece of code, it is supposed that the maximum columns shift in a frame-to-frame period is 10 *pixels* in modulus. On one hand, checking on a large shift domain increases the possibility to track a turning at high yawing rates. On the other hand, if the platform is either not able to reach such rates or if these speeds do not suit its mission profile, the computation in the full $[-10, 10]$ range brings about a useless slow down in the execution and also increases the possibility of tracking a window whose integral is similar to the current one, but which does not represent the actual steering.

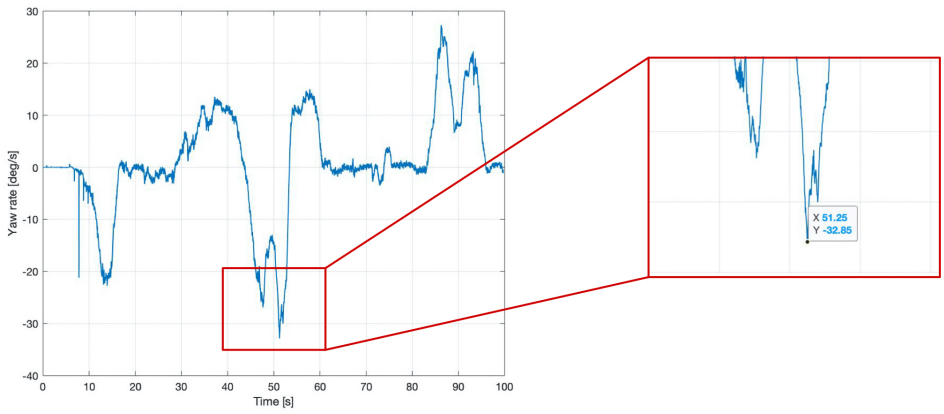


Figure 38: Yaw rate trend on a real dataset based on the Polaris platform.

In order to choose the right range which respects the yawing trend of the vehicle, it is necessary to make use of a dataset of the platform steering angle and speed. Once the maximum yawing rate (in modulus) has been identified, the corresponding dimensioning is made by converting the speed information in a pixels datum, according to the relationship:

$$\dot{\theta} \cdot T_{frame} = \frac{360}{C} \cdot N \tag{51}$$

where $\dot{\theta}$ is the yaw rate, T_{frame} is the reciprocal of the camera working frequency and N is the corresponding number of pixels shifted laterally in the scanning box. After analysing the chunk of data which contains the highest yaw rate, it is possible to fix the maximum turning speed modulus at $32.85\frac{deg}{s}$, thus it is possible to compute the maximum detectable shift in pixels. Given a working frequency of 10 Hz and a resolution of 0.5×, N can be computed like:

$$N = \frac{C_{mean}}{360} \cdot \dot{\theta}_{max} \cdot \frac{1}{f_{camera}} \simeq 13 \; pixels \tag{52}$$

which means that it is necessary to increase the shift range from $[-10, 10]$ to $[-13, 13]$ pixels in order to guarantee that each $\Delta\theta$ between two successive frames can be properly detected by the visual compass.

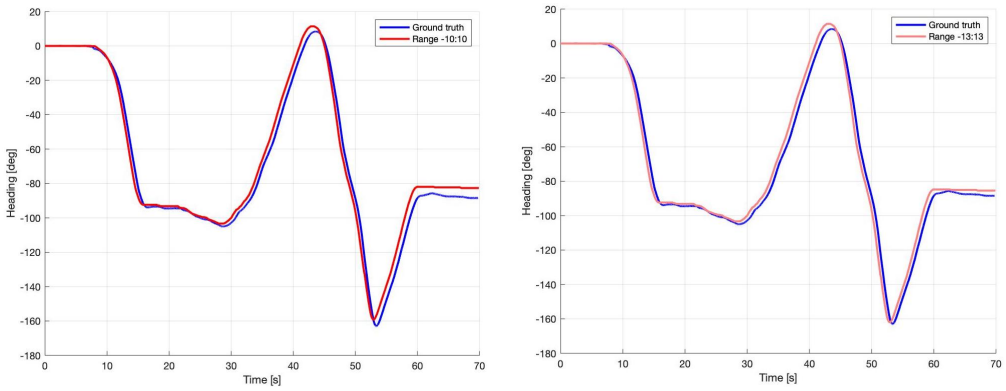


Figure 39: An improvement in the accuracy of the estimate is obtained when the range of check is increased from $[-10, 10]$ (on the left) to $[-13, 13]$ (on the right).

4.3 Random Sample Consensus

```
% RANSAC Algorithm for estimating H

A=double([matchedPoints1.Location, ...
 matchedPoints2.Location]); %[(x1 y1)(x2 y2)]

N=100; Ncount=0; % Number of interations
    initialisation.
e=0.2; % Probability of random-(x,y) to be an outlier.
pb=0.99; % Probability of all attemps not to contain
    outliers.
```

```
9  s=8; % Minimal seeds set to initialise the model
10 th=0.7*length(A); % consensus threshold
11 max_cons=0; % Initialisation of the highest consensus
       set
12
13 % Initialisation of the homography model by mean of
14 % the yaw angle from the visual compass
15
16 psi=alfa(ind)*(360/C)*(pi/180);
17 H_in=[cos(psi) -sin(psi) 0;
18        sin(psi)  cos(psi) 0
19         0          0       1];
20
21 % We use such H to determine a consensus set S1
22 % which is a set of points whose reprojection error
23 % is lower than a threshold d.
24
25 while N>N_count
26
27     err=zeros(1,length(A)); % Initialisation
28
29     for h=1:length(A)
30
31         err(h)=norm([A(h,3:4) 1]'-H_in*[A(h,1:2) 1]')^2
               ...
32             +norm([A(h,1:2) 1]'-H_in\[A(h,3:4) 1]')^2;
33
34     end
35
36     d=mad(err); % Threshold computed as MAD
37     S1=[]; % Initialisation of the consensus set (void
           )
38
39     for h=1:length(err)
40
41         if err(h)<=d
42
43             S1=[S1; A(h,:)];
44
45         end
46
47     end
```

```
48
49      % At this point we have obtained the first
50      % consensus set S1. It is needed to check if
51      % the number of points in S1 is larger than a
52      % threshold t. If so, a new H* is computed from
53      % S1, otherwise we start back from line 94.
54      % The process is over after a given number of
55      % iteration is completed.
56
57
58      if length(S1)>=th && length(S1)>max_cons
59
60          P=[zeros(length(S1),3) -S1(:,1) -S1(:,2) ...
61              -1*(1+zeros(length(S1),1)) S1(:,4).*S1(:,1)
                    ...
62              S1(:,4).*S1(:,2) S1(:,1)  S1(:,2) ...
63              1+zeros(length(S1),1) zeros(length(S1),3)
                    ...
64              -S1(:,3).*S1(:,1) -S1(:,3).*S1(:,2)];
65
66          T=[-S1(:,2); S1(:,3)];
67
68          Hcoeff=(P'*P)\(P'*T);
69          H=[Hcoeff(1) Hcoeff(2) Hcoeff(3);...
70              Hcoeff(4) Hcoeff(5) Hcoeff(6);...
71              Hcoeff(7) Hcoeff(8) 1];
72
73          max_cons=length(S1);
74          e=1-(max_cons/length(A));
75          N=log(1-pb)/log(1-(1-e)^s);
76
77      end
78
79      Ncount=Ncount+1;
80
81  end
```

The Random Sample Consensus (RANSAC) is employed to find a robust homography model. Generally, the model hypothesis is initialised with the minimal seeds number by a random draw from a wider dataset **A**. In [15] it has been demonstrated that if the initialisation is made with the yaw angle already computed with the visual compass, the overall performance speeds up.

With this in mind, the initial homography matrix is assumed to be a simple Euclidean transformation with no translation:

$$\mathbf{H_{in}} = \begin{bmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 \\ \sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{53}$$

where ψ_i is the yaw angle detected after a comparison between the frame i and the frame $i-1$. Once a first model has been initialised, it is required to determine the first consensus set **S1**, namely the set of all features matches whose frame-to-frame transformation can be approximated by mean of $\mathbf{H}_{in}$ within an error threshold d. The error function is the same as [15] and it is based on the sum of the differences between a feature and its projection on the other frame by mean of the instantiated homography $\mathbf{H}_{in}$. When a new best consensus set if found (*lines 62-81*), the homography is estimated in the LS sense using only the points correspondences in **S1**. Finally, the maximum number of iterations and the statistical hypothesis on the full dataset are updated (*lines 77-79*).

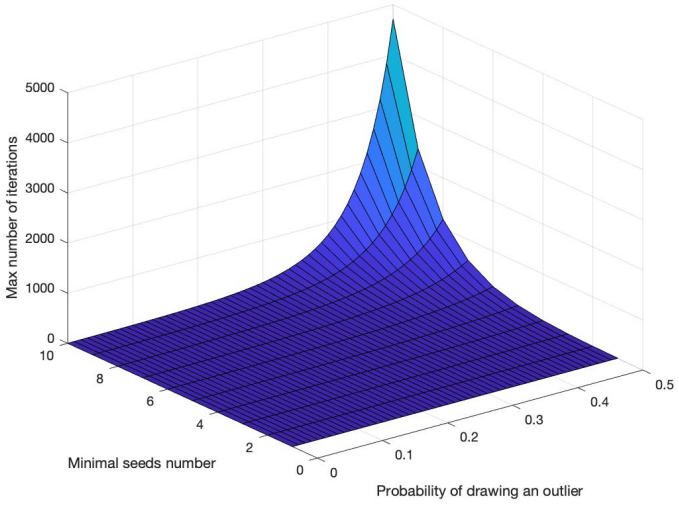


Figure 40: Trend of the maximum number of iterations as a function of the outlier probability and the minimal seeds to parametrise the model hypothesis.

A problem of RANSAC is the number of iterations required to reach an accurate model estimate. Such number grows with the minimal seeds and in case of an homography such trend has a considerable weight on the execution time of the application. The RANSAC algorithm terminates when the computed number of maximum iterations N reaches the number of actual concluded iteration N_count (*lines 25*).

4.4 Homography decomposition

```
%% Euclidean hypotesis to extract the translation
% Since the transformation is up to a scale factor,
% what we actually have found is Hl=k*H. In order
% to find H, we find sigma2(Hl), and k is the
% second-largest singular value.

S = svd(H);
H = H./S(2); % Recovery of the depth factor ratio

% Since the shift from one frame to the next is
% so small, the homography can be approximated to
% an almost euclidean 3DOF transformation. From
% this assumption on, it is possible to decomposed
% H to find [t] and [theta]. Theta will be used as
% confirmation for the visual compass.

Q = H(1:2,1:2); % Rotation sub-matrix from H.
[U,S,V] = svd(Q); % Singular Value Decomposition of Q.
Rd = U*V'; % Orthonormal matrix closest to Q.

height = 2; %[m] - Camera/Ground distance
t = H(1:2,3).*height; % Translation [tx, ty]'
```

At the end of the RANSAC while-cycle, a robust estimate of the homography has been obtained. Following the model proposed by [15], the hypothesis of ground planarity is adopted, so that it is possible to compute the real world kinematic variables only by mean of the camera height from the ground (the full mathematical derivation of the model is provided in Chapter 2.3 , Homography-based ground plane navigation). According to the unified camera model in unitary sphere coordinates, each point is defined by a vector up to a scale factor which is a depth-parameter λ. For such reason, when the homography matrix $\mathbf{H}_L$ computed on the image points is employed to describe the world kinematics, it is necessary to recover the normalisation factor such that

$$\mathbf{H} = \frac{\mathbf{H}_L}{\lambda} = \frac{\mathbf{H}_L}{\sigma_2(\mathbf{H}_L)} \tag{54}$$

where $\sigma_2(\mathbf{H}_L) \in \mathbb{R}$ is the second largest singular value of the matrix $\mathbf{H}_L$. Such scale recovery procedure is proved and shown in [19].

Euclidean hypothesis When the majority of the tracked features are enclosed in a portion of the whole camera FoV, it is possible to think at the $\mathbf{H}$ matrix as an Euclidean transformation of a plane (i.e. simple rotation and translation of the ground). With this in mind, the estimated matrix must assume the shape:

$$\mathbf{H} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & -t_x/h \\ \sin(\psi) & \cos(\psi) & -t_y/h \\ 0 & 0 & 1 \end{bmatrix} \tag{55}$$

where h is the vertical distance between the camera and the world reference frame. In this regard, given h the recovery of the translation vector is straightforward.

$$\mathbf{T} = \begin{bmatrix} T_x \\ T_y \end{bmatrix} = h \cdot \begin{bmatrix} H_{13} \\ H_{23} \end{bmatrix} \tag{56}$$

As far as the rotation is concerned, a sub-matrix $\mathbf{Q}_{11\to22}$ must be taken into account. Actually, because of the noise induced by the outliers, the matrix $\mathbf{Q}$ is unlikely to be orthonormal. In order to deal with such problem, the linear algebra technique of SVD is employed to find the closest rotation matrix $\mathbf{R}$ constrained to the orthonormality condition $\mathbf{R}\mathbf{R}^T = \mathbf{I}$.

$$\mathbf{Q} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}} = \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{v_1}^T \\ \mathbf{v_2}^T \end{bmatrix} \tag{57}$$

The Singular Value Decomposition returns three matrices. The matrix $\mathbf{\Sigma}$ is a diagonal matrix which contains the eigenvalues of $\mathbf{Q}^T\mathbf{Q}$, whereas $\mathbf{U}$ and $\mathbf{V}$ are the eigenvectors respectively of the matrices $\mathbf{Q}\mathbf{Q}^T$ and $\mathbf{Q}^T\mathbf{Q}$. The orthonormal rotation matrix $\mathbf{R}$ closest to $\mathbf{Q}$ can be determined as

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T = \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} \end{bmatrix} \begin{bmatrix} \mathbf{v_1}^T \\ \mathbf{v_2}^T \end{bmatrix} \simeq \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{58}$$

Homography hypothesis When the features employed in RANSAC for estimating $\mathbf{H}$ are sparse on the full camera FoV, the Euclidean hypothesis is not accurate enough for computing back the world kinematic variables. In such cases, it is necessary to keep a full 8DOF model of the shape:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = h_{33} \cdot \begin{bmatrix} h_{11}/h_{33} & h_{12}/h_{33} & h_{13}/h_{33} \\ h_{21}/h_{33} & h_{22}/h_{33} & h_{23}/h_{33} \\ h_{31}/h_{33} & h_{32}/h_{33} & 1 \end{bmatrix} \tag{59}$$

The homography is the most general transformation which can be obtained and it is characterised by 9 entries ($h_{11} \rightarrow h_{33}$). However, in such linear application only the relationship between each h_{ij} element is important, thus after the matrix has been scaled by the last entry h_{33}, only 8 unknowns are left to be determined. The decomposition of the full homography can be done with the Triggs algorithm [20]. The decomposition proposed by Bill Triggs is fundamentally based on the SVD, but it returns a more robust and stable solution in case of a homography. The possible configurations which are obtained from **H** with such algorithm are four. Two of the four solutions can be easily discarded by some geometrical considerations, however the ambiguity cannot be completely solved for the remaining two configurations. In the present case, the modelling of the ground plane can be used as a final discriminant to choose the correct solution.

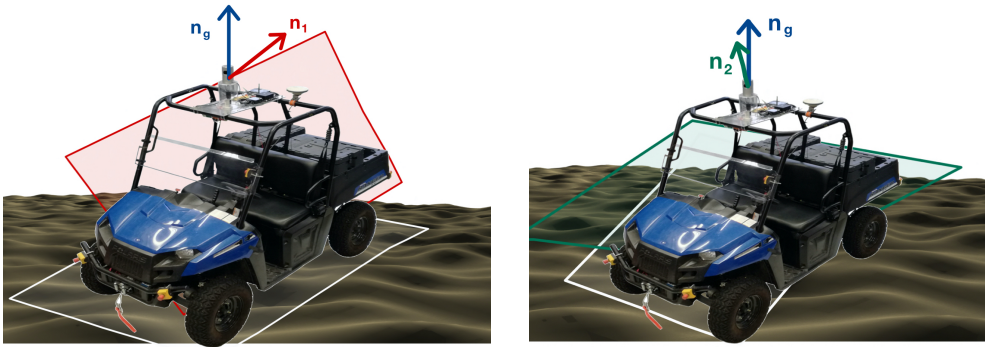


Figure 41: Two plausible solutions of the Triggs decomposition algorithm.

```
%% Triggs decomposition - Homography Hypotesis
% With reference to the triggs() function, the
% given H is decomposed.

[R1, t1, n1, R2, t2, n2, zeta] = triggs(H);
distance1 = norm(n1-[0; 0; 1]);
distance2 = norm(n2-[0; 0; 1]);

if distance1 > distance2

    t = t2(1:2);

else

```

```
15        t = t1(1:2);
16
17  end
```

Given the homography matrix $\mathbf{H}$, the Triggs algorithm returns the couple of solutions $(\mathbf{R}_1,\ \mathbf{t}_1,\ \mathbf{n}_1)$ and $(\mathbf{R}_2,\ \mathbf{t}_2,\ \mathbf{n}_2)$ (*line 5*). In order to choose the most appropriate configuration, we take into account the axes of rotation, namely $\mathbf{n}_1$ and $\mathbf{n}_2$. According to the reference frame shown in Figure 41, the ground plane is described by the vector $\mathbf{n}_g = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. The solution to be chosen is the one whose plane normal $\mathbf{n}_i$ is closer to the ground normal (*lines 6-17*).
Because of the considerable image distortion caused by the omnidirectional camera, in both the euclidean and the homography hypotheses the heading estimate is put aside, since best results have been provided by the visual compass.

4.5 Motion integration

```
1  %% Integration of the motion
2
3  T = norm(t)*height; % 2D vector of the displacement.
4
5  x(ii) = x(ii-1) + T*cos(heading(ii)*(pi/180)); %[m]
6  y(ii) = y(ii-1) + T*sin(heading(ii)*(pi/180)); %[m]
```

The kinematic model adopted for the camera is the simplest possible. Each coordinate is updated singularly by projecting the displacement vector magnitude along the actual heading, previously computed by mean of the visual compass. A first problem with such kinematic model from [15] can be found in the way the displacement is defined (*line 3*). In essence, the sign of T is always positive and such positive value increases the vehicle position (x, y) always in the direction of the heading. As a consequence, it is not possible to track eventual reverse gear manoeuvres.
A possible solution consists in multiplying the motion increments of both x and y by a flag which assumes negative values when the speed changes in sign. However, no external information about speed is available in case of pure VO approaches, thus such limit is kept in the algorithm.

Motion enhancement Another important issue concerning the motion estimate regards the background noise of the image when the vehicle is not moving. Even in absence of motion, pixels do not stay fixed on subsequent frames because of little vibrations, external brightness changes etc. This way, in a

frame-to-frame evaluation of $\mathbf{T}$, it is possible that the noise integration causes an apparent motion. Such problem arises if the subject noise is not a zero-mean noise. The non-zero-mean hypothesis has been confirmed experimentally.
In order to avoid the apparent motion triggered by the image noise, it is possible to make a statistical study on the frame-to-frame displacements of the features for some random frames taken from the dataset.

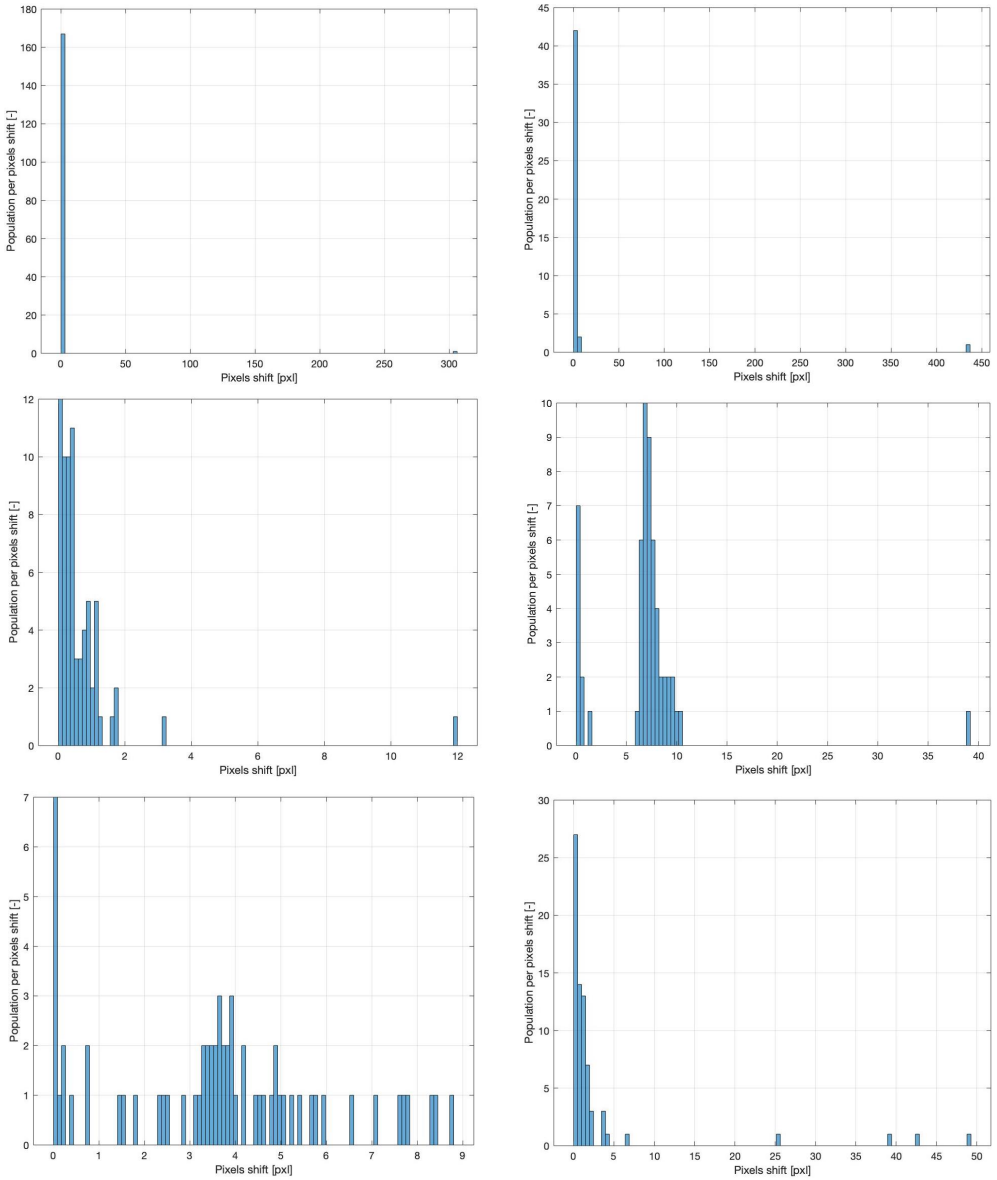


Figure 42: The first row shows a statistics of the features displacement in absence of motion. The other pictures represent the features movements in a motion condition.

Once a trend has been identified, it is possible to instantiate the motion only if a given percentage of features in the image shifted more than a threshold value. From the analysis of the histograms it is not easy to understand how to set a threshold for enabling the motion integration. The main issue regards the presence of outliers (fake matchings) which rescale the histogram distribution such that every value around zero is not detected because of a decrease in the columns resolution.
In order to deal with such an issue, it is necessary to choose a more robust statistics. On one hand, the usage of the mean value would not return a reliable estimate because of the presence of outliers with a huge value of displacement. Conversely, the median can be employed to obtain a sort of trend of the features displacement in several motion conditions. This time, instead of sampling the dataset frames it is required to use the whole sequence in order to get a continuous line which represent the median of the shift as a function of the number of frame.
The beginning of the real motion must be detected directly by the user watching the camera frames or even by the analysis of the vehicle speed trend (other sensors must be involved). With this in mind, it has been established that the Polaris platform starts the driving nearby the frame #60. The next step consists in checking the median value corresponding to the 60th frame and formulate a condition to enable the motion integration.

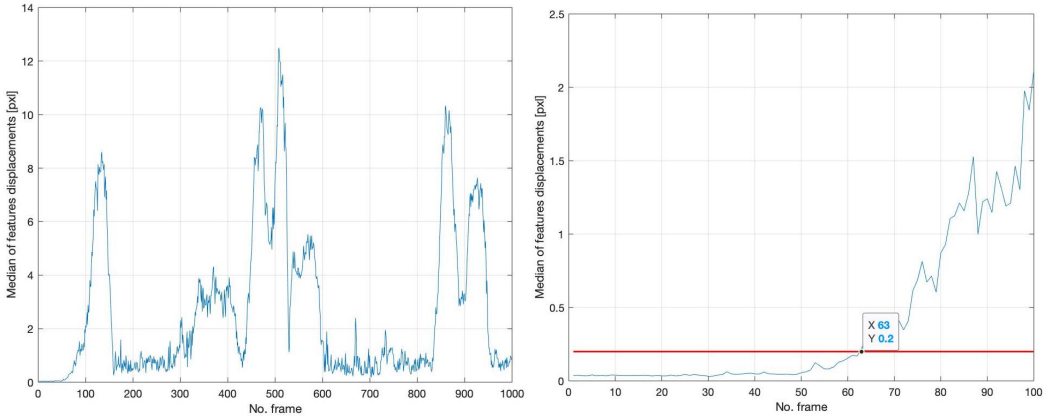


Figure 43: On the left, the median trend over 1000 frames. On the right, a detail of the threshold for motion beginning.

From the median trend analysis it is now clear that the motion can be enabled when the median of the features displacements assumes a value of about 0.2. Supposing that even in presence of a real motion not every feature undergoes to a displacement above the threshold limit, it is reasonable to establish that at least the 70% of the whole features set must be over the threshold.

```
1  %% Motion condition
2
3  pixel_diff = sqrt((pts1(1,:)-pts2(1,:)).^2+...
4      (pts1(2,:)-pts2(2,:)).^2);
5
6  motion_flag = sum(pixel_diff>0.2)/length(pixel_diff);
7
8  if motion_flag >= 0.7
9
10     % [Visual Compass]
11
12     % [RANSAC]
13
14     % [Homography decomposition]
15
16     % [Motion integration]
17
18 else
19
20     % The direction is kept unvaried with respect to
21     % the one computed on the previous frames.
22     heading(ii) = heading(ii-1);
23
24     % No translation occurs.
25     T = 0;
26
27 end
```

In order to save the time needed for the computation of all the kinematic variables (Visual Compass, RANSAC, Homography Decomposition and Motion Integration), it is a good choice to check the motion flag (*line 8*) suddenly after the features coordinates extraction.

4.6 Results

After all the previous steps have been accomplished, it is possible to visualise the path reconstruction coming out from the pure VO approach. From Figure 44 it is suddenly clear that the absolute scale of the path has not been recovered properly. Moreover, some issues about the relative scale are also detectable, since neither the shape of the the path has been properly predicted. After a careful evaluation of the results, it is possible to address such failure to the camera model.

Failure factors There are two main sources of error which influence the final result: the former is the adopted VO technique which affects the relative scale, while the latter is the chosen omnidirectional camera set-up which brings about a wrong recovery of the absolute scale.

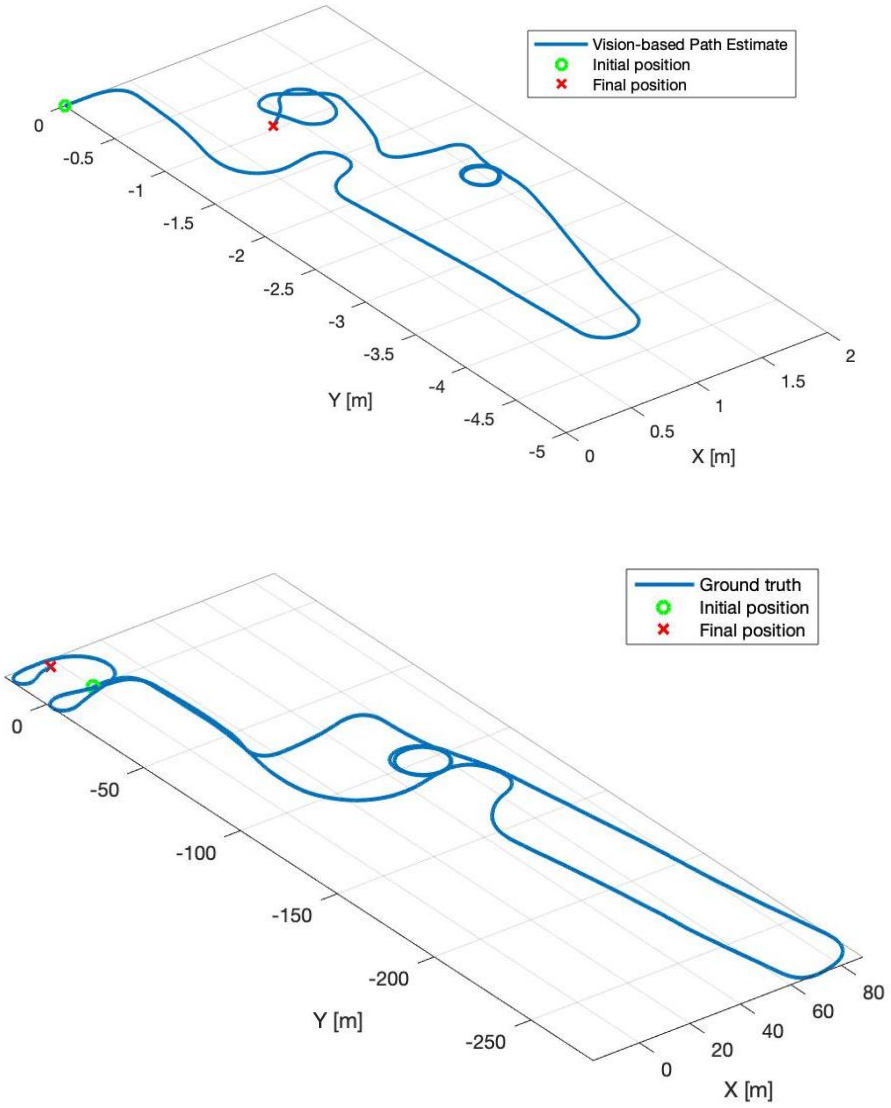


Figure 44: Comparison between the VO estimate and the GPS estimate, which is assumed to be the ground truth.

After a careful analysis of the features extraction process, it is remarkable to highlight that the most of the key points are tracked on the background (i.e. trees, buildings, other vehicles) and only a very small subset regards only the ground plane.

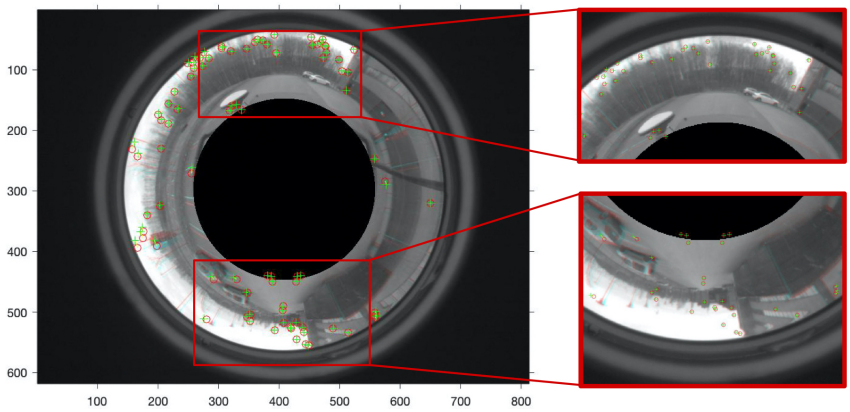


Figure 45: The great part of the tracked features are outliers, thus the RANSAC algorithm fails in estimating a good homography matrix based on the ground planarity.

Such issue affects the hypothesis of *planarity* of the ground and so it causes a wrong perception of the plane movements during the homography estimate in the RANSAC process. Indeed, though a RANSAC approach is robust to eventual outliers, it is anyway required that a good part of the key points are valid enough to define a model. A possible solution to the presented problem could be the change of the camera FoV, such that a greater portion of the image plane faces in the ground direction. To do that, a part of the Polaris equipment should be moved and the overall centre of mass should be re-defined.

Another important issue regards the lack of the absolute scale recovery. Such problem is addressed to the omnidirectional camera set-up and it can be easily verified by mean of the Ocam Calib Toolbox by D. Scaramuzza [13]. Instead of making use of the custom-made *AffineTrans.m* Matlab function, it is possible to obtain such estimate directly from the toolbox itself. The refinement of the calibration parameters can be obtained after the functionality "*Find centre*" has completed its iterations. However, it seems that the number of iterations needed to find the camera centre tends to a very high value and the estimate practically never converges.

Recalling the theory on omnidirectional catadioptric camera in Chapter 2, there are two main categories according to the camera-mirror coupling:

1 - *Non-degenerate configurations* are obtained when an orthographic camera is coupled with a parabolic mirror or when a perspective camera is couple with hyperboloidal, ellipsoidal, planar mirrors. Such assembly can be modelled with the single effective viewpoint assumption.

2 - *Degenerate configurations* are obtained when a perspective camera is coupled with either a spherical or a conical mirror. Such set-up does not presents a single viewpoint.

The actual set-up of the Polaris platform offers a perspective camera coupled with a parabolic mirror, thus the mathematical model employed for describing the projection and re-projection of the features cannot be considered valid. This way, the Taylor polynomial defined in the calibration phase is not correct, thus the key points coordinates $\mathbf{u}'$ are incorrectly normalised onto the unit vector $\mathbf{q}''$. Such error propagation is the same for every image point, thus it affects the final result in the same way. All in all, it results to an improper absolute scale estimate.

For all the reasons mentioned so far, the actual Polaris equipment does not allow a pure VO method in order to deal with the self-localisation problem. In order to cope with such problem, it is possible to slightly modify the application in order to avoid the trust on a feature matching procedure.

5 Hybrid Visual Odometry

In this chapter some modifications to the application developed so far are made. In the previous chapter, it has been realised that the current Polaris equipment is not fully prepared for a pure VO application. In the following pages a Hybrid Visual Odometry is presented in order to accomplish to the platform self-localisation problem.

5.1 HVO framework

A wrong camera mathematical model denies the feature-based procedure which has been employed to estimate the translation vector **T** (features extraction, RANSAC estimate, homography decomposition). Conversely, the estimate of the vehicle heading has been assigned to the visual compass, which is a direct visual approach without any dependence on the camera model. With this in mind, it is possible to decouple the direction and the translation of the Polaris platform. A sketch of the solution is proposed in Figure 46.

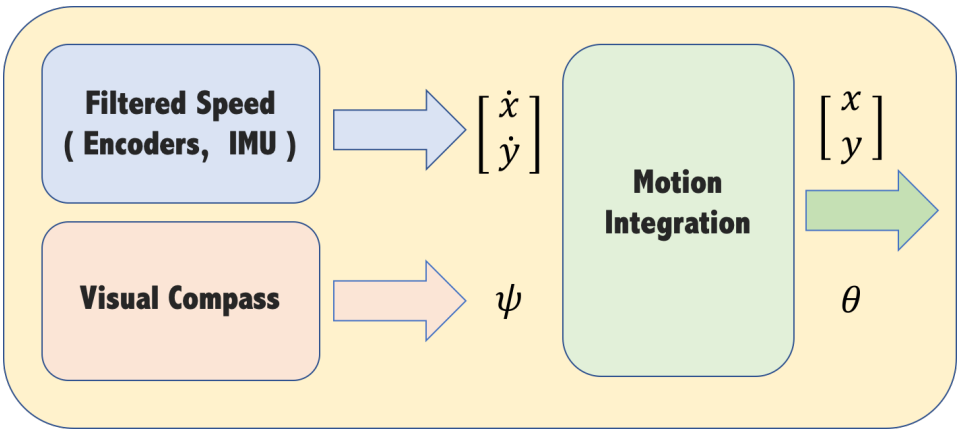


Figure 46: Framework of the proposed Hybrid Visual Odometry application.

The visual compass works at a frequency of 10 Hz since it is subjected to the camera frame rate, whereas the speed estimate which comes from the filtered odometry has a smaller integration step. As a consequence, the first issue to be solved is the synchronisation of the two sources of data in order to obtain one speed information and one yaw angle at each machine cycle. Since no HW/SW Integration will be treated in the following research work, the synchronisation problem is momentarily crossed with a linear interpolation of the speed data, in order to establish a one-to-one relationship with the information coming from the visual compass.

5.2 Linear speed estimation (EKF)

The real model of an autonomous mobile robot is actually a non-linear model. When the non-linearities are not predominant on the platform dynamics, it is always preferable to linearise the model and to employ a conventional Kalman Filter (KF) for estimation purposes.
Vice versa, some non-linearities cannot be ignored and so two approaches can be adopted in order to accomplish to estimation tasks: an Extended Kalman Filter (EKF) or a Particle Filter (PF). The former is the most employed approach in Robotics because of its computational efficiency. Conversely, the PF is preferred when dealing with non-Gaussian highly non-linear systems.

In the following application the linear speed information is obtained by EKF, exploiting the other proprioceptive sensors on the Polaris platform (e.g. IMU, Encoders). The EKF framework can be described with three main steps: state representation, design of a measurement model and update step. The ROS implementation of such procedure has been already presented in [7]. Graphical results are provided in Figure 47.

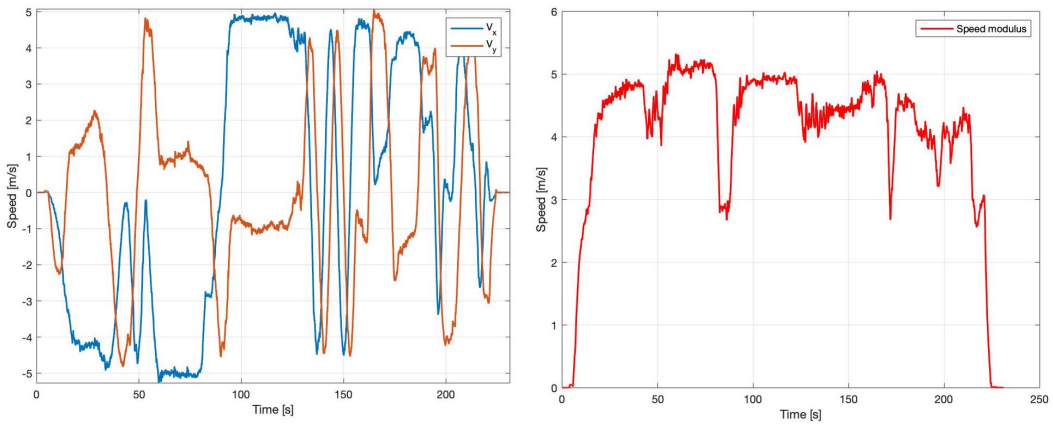


Figure 47: On the left, the trend of V_x and V_y in the world reference frame. On the right, the speed modulus $\|\mathbf{V}\|$ throughout the whole dataset.

5.3 Motion integration

The motion integration model already shown in Subsection 4.5 needs to be changed a little when passing from a pure VO to a HVO approach. At first, the translation vector $\mathbf{T}$, thus its norm, is no longer available since it was computed by mean of a feature-based method. However, the knowledge of the linear velocity allows to estimate the frame-to-frame displacement assuming constant speed in the inter-frame period. This way, the time spent from one

frame to the next one (reciprocal of the camera frame rate) defines the length of the model integration interval.

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \frac{\|\mathbf{V}\|}{f_{camera}} \cdot cos\theta \tag{60}$$

$$\mathbf{y}(k+1) = \mathbf{y}(k) + \frac{\|\mathbf{V}\|}{f_{camera}} \cdot sin\theta \tag{61}$$

5.4 Results

The hypothesis of constant speed in the $1/f_{camera}$ interval is stronger with higher camera working frequencies. In the present case, the weight of the Polaris vehicle does not allow a dynamics faster than 10Hz, so the employed frame period of 0.1s is more than sufficient to properly describe the platform kinematics. The Model-In-the-Loop results of the designed application are provided in Figure 48.

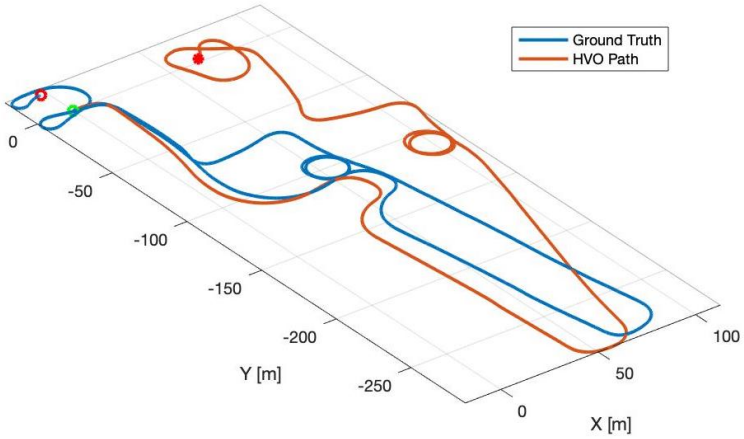


Figure 48: Path estimate with 0.5× image resolution and a visual compass window scanning ±13 pixels laterally.

From the results, it is suddenly evident that the algorithm returns a good estimate only in the first hundred of meters, then it completely degenerates in proximity of the U-shaped bend.
Recalling Chapter 4.2, it is evident that the pixels range implemented in the visual compass is still not sufficient for detecting such a steering action. With this in mind, it is suggested to proceed empirically for the correction of the visual compass parameters, i.e. the compass range and the image resolution, in order to find the combination which best fits the ground truth. With reference

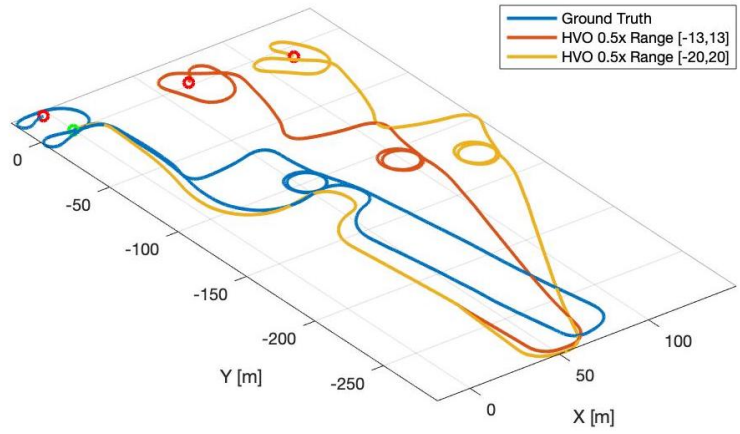


Figure 49: Path estimate with 0.5× image resolution increasing the compass range from $[-13, 13]$ to $[-20, 20]$ pixels.

to the results provided in Figure 49, it clearly comes up that the increase of the compass range brings about an impoverishment of the performance. Indeed, the analysis of a large lateral portion of pixels could cause a flatness of the integral images values, thus the search of the right shift gets even harder.
Another attempt can be done by reducing the image resolution from 0.5× to 0.25×, and check two other compass range suitable for that scale.
The ranges of $[-13, 13]$ and $[-7, 7]$ have been tested and the results are provided in Figure 50.

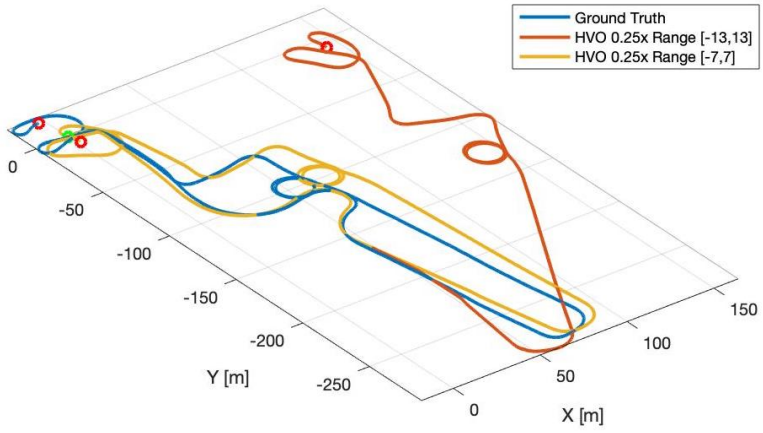


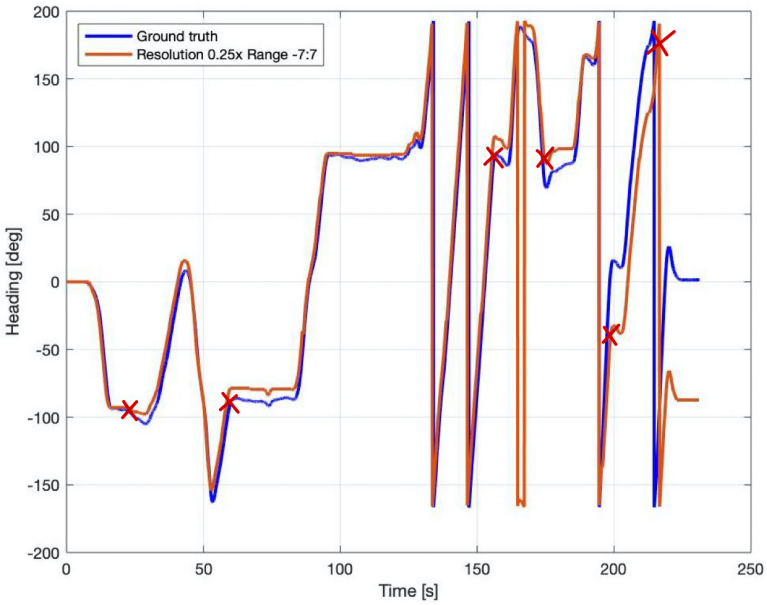
Figure 50: Path estimate with 0.25× image resolution and visual compass ranges of $[-13, 13]$ and $[-7, 7]$ pixels.

Among all the parameters variations tested, the best result is obtained when an image resolution of 0.25× is coupled with a visual compass whose range is $[-7, 7]$ pixels (yellow path). Though the obtained results represent a good estimate of the proposed ground truth, it is remarkable to analyse the heading trend in order to highlight the points at which the precision is weakened.

Resolution	**Compass** (pixels)	$\mathbf{RMS_x}$ (m)	$\mathbf{RMS_y}$ (m)	$\mathbf{RMS_\theta}$ (deg)
0.5×	$-20 < p < 20$	80.19	20.46	11.49
0.5×	$-13 < p < 13$	61.40	14.29	6.900
0.25×	$-13 < p < 13$	92.66	21.62	13.70
0.25×	$-7 < p < 7$	43.59	15.16	5.721

Table 1: Results obtained with several combinations of the parameters.

Analysis of the error Once the best setting has been chosen (0.25× Res, [-7,7] Compass range), it is recommendable to analyse the error distribution in order to detect eventual systematic defects of the designed application. The worst estimates of the platform direction occur when the vehicle is on a straight path. However from Figure 51 it is clear that the estimate is not completely wrong, but it presents an offset. With this in mind, it is clear that the initial and the final steps of a bend are not properly detected because of the low precision of the visual compass method (precision of 1 pixel in a range of $[-7, 7]$ pixels). This problem can be considered as a drawback of the resizing of the image and it is the cost to be paid in order to speed-up the application for a real-time usage.



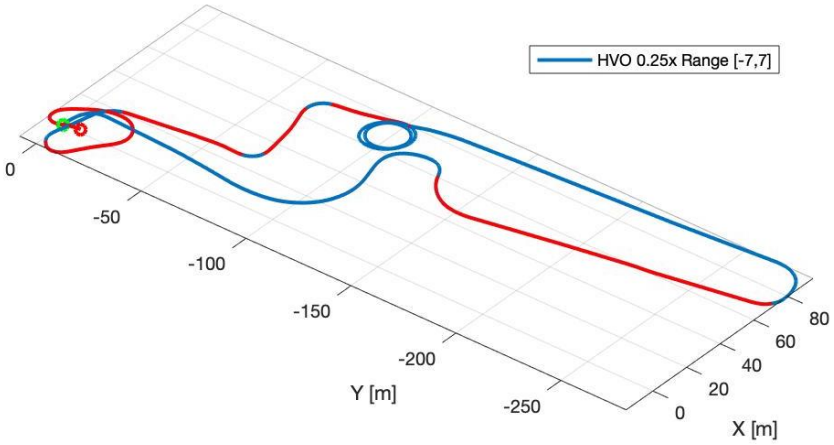


Figure 51: It is visible that the most notable shift from the ideal heading occurs after a bend. The worst heading estimates are highligthed in red.

In order to evaluate the path estimate directly on the real track, in Figure 52 the numerical results obtained from a Matlab script are overlapped on a satellite image. With respect to the dataset employed for VO problems, the one presented is rather challenging, thus results can be said rather satisfying.

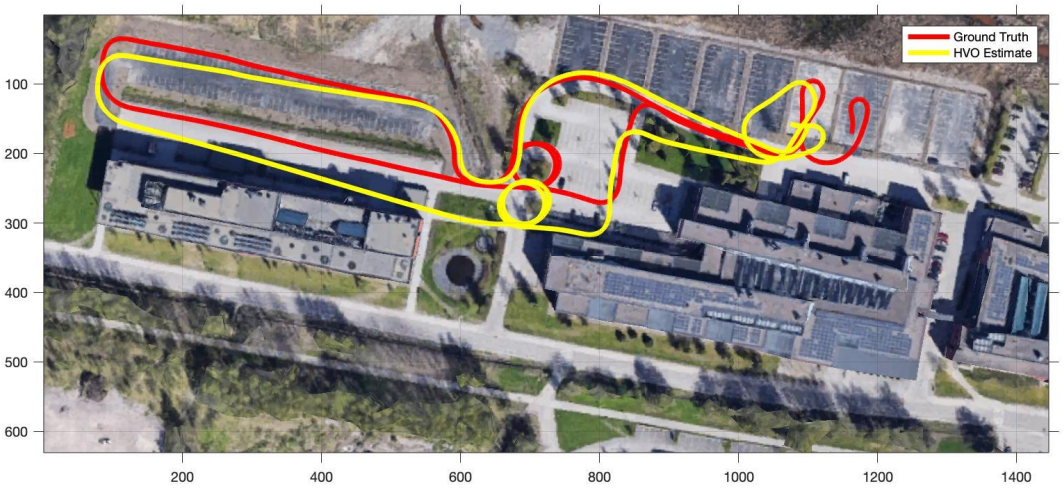


Figure 52: Comparison between numerical results and the real track shape.

6 Software-In-the-Loop

Once the best setting for the application has been found, it is possible to fix the parameters and move towards the architecture definition. In order to generate a C++ source for ROS, it is necessary to distinguish which lines of the code are part of the HVO function and which ones accomplish to external functionalities (e.g. sending the images from the camera).

6.1 Software framework

Starting from the Matlab script, it is possible to collect some pieces of the code in subsections according to several levels of organisation. In order to distinguish the code addressed to a pure odometry purpose from external auxiliary functionalities it is necessary to understand at which level a given line must be addressed. The general script framework is presented in Figure 53.

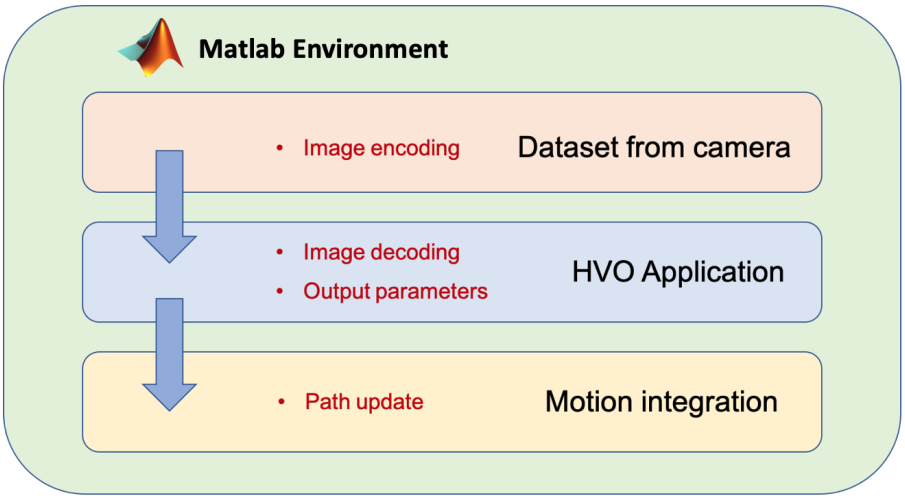


Figure 53: Organisation levels of the Matlab model.

The first block titled *Dataset from camera* contains the raw images directly extracted from the *camera.bag* file from ROS. In such section the image is encoded in order to be sent to the application box. Such piece of code has been written in order to simulate the camera behaviour and a memory element which allows to save the previous frame and send to the next stage a couple of subsequent frames.
The light blue box *HVO Application* contains all the lines which are addressed only to odometry purposes. At this stage, the images are decoded and transformed in unsigned integer 8-bit matrices. Once the information have been

extracted, the application returns two motion parameters: the absolute displacement ρ computed in the inter-frame period and the yaw estimated by the visual compass ψ. In order to avoid the occupancy of memory inside the application itself, the *Motion Integration* is thought to be made on a separate box which also provides a user interface.

6.2 HVO.m Matlab file

According to the subdivision made in the last section, the portion of the code to be separated and generated in a C++ source is the HVO Application, whose content is specified here below.

```
function [rho, yaw] = HVO(cam_time1, cam_time2,...
                          frame1, frame2, vx, vy)



    %% Preprocessing of the images received

    % In order to speed-up the computation, we
    % scale by 4 the image resolution.The actual
    % image dimension is expected to be 1236x1628
       pixels.
    % The low-resolution version is expected
    % to be 309x407 pixels.

    img1 = imresize(frame1, 0.25);
    img2 = imresize(frame2, 0.25);

    %% Visual Compass for getting the yaw
    %% angle (frame-to-frame)

    % Since the camera center is not well defined
    % because of problems of construction (i.e.
    % perpendicularity of the axes, orthogonality of
       the
    % image plane etc), we measure the steering by
       mean
    % of two windows (above below the supposed camera
    % center). The final output is a mean value.

    alfa = -7:7; % Possible shift in pixels
```

```
28      diff_down = zeros(length(alfa), 1);
29      diff_up = zeros(length(alfa), 1);
30
31          for a = 1:length(alfa)
32
33              diff_down(a) = sum(sum(img1
                    (238:273,179:229)...
34                   - img2(238:273,179+alfa(a):229+alfa(a))
                        ));
35              diff_up(a) = sum(sum(img1(50:120,179:229)
                    ...
36                   - img2(50:120,179+alfa(a):229+alfa(a)))
                        );
37
38          end
39
40      sqrdiff_down = sqrt(double(diff_down));
41      sqrdiff_up = sqrt(double(diff_up));
42
43      [~, ind_down] = min(sqrdiff_down);
44      [~, ind_up] = min(sqrdiff_up);
45
46      % Once we get the number of shift in pixel, one
47      % wants to pass in degrees and so it is necessary
48      % to multiply each results by a gain (practically,
49      % a density of deg/pixel). Such values are
            computed
50      % according to the mean-radius circumference
            passing
51      % through the centre of the windows.
52      % C = 2*pi*r_mean ==> density = 360/C
53
54      d_up = -0.5406;
55      d_down = 0.5354;
56
57      yaw_up = alfa(ind_up) * d_up;
58      yaw_down = alfa(ind_down) * d_down;
59
60      % Estimate of the frame-to-frame steering
61      yaw = (yaw_up + yaw_down) / 2;
62
63
```

```
64      %% Absolute scale recovery
65
66      v = sqrt(vx^2 + vy^2); %[m/s] Speed module
67      dt = cam_time2 - cam_time1; %[s] Integr. time
68
69      rho = v * dt; %[m] Absolute real distance driven
70                    % in the frame-to-frame interval
71
72
73  end
```

The first arguments which the *HVO.m* function receives are the camera times at which each frame is taken. Instead of fixing the camera rate, a more robust solution is to estimate the inter-frame period directly by difference (*line 67*), since some external factors could affect the camera working condition. In such case, the frequency may change a little. Secondly, a couple of subsequent frames are obtained, as well as the speed components V_x and V_y filtered from the already implemented EKF.

6.3 Matlab EXecutable (MEX)

Once the model is defined and optimised, it is possible to iterate between the SW Design and the Code Generation steps of the V-shaped process. In such phase, it is required to generate according to the language supported by the target platform. This process is named after Software-In-the-Loop (SIL). In this regard, the application is written in C++, whereas the rest of the actors are still in the shape of Matlab files.

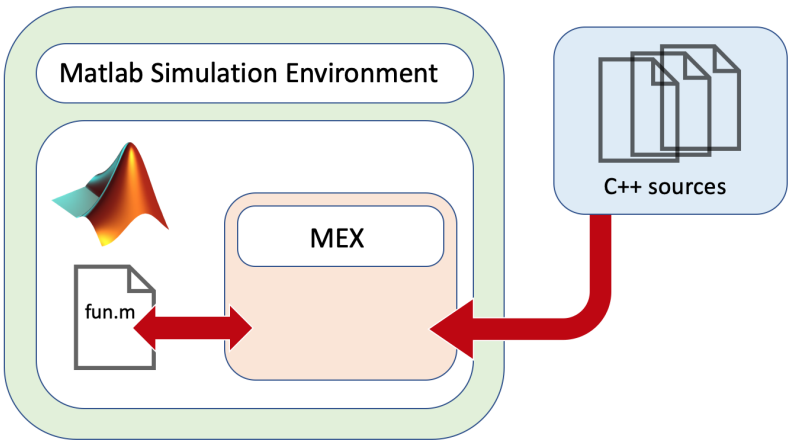


Figure 54: Matlab EXecutable concept for SIL tests.

The Matlab Coder functionality allows to test the application directly written in the target platform language. It can be done by generating the target code and then creating an internal environment in which such code can be executed. The reason why it is necessary to test the code written in the target language is to be addressed to eventual modifications automatically made by the coder which could affect the performances of the application (e.g. the change in the variables type).

6.4 Code generation and Performances

In order to test and compare the performances from MIL to SIL, it is required to create a test bench on the simulation environment in which the generated function is called. At this step it is not necessary to test the whole dataset but a significant bunch of frames are enough in order to verify if the the path is estimated likewise.

```
%% ================== HVO TEST ==========================

for i = 2 : 1000 % No. of frames

    % Image decoding

    img1 = copyImage(msgCam{i-1});
    img2 = copyImage(msgCam{i});

    img1 = readImage(img1);
    img2 = readImage(img2);

    % ========== Function call ==============

    [rho, yaw] = HVO(time(i-1), time(i), ...
        img1, img2, vx(i), vy(i), df, range);

    [rho, yaw] = HVO_mex(time(i-1), time(i), ...
        img1, img2, vx(i), vy(i), df, range);

    % ======================================
end
```

The automatic code generation produced a C++ application whose performances in terms of estimate perfectly follow in the steps of the *HVO.m* script. The last check to be done regards the execution time. It is remarkable to

remind that the final aim of the produced algorithm is to be run real-time directly on the Polaris platform in order to accomplish to the self-localisation problem. The final test which is needed to assure a real-time execution will be made in an HW/SW Integration phase, however at this stage it is important to guarantee that the execution time is at least lower than the camera working frequency.

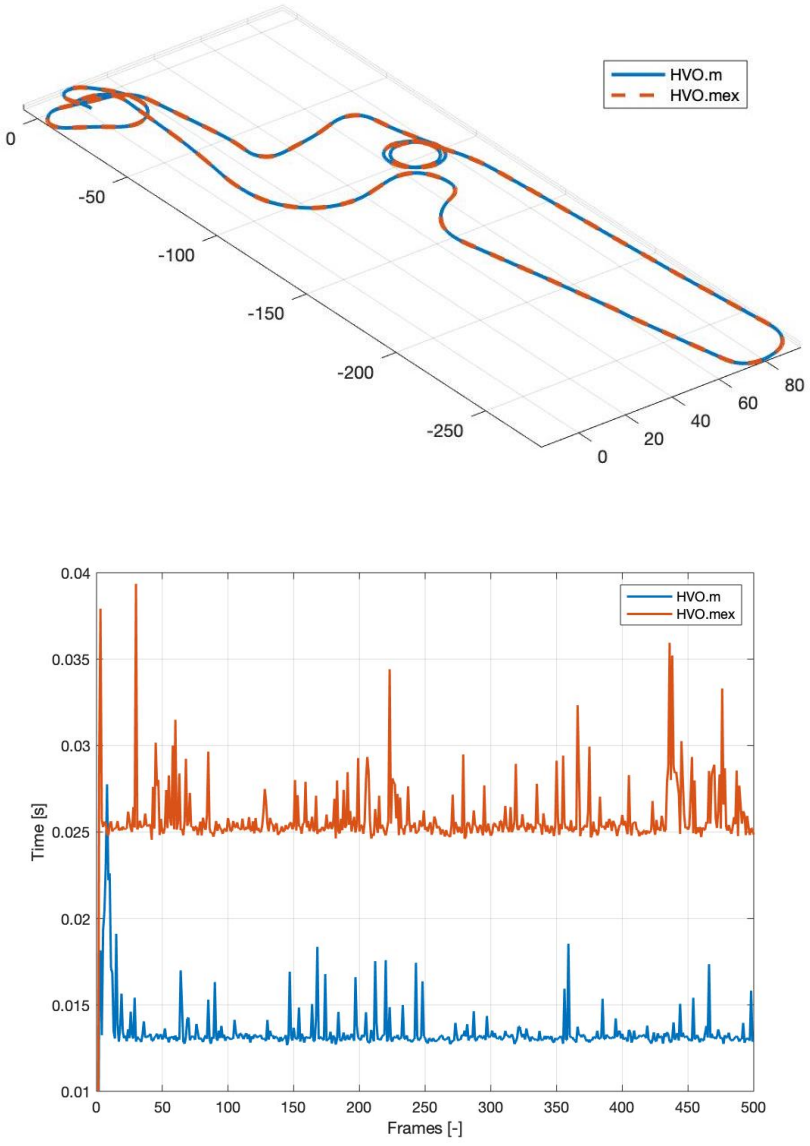


Figure 55: On the left, a comparison between the generated application results and the estimate made in the Model-In-the-Loop phase. On the right, the trend of the execution times both in Matlab and C++ shape.

To estimate the execution time it is possible to make use of the Matlab *tic()* and *toc()* functions right before and after the function call. Both the Matlab file and the generated C++ code are tested and the respective execution time is reported in Figure 55.
It is clear that the generated code *HVO.mex* requires a longer execution time with respect to the original model designed in the Matlab environment *HVO.m*. Indeed, the mean execution time is doubled from 0.0134s to 0.0258s. In terms of frequency it means that the Matlab function is able to process each couple of frames at a frequency of about 74Hz, while the MEX application guarantees a correct functioning not over the frequency of 38Hz. In any case, the obtained results are rather satisfying since the working frequency of the camera has been previously fixed at about 10Hz. Practically, it means that the odometry parameters are computed in about 1/4 of the frame-to-frame period.

7 Conclusion

7.1 Summary

The main purpose of this research work is the design of an application to accomplish to the self-localisation problem of the Polaris platform. In the Introduction, we presented the basics of Computer Vision which are necessary to follow the subsequent treatment. Particular attention is addressed to the feature tracking problem as well as to some direct approaches such as the Lukas-Kanade-Tomasi algorithm. In the same chapter, a presentation of the experimental platform is provided with a particular remark on the description of the mounting sensors. Finally, the adopted V-shaped process is presented and the research work framework is fixed.

The second chapter puts under the limelight the adopted camera model and gives a detailed presentation of the State-of-the-Art of Visual Odometry algorithms. A clear distinction is made between a pure VO approach and some Hybrid VO methods which make use of sensors other than camera. The pure VO approach has been presented in [14] and [15]. The former compares the performances returned by both an Optical Flow and the implementation of an Iterated Extended Kalman Filter and promotes the second method, though it requires a higher computational demand. The latter simplifies the estimation problem assuming the hypothesis of ground-planarity and ensures a real-time application. As a drawback, the implementation of the last procedure might bring to a failure if the majority of the features are not tracked on the ground. At the moment, more robust approaches fuse the camera information with other on-board sensors. Such methods are named after Hybrid Visual Odometry. In [16], only the heading is computed from the camera by a 1-Point-RANSAC algorithm, which makes use of the non-holonomic constraints in order to simplify the model and speed-up the execution.

In the third chapter, a first attempt of Camera calibration is proposed by mean of the Ocam Calib Toolbox [13]. The toolbox returns the Taylor polynomial calibration coefficients, while the affine transformation as well as the normalisation onto the unit sphere are implemented in custom-made Matlab functions. In the same chapter, the needed Image pre-processing procedure is described. The main steps are: RGB-to-Grey conversion, image resolution definition and a mask filter to reject useless image portions.

In the fourth chapter, a proposal of Pure VO application is provided. The design is mainly based on the ground-planarity method explained in [15]. The

full VO process can be listed as: features tracking, yaw detection by visual compass, model estimation by RANSAC, homography decomposition and motion integration. However it came up that the actual Polaris equipment is not suitable for pure VO methods, since the coupling camera-mirror is a degenerate configuration with no single effective viewpoint. As a result, the absolute path scale has not been recovered. Moreover, the majority of the features are tracked on the background because of the camera FoV, thus not enough data are returned to properly estimate the homography matrix.

In the fifth chapter, we stepped from a pure VO to a Hybrid Visual Odometry which makes use of the visual compass to estimate the platform heading and recovers the speed information from an EKF which fuses the other on-board sensors estimates. Several settings regarding a different combination of image resolution and compass ranges are tested. Finally, the best result is obtained after scaling of 1/4 the image size and looking for a lateral shift in a range of $[-7, 7]$ pixels. In the end, an overall analysis is carried out and possible sources of errors are highlighted.

Once the MIL simulations returned the wished results, in the sixth chapter we stepped into a Software-In-the-Loop phase. At this stage, it is required to define the software architecture and to separate the piece of code which must be turned into a C++ source suitable for a ROS implementation. The generation is made by mean of the Matlab Coder application and the simulations are carried out by calling the C++ source under the shape of a MEX file. Finally, performances are tested with an appropriate bench test script. Results are outstanding as far as the precision in the estimate is concerned. As the execution time is regarded, the generated application is able to compute the odometry parameters from a couple of frames at the rate of 38Hz. Given a camera working frequency of 10Hz, results can be said satisfying.

7.2 Future challenges

At the end of the presented research work, it is possible to say that the Polaris Ranger platform is now able to deal with the self-localisation problem with a good accuracy. With respect to the State-of-the-Art scientific papers, the ground truth employed for testing the VO and HVO applications is rather challenging, thus even better performances are expected on less sophisticated tracks.

One of the main issues regarding the developed algorithm is the high specificity of the chosen parameters. It means that such configuration returns good path estimates if the camera configuration and the relative FoV are kept unvaried. The change of such settings implies the necessity to re-calibrate the visual

compass box position. As it regards, it is recommendable to develop an optimisation algorithm which is able to compute the optimum box position according to a pre-established path every time the odometry functionality is called for the first time.

If one wants to implement a pure VO functionality, it is of primary importance the correction of the camera-mirror coupling. In order to tone down the cost of the modification, it is not recommendable to change the perspective camera with an orthographic one. A first solution can be the reduction of the camera-mirror distance. Though a perspective projection is subjected to a vanish point, if the zoom is high enough, it can be easily confused with an orthographic projection. A simplified example is shown in Figure 57.

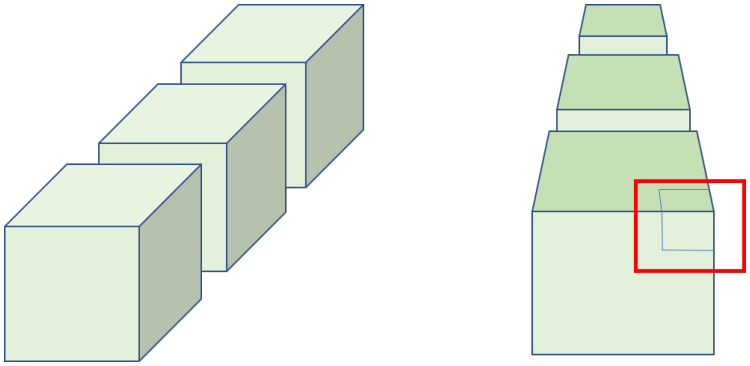


Figure 56: On the left, an example of orthographic projection where no depth factor is detectable. On the right, a perspective projection example with a vanish point and a depth definition. A zoom on a perspective projection assumes similar characteristics of an orthographic projection.

If such correction is not enough, the best solution is the change of the mirror. According to the study conducted by Baker and Nayar in 1998, a non-degenerate configuration can be obtained by putting together a perspective camera with a hyperboloidal, planar or ellipsoidal mirror. Following in the steps of [16], it is possible to replace the current parabolic mirror with a *KAIDAN 360 One VR* hyperbolic model. The verification of the SVP property can be lead by the Ocam Calib Toolbox "Find Centre" functionality. If a single viewpoint exists, the implemented optimisation process will return the image plane centre after a finite and relative small number of iterations.

References

[1] Sherif A. S. Mohamed et al., *"A Survey on Odometry for Autonomous Navigation Systems"*, *IEEE Access*, 6 Aug. 2019.

[2] R. Szeliski, *"Computer Vision: Algorithms and Applications"*, Springer, 3 Sept. 2010, pp. 207-237.

[3] S. Lazebnik, *"Corner detection"*, Computer Vision course slides, University of Illinois, 7 Feb. 2019, refer to slazebni.cs.illinois.edu.

[4] R. Hartley, A. Zisserman, *"Multiple View in Computer Vision"*, Second edition, Cambridge University Press, 2003, pp. 239-247.

[5] B. D. Lukas, T. Kanade, *"An iterative image registration technique with an application to stereo vision"*, in Proc. 7th Int. Joint Conf. Artif. Intell. (IJCAI), vol. 2, San Francisco, 1981, pp. 674-679.

[6] Polaris Inc., *"Device Description of Polaris Ranger EV"*, refer to https://ranger.polaris.com/en-us/ranger-ev/.

[7] T. Badar, *"Implementation of the autonomous functionalities on an electric vehicle platform for research and education"*, M.Sc. Thesis, Pakistan, 20 May 2019, pp. 1-4 and 36-44.

[8] V. Kukkonen et al., *"Perception Platform for Autonomous Vehicles"*, Project Work course, Aalto University, 22 May 2017.

[9] P. Bing, S. Wentao, L. Gilles, *"Effect of camera temperature variations on stereo-digital image correlation measurements"*, The Optical Society, Applied Optics, 2015.

[10] D. Scaramuzza, *"Omnidirectional vision: from calibration to robot motion estimation"*, PhD Thesis, ETH Zurick, 2008, pp. 9-46.

[11] B. Micusik, *"Two View Geometry of Omnidirectional Cameras"*, PhD Thesis, Center for Machine Perception, Czech Technical University in Prague, 2004.

[12] C. Geyer, K. Dandiilidis, *"A unifying theory for central panoramic systems and practical applications"*, European Conference on Computer Vision (ECCV), Jun. 2000, pp.445-461.

[13] D. Scaramuzza, A. Martinelli, R. Siegwart, *"A Toolbox for Easy Calibrating Omnidirectional Cameras"*, Proceedings to IEEE International Conference on Intelligent Robots and Systems (IROS 2006), Beijing China, 7-15 Oct. 2006.

[14] P. Corke, D. Strelow, S. Singh, *"Omnidirectional Visual Odometry for a Planetary Rover"*, Robotics Institute, Pittsburgh, USA, July-Oct. 2003.

[15] D. Scaramuzza, R. Siegwart, *"Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles"*, IEEE Trans. on Rob. vol. 24 no. 5, 18 May 2014.

[16] D. Scaramuzza, *"1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints"*, Springer Science, Int. J. Comput. Vis., 7 Apr. 2011.

[17] D. Tyagi, *"Introduction to SURF (Speed-Up Robust Features)"*, Data Breach, 20 Mar. 2019, refer to https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e

[18] P. Thipkham, *"Image Processing Class #5 - Edge and Contour"*, Towards Data Science, 26 Dec. 2018, refer to https://towardsdatascience.com/image-processing-class-egbe443-5-edge-and-contour-d5d410f4483c

[19] Y. Ma, S. Soatto, J. Kosecka and S. Sastry, *"An Invitation to 3D Vision: From Images to Models"*, Springer-Verlag, New York, Dec. 2003.

[20] B. Triggs, *"Autocalibration from Planar Scenes"*, European Conference on Computer Vision (ECCV '98), Freiburg, Germany, Jun. 1998, pp.89-105.

[21] D. Nister, *"An efficient solution to the five-point relative pose problem"*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 6, pp. 756-770.

[22] S. Baker, S. Nayar, *"A theory of single-viewpoint catadioptric image formation"*, International Journal of Computer Vision, 1998.