CRANFIELD UNIVERSITY

LOUNIS CHERMAK

STANDALONE AND EMBEDDED STEREO VISUAL ODOMETRY BASED NAVIGATION SOLUTION

CRANFIELD DEFENCE AND SECURITY

PHD
Academic Year: 2014 - 2015

Supervisor: Dr N AOUF
August 2014

CRANFIELD UNIVERSITY

CRANFIELD DEFENCE AND SECURITY

PHD THESIS

Academic Year 2014 - 2015

LOUNIS CHERMAK

STANDALONE AND EMBEDDED STEREO VISUAL ODOMETRY BASED NAVIGATION SOLUTION

Supervisor: Dr. N AOUF
August 2014

ABSTRACT

This thesis investigates techniques and designs an autonomous visual stereo based navigation sensor to improve stereo visual odometry for purpose of navigation in unknown environments. In particular, autonomous navigation in a space mission context which imposes challenging constraints on algorithm development and hardware requirements. For instance, Global Positioning System (GPS) is not available in this context. Thus, a solution for navigation cannot rely on similar external sources of information. Support to handle this problem is required with the conception of an intelligent perception-sensing device that provides precise outputs related to absolute and relative 6 degrees of freedom (DOF) positioning. This is achieved using only images from stereo calibrated cameras possibly coupled with an inertial measurement unit (IMU) while fulfilling real time processing requirements. Moreover, no prior knowledge about the environment is assumed.

Robotic navigation has been the motivating research to investigate different and complementary areas such as stereovision, visual motion estimation, optimisation and data fusion. Several contributions have been made in these areas. Firstly, an efficient feature detection, stereo matching and feature tracking strategy based on Kanade-Lucas-Tomasi (KLT) feature tracker is proposed to form the base of the visual motion estimation. Secondly, in order to cope with extreme illumination changes, High dynamic range (HDR) imaging solution is investigated and a comparative assessment of feature tracking performance is conducted. Thirdly, a two views local bundle adjustment scheme based on trust region minimisation is proposed for precise visual motion estimation. Fourthly, a novel KLT feature tracker using IMU information is integrated into the visual odometry pipeline. Finally, a smart standalone stereo visual/IMU navigation sensor has been designed integrating an innovative combination of hardware as well as the novel software solutions proposed above. As a result of a balanced combination of hardware and software implementation, we achieved 5fps frame rate processing up to 750 initials features at a resolution of 1280x960. This is the highest reached resolution in real time for visual odometry applications to our knowledge. In addition visual odometry accuracy of our algorithm achieves the state of the art with less than 1% relative error in the estimated trajectories.

ACKNOWLEDGEMENTS

In the name of Allah, the Entirely Merciful, the Especially Merciful. All praise to my Lord Allah who has given me the strength to fulfil this PhD.

I would like to thank warmly the following for supporting me.

To my supervisor Dr. Nabil Aouf for believing in me and for giving me the opportunity to do this project, also for his continuous guidance and technical advice.

To my committee members, especially Professor Mark Richardson for his advices.

To Cranfield University and the Shrivenham Defence Academy for providing a good environment for research. To Thales and the European Space Research and Technology Centre of the European Space Agency for funding this research project.

To my lab fellows with whom I shared a lot of good moments, interesting discussions and for their everyday support. My old friend Saad, Diego, Xiaodong, Yassine, Redouane, Javier, Saif, Tarek, Riad, Mohammed, Abdenour, Oualid, Abderahim, Abdelmadjid, Amine, Abdelalim, and Bilel.

To the people at ESTEC, Gianfranco, Martin, Tim, Robin, Jan Hein, Carlos, Michel, Pantelis. Thank you for welcoming me there, and for your support.

To my parents to whom I owe almost everything, thank you so much for your prayers, love and support.

To my brothers Edriss and Gebril, for their love, support and precious advises.

To my dear wife Naima and my two daughters Maryam and Assya. You are my everyday motivation, thank you for your encouragement and enduring patience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	Two Dimensions
3D	Three Dimensions
AHRS	Attitude and Heading Reference System
ALTW	Adaptive Local Tracking Windows
API	Application Programming Interface
BA	Bundle Adjustment
CENSURE	Centre Surround Extremas for Realtime
CMOS	Complementary Metal–Oxide–Semiconductor
CP	Cauchy Point
CPU	Central Processing Unit
LAGR	Learning Applied to Ground Vehicles
DDL	Double Dogleg
DL	Double Dogleg
DOF	Degrees Of Freedom
DSP	Digital Signal Processor
EKF	Extended Kalman Filter
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
FAST	Features from Accelerated Segment Test
FPGA	Field-Programmable Gate Array
fps	Frames per second
GFTT	Good Features To Track
GN	Gauss Newton
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDR	High Dynamic Range
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IR	Infrared
KLT	Kanade Lucas Tomasi
LBA	Local Bundle Adjustment
LM	Levenberg Maquardt

MEMS	Microelectromechanical Systems
MP	Megapixel
NASA	National Aeronautics and Space Administration
OpenCV	Open Computer Vision
PSU	Power Supply Unit
RANSAC	Random Sample Consensus
RGB	Red Green Blue
SBC	Single Board Computer
SDK	Software Development Kit
SGM	Semi Global Matching
SIFT	Scale-Invariant Feature Transform
SSD	Solid State Drive
SSH	Secure Shell
SURF	Speed Up Robust Features
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
VO	Visual Odometry

1. Introduction

This thesis investigates stereo visual odometry for autonomous navigation in space mission context. Visual odometry is the process of estimating camera motion from the analysis of feature correspondences within a sequence of images over time. The accumulation of these relative inter-frame motions leads to the generation of a full trajectory that a camera equipped platform has travelled. Stereo vision allows depth information recovering via stereoscopy. This is not the case for monocular based visual odometry solutions, which are only able to recover the motion up to a scale. Although the aim of the PhD project targets space mission constraints, applications of visual odometry are wide and include robotics, automotive, wearable computing, and augmented reality. The aim of this work is to design an autonomous visual stereo based navigation sensor to equip any type of moving platforms enabling the generation of accurate trajectories for platform navigation purpose.

The process of visual odometry is composed of several and subsequent stages including feature detection, stereo matching, feature tracking and motion estimation. This thesis deals with each of these stages by proposing novel techniques such as IMU assisted feature tracking and the adoption of the double Dogleg minimisation algorithm to solve the local bundle adjustment problem for accurate motion estimation.

The visual odometry solution was developed with the assumption of no prior knowledge of the explored environment. The performance of the developed techniques is tested on various types of environments and illustrated all along the thesis.

1.1 Background

A large variety of mobile robots, robotic manipulators and other autonomous systems, compose the essential part of the required devices for exploration and operational tasks in space missions. These can be used on planetary surfaces, on orbit or to assist astronauts. In such missions, contribution of a robot is essential by its ability to perform the work independently to which it has been assigned. Being independent from the command control station or any other third party human operators is not an easy task. This requires capabilities for sensing and perception of the unstructured surroundings and possibly occluded environments as well as interpreting acquired data to achieve the mission objectives in a precise and reliable manner. Challenges offered by space missions require a high level of efficiency in order to operate within the strict limitations of data processing and computational hardware resources while also dealing with effects of the challenging space environment.

In order for unmanned robotic systems to achieve their goals, it is essential that they first acquire meaningful information about their environment. This information is crucial as it first enables the platform to have the knowledge of its surrounding but also to have the knowledge of its own status. Consequently, the right decisions can be taken in order to achieve a specific task. To be able to reach that, unmanned robotic systems need to be equipped with a cleverly thought combination of sensors able to cope with space mission high's standard demands. Indeed, due to the nature of space environments some sensors cannot be used. For instance, the Global Positioning System (GPS), which is widely used for earthbound applications, is unfortunately not usable in certain space context (Mars for ex). Even for operations on the surface of the Earth, satellite signals are not available all the time. Obviously a perfect sensor does not exist. Ideally a complementary combination of passive sensors has to be found. These have to be also reasonable in term of energy consumption.

1.2 Aims and Constraints

The aim of this thesis have been in part defined from the request of our sponsor, the European Space Agency (ESA) to efficiently tackle autonomous navigation generally and in particular in space mission context by providing accurate related absolute and relative 6 degrees of freedom (DOF) positioning of a dedicated navigation platform. This involves research in the areas of stereovision, visual motion estimation, optimisation, data fusion and sensor design. This aim is reached by conceiving novel methods and also by adapting or refining existing ones to reduce accuracy errors, increase robustness and to achieve real-time performance processing. The application of these techniques is the estimation of the trajectory that a camera equipped platform has travelled the most accurately and this regardless of the environment constraints. This is materialised with the conception of a smart standalone stereo visual/IMU navigation sensor that meets the previously stated requirements. Notably, the scope of this thesis is limited to the generation of accurate navigation. Mapping, path planning, and decision making are outside the scope of this work. There are a number of constraints that have to be considered in the perspective of realising this aim. These are listed below:

- The main sensor to be used is stereo cameras. This gives a cheap and a reliable portative solution and enables 3D position recovering. An inertial measurement unit (IMU) passive sensor is also used as a complementary source of information.
- The visual navigation sensor solution has to be autonomous and not depending on an external source of information or human interaction.
- The visual navigation sensor solution has to cope with arbitrary three-dimensional movements as it is aimed to adapt and being mounted on to various kinds of platforms (vehicle, robotic arm, handheld operator). These can imply instability, as well as uneven motions.

- The visual navigation sensor solution has to achieve real time performance. This includes all the operations, from stereo image and inertial data acquisition to all the processes composing the complete visual odometry algorithm. A frame rate of 5 frames per second (fps) is considered as sufficient to achieve accurate navigation.
- It is assumed that the environment is mostly static. However, the software solution developed has to cope with dynamic information that might occur during exploration tasks.

It is also assumed that the environment contains a minimum of texture and visual features. Structured and unstructured scenes have to be tackled

1.3 Organisation of the Thesis and Contributions

The thesis is composed of six chapters in addition to the present introduction. The following gives an insight of the content of each chapter. For clarity and best readability of the thesis, a dedicated review of the chapter related work is given in each chapter instead of including a global and long related work section here in the introduction.

Chapter 2: Stereo Visual Odometry

In this chapter, the concept of visual odometry is presented. It starts with very basic understanding of imaging, camera model and stereovision. It also includes explanation of feature detection, stereo matching and feature tracking methods as well as the strategy built to form the basis of our stereo visual odometry algorithm.

Chapter 3: HDR imaging: Feature Detection and Tracking

This chapter presents an innovative experimental application of the comparative use of a general camera and an HDR imaging sensor for feature detection and tracking operations. This includes a large range of extreme illumination conditions experiments in outdoor and indoor environments.

Chapter 4: Motion Estimation

In this chapter, two different motion estimation methods are presented. The first one is a 3D to 3D structure based motion estimation approach solved with the quaternion motion estimation method combined with a geometric constraints based outliers rejection scheme. The second method presents an innovative two view local bundle adjustment scheme based on trust region minimisation method. This solves 3D structure to 2D image plane re-projection problem for precise motion estimation. It also included comparison between the widely used Levenberg-Marquardt minimisation algorithm and the novel double Dogleg minimisation algorithm that is newly introduced in visual odometry context. Experimental results in outdoors urban datasets are presented and showed a clear superiority in terms of accuracy for 3D to 2D motion estimation scheme than for 3D to 3D structure approach. It is also shown that using double Dogleg algorithm is generally providing more accurate and stable trajectories than with Levenberg-Marquardt algorithm.

Chapter 5: Improved Visual Motion Estimation with IMU Assisted Feature Tracking

In this chapter, a novel stereo visual IMU-assisted technique that extends the use of the KLT tracker (Kanade-Lucas-Tomasi) to a large inter-frame motion by associating the IMU readings to visual data is proposed. The high sampling rate of IMU gives a constrained and coherent inter-frame pose. When applied to features via 3D geometry and stereoscopy properties, it predicts efficiently the projection of the optical flow in subsequent images. Accurate adaptive tracking windows limit tracking areas resulting in a minimum of lost features and avoid tracking dynamic objects. This new feature tracking approach is adopted as part of a fast and robust visual odometry algorithm. The motion estimation algorithm uses double Dogleg trust region method in an inter-frame bundle adjustment scheme. The resulting pose is then used to update the IMU. On an urban environment dataset, it is demonstrated that our technique greatly improves the original KLT tracker. Comparisons with gyro-aided KLT and IMU variants of like KLT at different frame rates show that our technique is able to maintain minimum loss of features and low computational cost even on image sequences presenting important scale change.

The visual odometry solution based on this IMU-assisted KLT gives more accurate result than INS/GPS solution for trajectory generation in a certain contexts.

Chapter 6: Smart Visual/IMU Standalone Navigation Sensor

This chapter presents the very innovatively designed smart, standalone multi-platform Stereo/IMU based navigation system providing ego-motion estimation. Full details on the choice of the material and components of the visual/IMU navigation sensor are given. Also software and hardware implementation including the Graphics Processing Unit (GPU) part enabling us to achieve real-time processing are thoroughly explained. The experimental validation of our designed sensor was conducted at European Space Research and Technology Center (ESTEC), in a pool reproducing Mars-like environment. This includes the preparation setup, data collection and results.

1.4 Software Tools

OpenCV computer vision library [103] was used to develop our feature detection, and feature tracking algorithms. A part from that, the full development of the work carried out for this PhD project has been coded from scratch in C/C++. This includes the architecture of the visual odometry pipeline, the IMU assisted feature tracking, motion estimation and minimisation algorithms. Note that the 3D visualisation and plotting were realised with MATLAB software [104].

2 Stereo Visual Odometry

2.1 Overview

The process of estimating camera motion from the analysis of feature correspondences within a sequence of images over time is called visual odometry. The accumulation of these relative motions leads to the generation of the full trajectory that a camera equipped platform has travelled. Visual odometry is thus a computer vision application which largely (or exclusively) depends on the acquired visual information. This chapter introduces the basic principles of imaging, which form the foundations of the work carried out in this project. This induction aims to help the reader to familiarise with the concepts of computer vision, which are present in all the study. After reviewing the related work that popularised visual odometry, basics on image representation are explained (Section 2.3). Then, the camera model (Section 2.4) giving the relationship between the image representation and its corresponding real world scene is presented. Stereo vision (Section 2.5), feature detection (Section 2.6), stereo and feature tracking (Section 2.7) are then introduced. This chapter ends with the presentation of the complete visual odometry pipeline.

2.2 Related Work

Over this last decade, numerous research works have been undertaken to investigate and exploit the huge potential of visual odometry. Initially, the use of cameras (monocular, stereo or multiple cameras) for autonomous ground vehicles was investigated as a complement of wheel odometry and inertial sensors. The later are likely to slip on uneven terrain (i.e. causing data corruption) and suffer from relative drift with the time respectively. In addition to be cheap, cameras contribute by acquiring visual data providing substantial information of the surrounding environment at a relatively high sampling rate. The analysis of an image sequence can be used to estimate the motion of a camera equipped moving platform. The term visual odometry firstly appeared in [1]. However, it was popularised by Nister [2] although the concept was described earlier in [3].Visual odometry development was mainly motivated by NASA Mars space exploration program for planetary Rovers to circumvent wheel slippage problems [3]–[6]. The visual system can be used in three configurations: monocular, stereo or multi-cameras. Monocular solutions for visual odometry are estimated up to a scale factor. It is determined at each new frame respectively to the two first camera poses using the trifocal tensor or using the known dimensions of a 3D object that appears in the scene [7]. Interesting results over long routes were shown over these past years [8]–[10]. The combination of stereo aligned cameras sharing a subsequent common field of view enables recovering the relative 3D position of the features by triangulation [11]. A large number of stereo visual odometry works demonstrated great achievements in terms of accuracy with a relative position error as approximately as lower than 1% to 2% [12], [13]. More details can be found in a survey and more recent tutorials giving a detailed picture of the different visual odometry approaches in general and stereo visual odometry approaches in particular [14]–[16].

2.3 Image Representation

An image is a multidimensional signal that derives from light captured from a digital camera and which contains visual information. An image is defined as a matrix where each cell, named pixel, is represented by a 2D index: u for the columns and v for the rows

(see Figure 2.1). A pixel stores a number representing the relative intensity of the captured light. The pixel intensity commonly takes a value from 0 to 255.

The capability (or size) of an image is characterised by its resolution, which is composed of three aspects:

Spatial resolution: This is the number of pixels (or matrix elements) of the image covering the visual space of the capture. It corresponds to the multiplication of the image columns (u) and rows (v) size. The size of the spatial resolution has an impact on the quality of the projection of the captured scene into an image. The larger the resolution, the better is the image quality. However, a higher resolution implies bigger size which may lead to storage problems.

Colour resolution: Generally, an image can be single or three channels. A single channel image is called grayscale. A grayscale image has a storage capability per pixel of 8 bits which means a pixel can take a value between 0 and 255. Pixel values approaching 0 are darker. Conversely, pixel approaching 255 are brighter. A three channels image is called colour image.

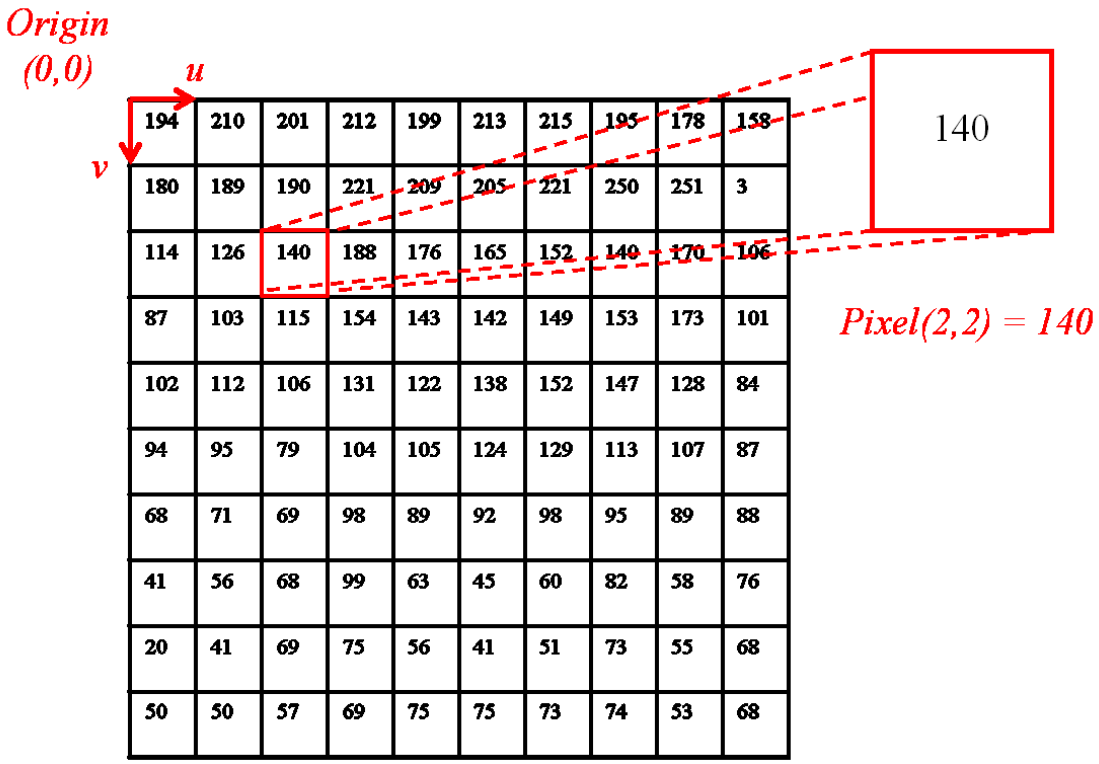


Figure 2.1. *Illustration of an image matrix representation.*

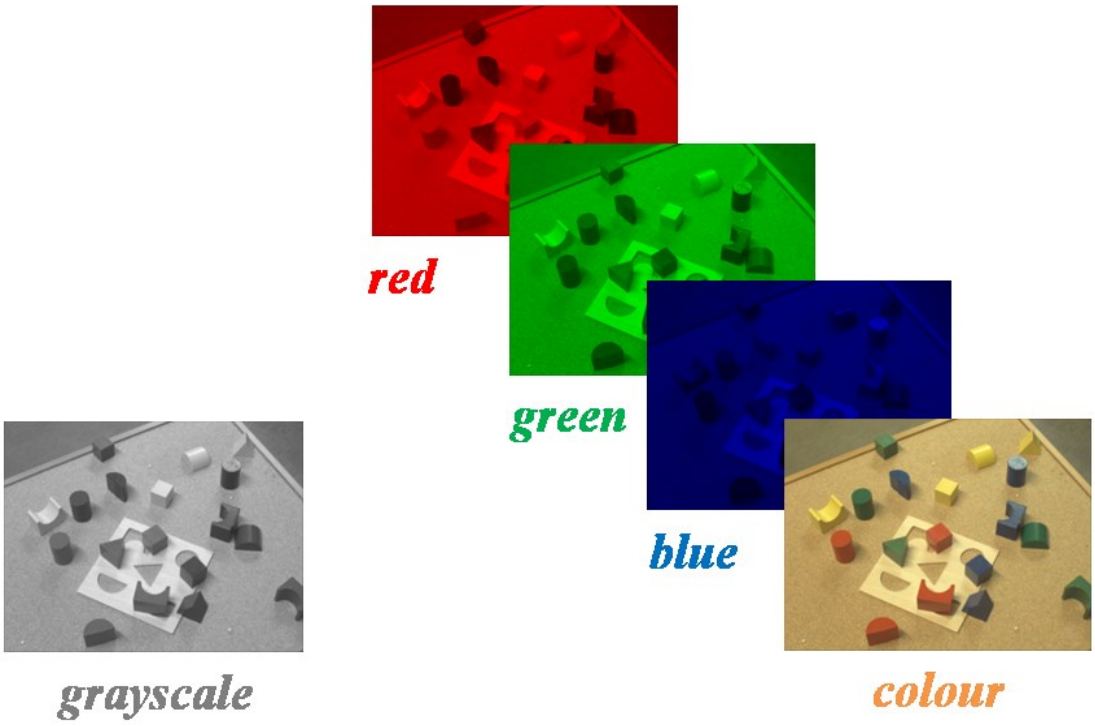


Figure 2.2 *Illustration of image colour resolution representations.*

A colour image capability is 24 bits (3x8 bits) where each channel represents a primary colour namely red, green, and blue (RGB). Figure 2.2 gives an illustration of these two image colour resolution representations.

Temporal resolution: This is the number of images continuously captured in a given time for a video device for instance. It is expressed as frames per second (fps).

The works carried out in this thesis, including tests and experimentations, make use of grayscale images.

2.4 Modelling Cameras

Since the aim of this project is to accurately estimate motion from visual data, it is important to define the transformation that links an image to real world metrics. The simplest model which describes this transformation is called *pinhole* perspective projection model. This model assumes that light rays reflecting from the scene pass through a small hole punched on a screen, which has for effect to invert the scene in the image plane as illustrated in Figure 2.3. This model gives the transformation between a

position (X, Y, Z) with Z characterising he object's depth into a pixel location (u,v) where the focal length *f* characterise the distance between the screen and the image.

Consequently, the relationship between the 3D and the 2D coordinates gives the perspective transform defined as follows:

$$\begin{cases} u = f\frac{X}{Z} \\ and \\ v = f\frac{Y}{Z} \end{cases} \tag{2.1}$$

Two notable proprieties derive from this projection model. First, the size of the object within the image depends on its distance from the camera. Hence, farther objects will appear small in the image. The other property is that parallel lines in the real world will appear converging in the image. It is also known as perspective distortion.

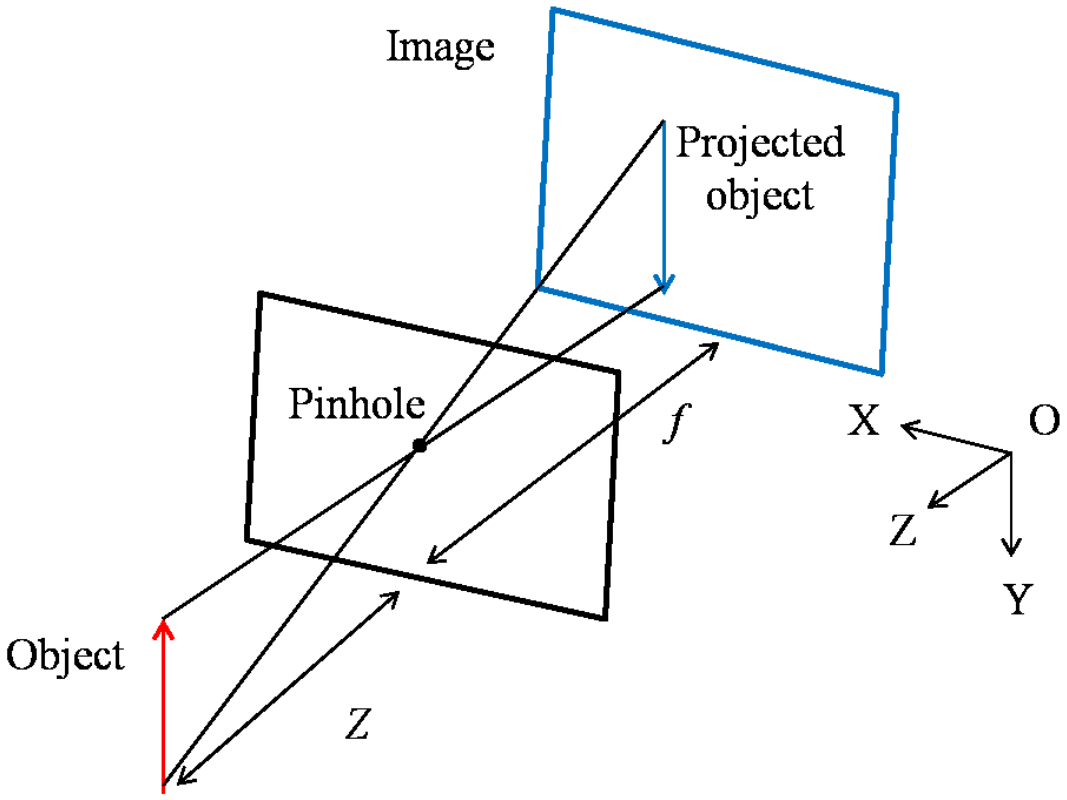


Figure 2.3. *Illustration of the pinhole perspective projection model.*

In reality, this model is more complicated as it implies lens distortion, which affects *f* resulting in an image with distorted corners. In order to have undistorted images, the camera needs to be calibrated, which means to estimate the extrinsic and intrinsic parameters of the camera.

The intrinsic transformation handles lens distortion and achieves the projection following the pinhole camera model. Optical distortion can be modelled following the radial lens model characterised with two parameters k_1 and k_2. Thus, distorted pixel coordinates can be undistorted as follows:

$$\begin{aligned} u &= u_d(1 + k_1(u_d^2 + v_d^2) + k_2(u_d^2 + v_d^2)^2) \\ v &= v_d(1 + k_1(u_d^2 + v_d^2) + k_2(u_d^2 + v_d^2)^2) \end{aligned} \tag{2.2}$$

where $[u_d, v_d]^T$ are the distorted pixel coordinates in the image.

The projection transform is characterised by a matrix *K*, which links homogenous coordinates of $\tilde{p}$ to its related 3D position *P* as follows:

$$\tilde{p} = KP$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.3}$$

The matrix *K* is composed of 5 parameters: f_u and f_v the focal length respectively in the horizontal and vertical directions, the skewing factor γ and the coordinates of the central pixel $[u_0, v_0]^T$. The central pixel is the position in the image where the optical axis crosses orthogonally the image plane. Generally, the central pixel does not coincide with the image centre. The relationship between f_u and f_v, is the following:

$$f_u = sf_v \tag{2.4}$$

where *s* is a scale factor, which is equal to 1 if the image pixels are square. Finally, the skew factor, which represents the angle between the directions $\vec{u}$ and $\vec{v}$ is usually fixed to zero because in reality its value is negligible as these two directions are perpendicular.

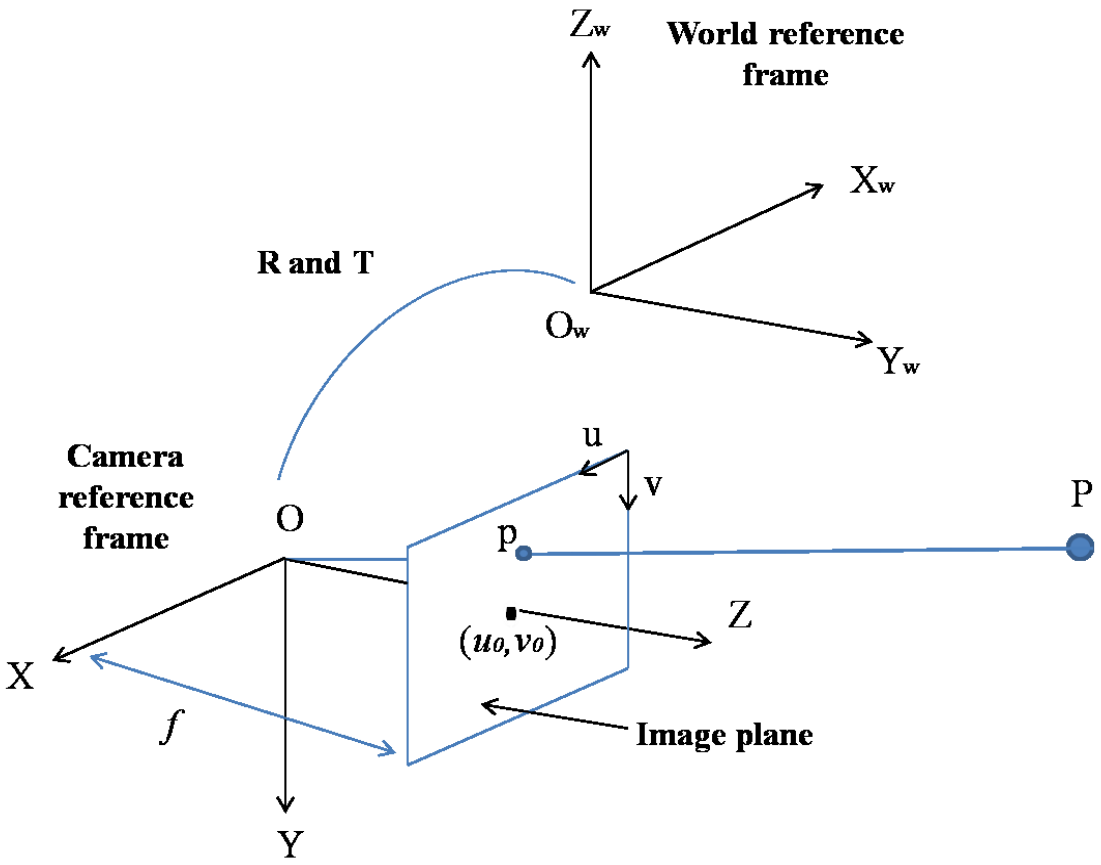


Figure 2.4. *Illustration of camera-world reference frame transformation.*

A rotation matrix R and a translation vector t define the extrinsic transformation. The later gives the relationship between the camera reference frame and world reference frame as illustrated in Figure 2.4. Thus, the operation, which moves a point P_w from the world reference frame to its related position P in the camera reference frame is described as:

$$P = RP_w + t \tag{2.5}$$

2.5 Stereo Vision

The work presented in this thesis is based on stereovision for visual odometry. A stereo setup is composed of two identical cameras aligned and mounted on a rigid frame. The distance separating the two cameras is called the Baseline (*B)*.

Stereo vision aims to reproduce the human visual system, which has the ability to deduce the 3D structure and distance information of an observed scene from two different views (i.e. eyes). Human vision offers perfect performance for almost every day usual tasks

without raising any difficulties. Applying this to computer vision brings two major challenges. The first one is the correspondence problem where the objective is to determine which objects in the left view correspond to the correct ones in the right view. This association problem is usually solved in part with different matching techniques. However, due to the stereo configuration some of the information will be missing in either in one or both views (e.g. occlusion of the background, sides of the images). The second problem relates to 3D reconstruction, which also depends on the first correspondence problem. Indeed, our brain is perfectly associating the items in the two views. Consequently, it can easily build a clear and continuous 3D map of the scene from the disparity (i.e. retinal position difference between an associated item in the two views). For computer vision, the quality of the generation of a 3D disparity map depends on its ability of matching features between the left and right views.

Ideally stereo images should be aligned exactly when the stereo cameras are mounted. However, this is not true in reality. Figure 2.5 illustrates the stereo camera configuration.

In a stereo vision configuration, the operation which moves a point P_L from the left camera reference frame to its related position P_R in the right camera reference frame is described as:

$$P_R = R_{LR}(P_L - t_{LR}) \tag{2.6}$$

where R_{LR} and t_{LR} are the extrinsic parameters linking the left and right cameras reference frames. An epipolar line is the line, which passes by the projection p into the image plane of the 3D point P and o (the image central pixel). The process, which aligns the stereo image, is called rectification. This manifests as having all the epipolar lines parallel with the horizontal axis. This property assures to the projections p_L and p_R of the point P to lie on the same horizontal coordinate (v) of the stereo image pair.

The main benefit of having p_L and p_R sharing the same horizontal coordinate $(v_L = v_R)$ is that it greatly facilitates feature matching between these two images.

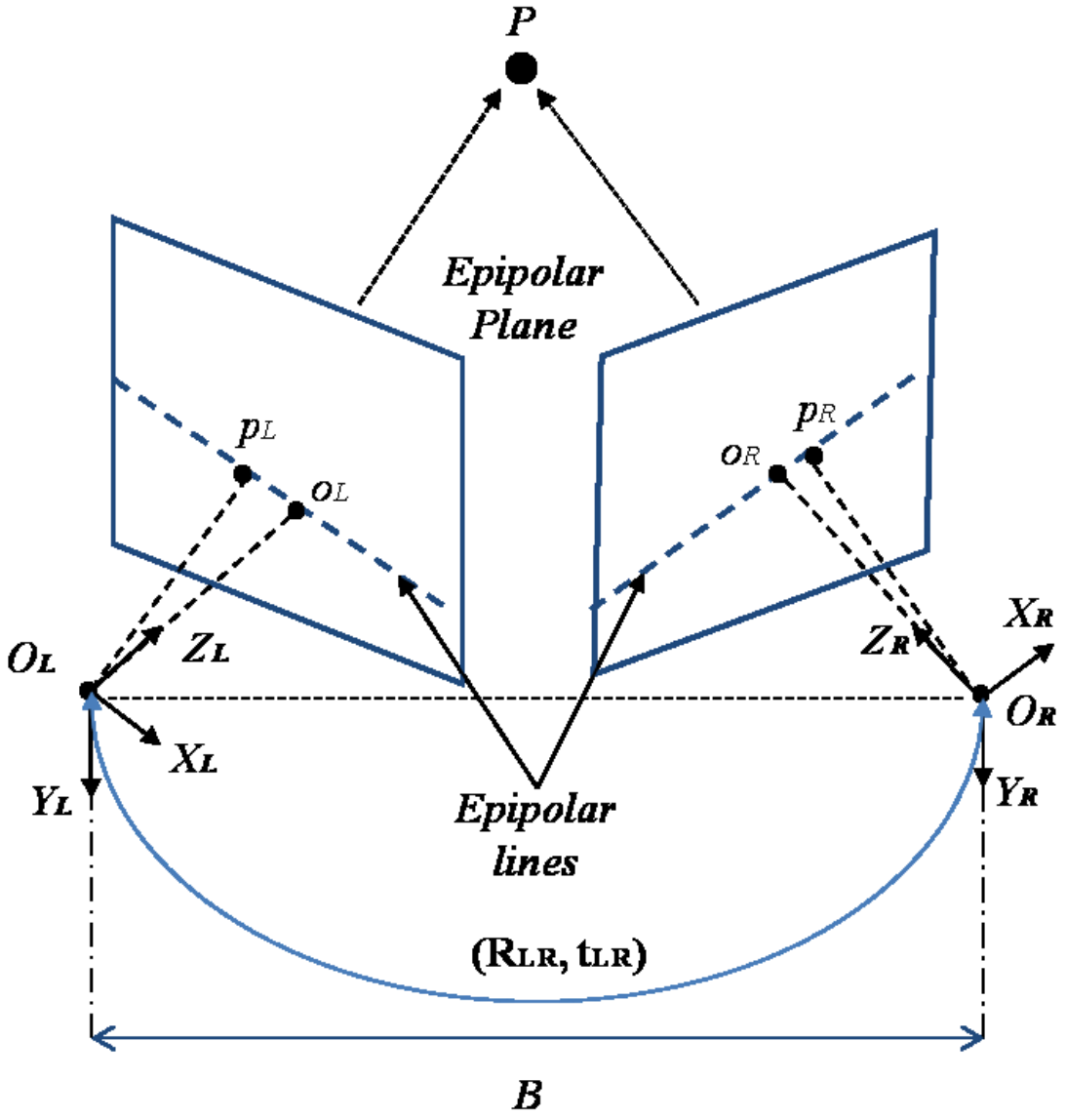


Figure 2.5. *Illustration of the stereo camera configuration.*

Thus, stereo image rectification consists in projecting the stereo images with the knowledge of both intrinsic (f, B, u_0 and v_0) and extrinsic camera parameters (R_{LR} and t_{LR}, linking both camera reference frames).

This should validate the condition that all the conjugate epipolar lines become collinear and parallel to the horizontal image axis. In this thesis, the stereo calibration parameters composed of the intrinsic and extrinsic parameters are assumed constant.

With the knowledge of the stereo camera calibration parameters, it is possible to reconstruct the 3D position P from a feature correspondence pair (p_L and p_R) by triangulation [11]. Let us consider $p_L = [u_L, v_L]^T$ and $p_R = [u_R, v_R]^T$ the left and right coordinates of a stereo correspondence . $P = [X, Y, Z]^T$ is the reconstructed point as follows:

$$\begin{cases} X = \frac{(u_L - u_0)B}{(u_L - u_R)} \\ Y = \frac{(v_L - v_0)B}{(u_L - u_R)} \\ Z = f\frac{B}{(u_L - u_R)} \end{cases} \tag{2.7}$$

The difference between the left and right vertical coordinates u_L and u_R is called the *disparity* which is inversely proportional to the depth Z.

2.6 Feature Detection

Feature detection stage is employed to find remarkable key points that can be easily identifiable and matched over a sequence of images. Detection of features is the starting operation for visual odometry without which the process would be impossible. An image can be roughly broken down into 4 main components:

- Flat regions: These can be defined as homogeneous areas, where pixels are merely sharing a similar intensity value. (Figure 2.6, Top left).
- Edges: These can be simply defined as one direction delimitation between two blocks of different pixel intensities. It is represented as a line. (Figure 2.6, Top right).
- Corners: These are usually formed by the intersection of two edges and represent the most precise way of defining local features within an image due to their singularity. (Figure 2.6, Bottom left)
- Blobs: These are a particular pattern that differs from its neighbourhood. It is neither a corner nor an edge. (Figure 2.6, Bottom right)

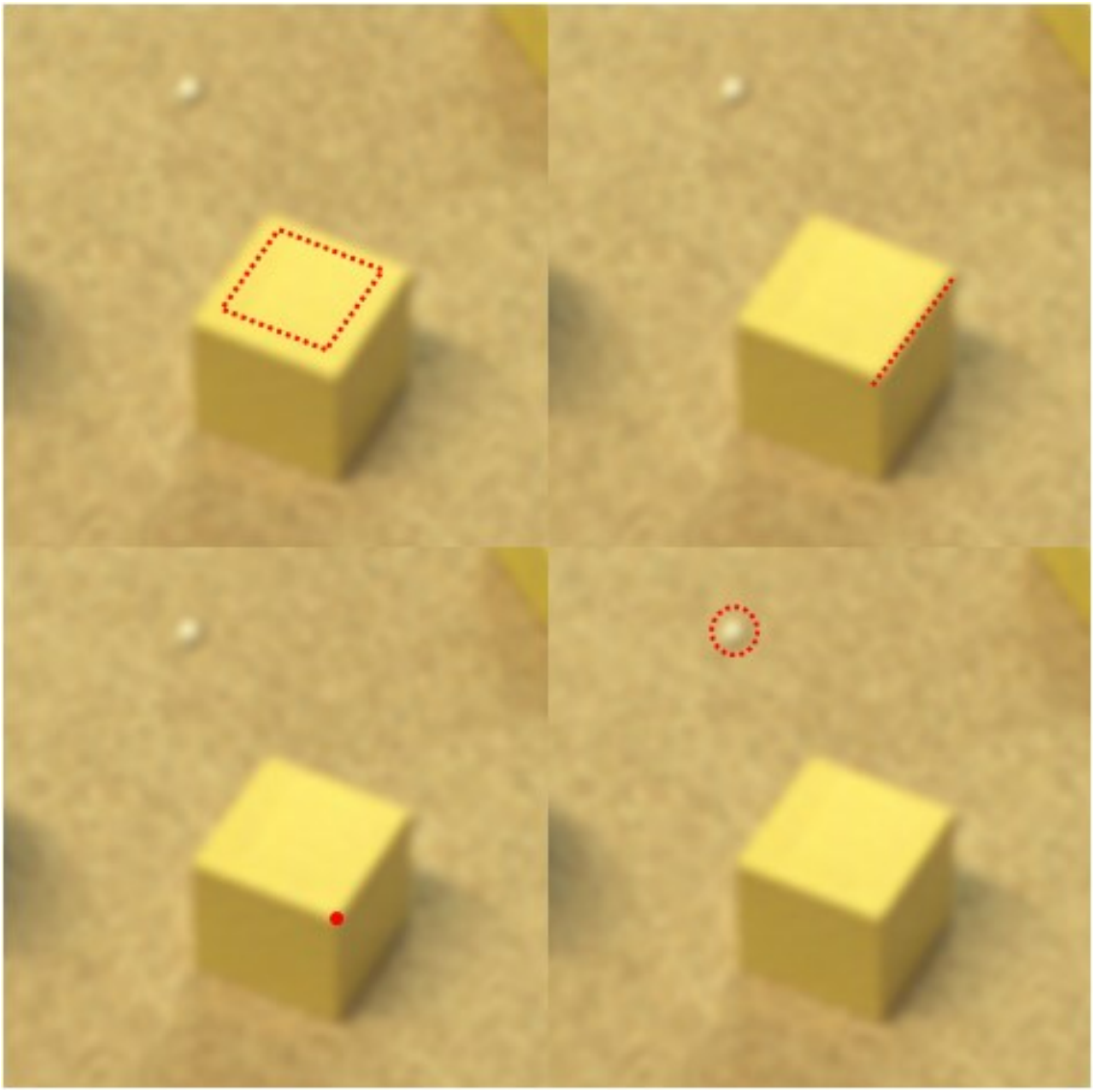


Figure 2.6. *Illustration of the row content of an image: flat region (top left), edge (top right), corner (bottom left), and blob (bottom right).*

Good feature detection methods are evaluated according to their ability to answer best to the following requirements: localisation accuracy, repeatability, robustness, computational efficiency, and distinctiveness. With this in mind, one can easily understand that corners, and to a less extent blobs, are the most suitable patterns for detection. A large number of feature detection techniques have been proposed. These are divided into two categories: blob detectors (including SIFT [17], SURF [18], and

CENSURE [19]) and corner detectors (including Harris [20], Shi-Tomasi [21], and FAST [22]).

Blob detectors provide a high level of distinctiveness resulting from a throughout extraction of the features neighbourhood which implies a subsequent computational cost. On the other hand, corner detectors are less distinctive but much faster. For more details, [23] gives a good overview of the mentioned detectors. Most of these techniques have been tested during and at different stages of the PhD project.

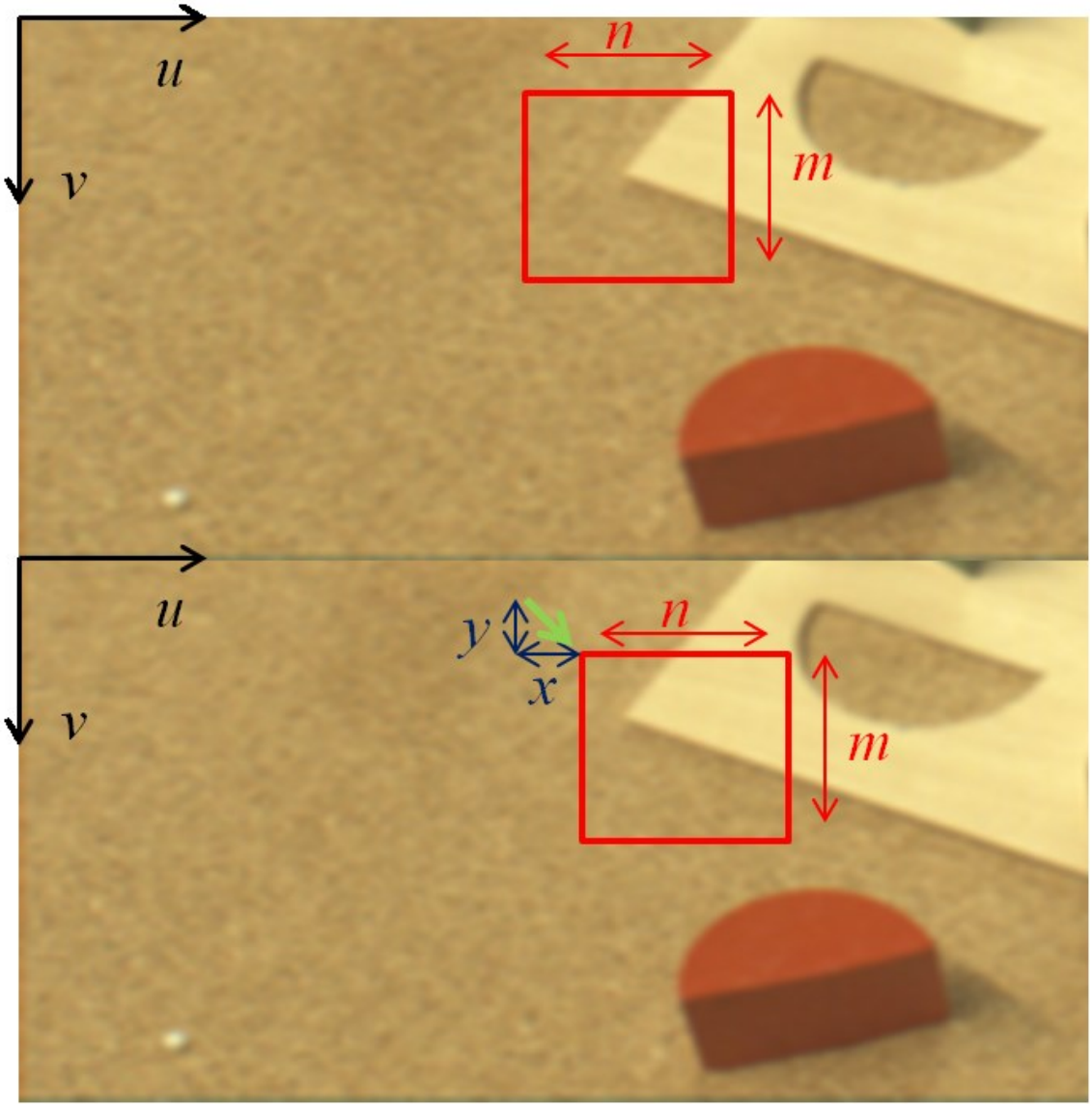


Figure 2.7. *Illustration of corner detection process: original patch (top), shifted window (bottom).*

Each of them presented positive aspects but also negative aspects. Taking into account the real time constraint required for this project, efficient solution was favoured on the detriment of high degree of distinctiveness. For this project, we finally decided to adopt Good Feature To Track detector (GFTT) from Shi-Tomasi [21]. This technique is an improvement of the Harris and Stephens corner detector method [20]. These two techniques are based on the analysis of the second derivative matrix *M* describing the gradient distribution of a local neighbourhood ($n \times m$ local window). Considering an image patch centred at (*u, v*) and shifted by (*x,y*) (see Figure 2.7), the weighted *sum of squared differences* (SSD) between these two patches, is denoted *D* and is defined as follows:

$$D(x,y) = \sum_{u=0}^{n} \sum_{v=0}^{m} [I(u,v) - I(u+x, v+y)]^2 \tag{2.8}$$

The second term in (2.8) represents the shifted window that can be re-written after approximation by a first order Taylor expansion as follows:

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y \tag{2.9}$$

Where $[I_x, I_y]^T$ is the spatial gradient of the image *I*. Giving this approximation, (2.8) can be expressed as follows:

$$D(x,y) \approx [x \quad y] M \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.10}$$

with

$$M = \begin{bmatrix} \sum_{u,v} I_x^2 & \sum_{u,v} I_x I_y \\ \sum_{u,v} I_x I_y & \sum_{u,v} I_y^2 \end{bmatrix} \tag{2.11}$$

Harris [20] and Shi-Tomasi [21] differ in the way to assess the corners according to eigenvalues (λ_1 and λ_2) of the matrix *M*. A scoring parameter *R* is introduced to highlight the two assessments criteria as follows:

For Harris:

$$R = \det(M) - k \times trace(M)^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2) \tag{2.12}$$

$k = 0.04$ in general

For Shi Tomasi:

$$R = min(\lambda_1, \lambda_2) \tag{2.13}$$

The difference between the two scoring parameters may not seem very significant but in practice Shi-Tomasi approach gives a higher number of detected features. Additionally, with (2.13) each of the image components is mathematically well segmented recalling the 3 main row image components (cited above) following this classification:

- Flat regions are characterised with two small eigenvalues (λ_1 and λ_2), which means that there is no significant variation of the gradient in both directions of the window displacement.
- Edges are characterised with only one large eigenvalue (λ_1 or λ_2) describing a notable variation of the gradient only in one direction (x or y, when the window is moving perpendicularly to the edge).
- Corners present two large eigenvalues (λ_1 and λ_2) describing a significant variation of the gradient in both directions (x and y) whatever the direction the window is moving in.

Figure 2.8 illustrates well the reason that motivated us to adopt Shi-Tomasi approach. Harris approach is too restrictive as criterion (2.12) limits the number of detected features.

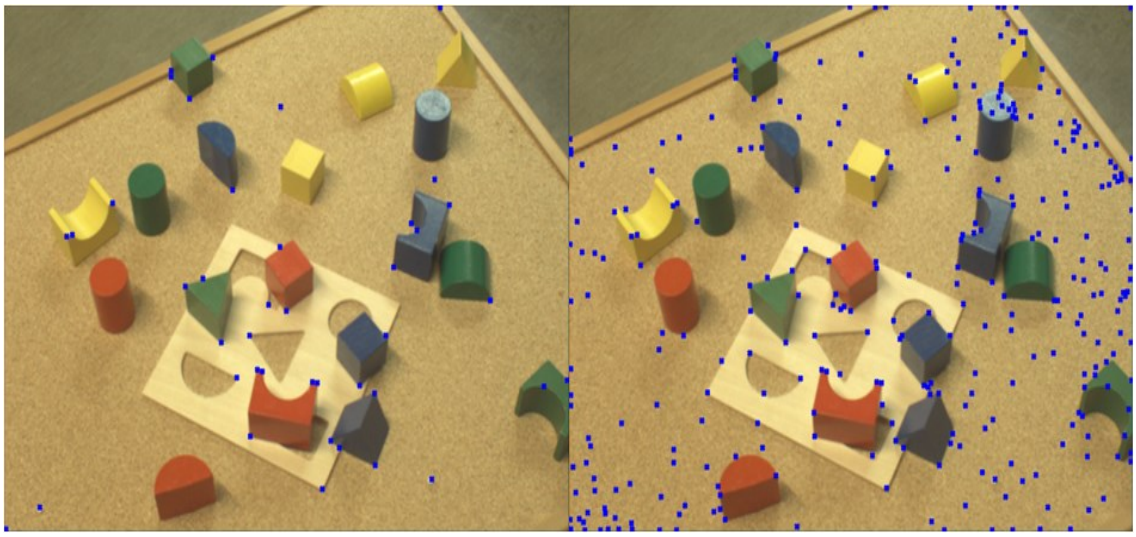


Figure 2.8. *Illustration of feature detection on a puzzle image: Harris (left), Shi-Tomasi (right).*

2.7 Stereo and Feature Tracking

According to the visual odometry context of this thesis, operations involved to find correspondences concern a set of four images. It consists of the previous left I_{pL} and right I_{pR} constituting the first stereo pair and the current left I_{cL} and right I_{cR} constituting the second current stereo pair (Figure 2.9). In Figure 2.9, R and t represent the inter-frame motion, which is aimed to be found with the visual odometry process. Correspondences search approaches use feature detection on each image of the inter-frame set. Then, various matching techniques are applied horizontally (respectively between I_{pL} and I_{pR} and between I_{cL} and I_{cR}) and vertically (respectively between I_{pL} and I_{cL} and between I_{pR} and I_{cR}). Feature matching is generally divided into template based and descriptor based techniques.

The simplest method to find the right correspondences between different images is to compare the entire set of feature template or descriptors related to an image with all the other feature template of descriptors of the other image.

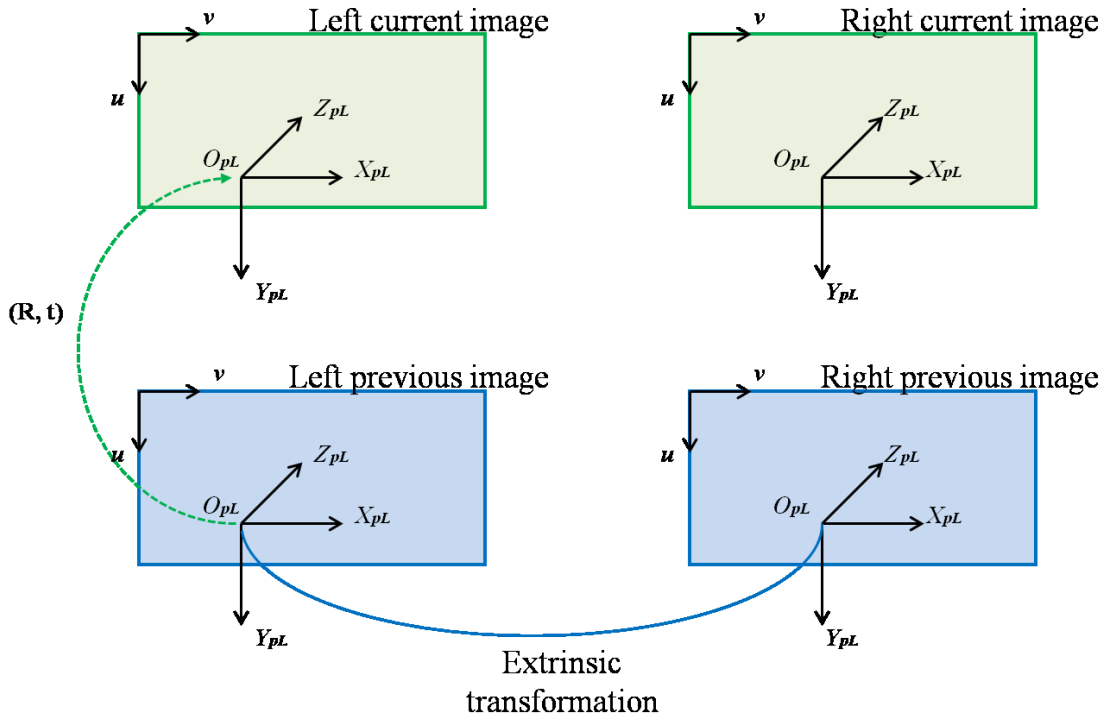


Figure 2.9. *Illustration of visual odometry image system.*

An additional, mutual consistency check can be even employed to robust the matching process. This consists in leading feature matching in both ways (i.e. I_{pL} to I_{pR} then I_{pR} to I_{pL}) and keep matches that were successful in both operations. Feature matching is a robust approach to find correct correspondences between two images. However, it has a significant computational cost that is quadratic to the number of features. In the context of stereo visual odometry, targeting real-time processing while having 4 input images as inputs to process makes the implementation of feature matching difficult or this will be at the detriment of other operations.

An alternative way to feature matching in order to tackle correspondence search is feature-tracking. Starting from a detected set of features in a first image, feature tracking search for their corresponding matches in a second image. Tracking features in an image involves only one feature detection operation (in the first image). However, one important assumption of this approach is that consecutive images have to present only small to reasonable change in appearance in order to work well. There exists a feature tracking method, which deals with subsequent appearance changes until a certain extent. This technique is the Kanade-Lucas-Tomasi (KLT) tracker [24] and uses an affine distortion model for each feature. The chosen strategy to solve the correspondences problem is based on feature tracking and the use of KLT feature tracker. The KLT tracker is defined as a nonlinear optimisation problem that aims to minimise the squared sum of the intensity difference e between two successive images I_p and I_c. This is led over a small patch of size $(2w_x+1)(2w_y+1)$ centred respectively at the position $x = [u, v]^T$ and the tracking motion model $w(x;p)$:

$$e = \sum_{-w_x}^{w_x} \sum_{-w_y}^{w_y} \left[I_p(\mathrm{x}) - I_c(\mathrm{w}(\mathrm{x};\mathrm{p})) \right]^2 \tag{2.14}$$

where w_x and w_y are two integers usually set to 7,8,10,… 21 according to [24]. The local patches inner pixels are used to create an over constrained system. The tracking motion model, also called *warping function* $w(x;p)$, is composed of $\mathrm{x} = [u, v]^T$ a pixel coordinate and p a vector of warping parameters. The KLT starts with an initial value of p, and iteratively searches for the δp that aligns the two image patches such that (2.14) is minimised. The warping functions can be written following two different models. First, the translational model expressed as:

$$w(\mathrm{x};\mathrm{p}) = w(\mathrm{x}+\mathrm{b}) \tag{2.15}$$

The second one is the affine model expressed as:

$$w(\mathrm{x};\mathrm{p}) = w(\mathrm{Ax}+\mathrm{b}) \tag{2.16}$$

where $\mathrm{A} = \begin{bmatrix} 1+\eta_{xx} & \eta_{yx} \\ \eta_{xy} & 1+\eta_{yy} \end{bmatrix}$ and $\mathrm{b} = \begin{bmatrix} \eta_x \\ \eta_y \end{bmatrix}$

The translational model has the advantage to be fast. However, it is only reliable if the appearance change between subsequent images remains small. Initialising p with a starting value close enough to its target enables translational-based KLT to cope with large optical flows.

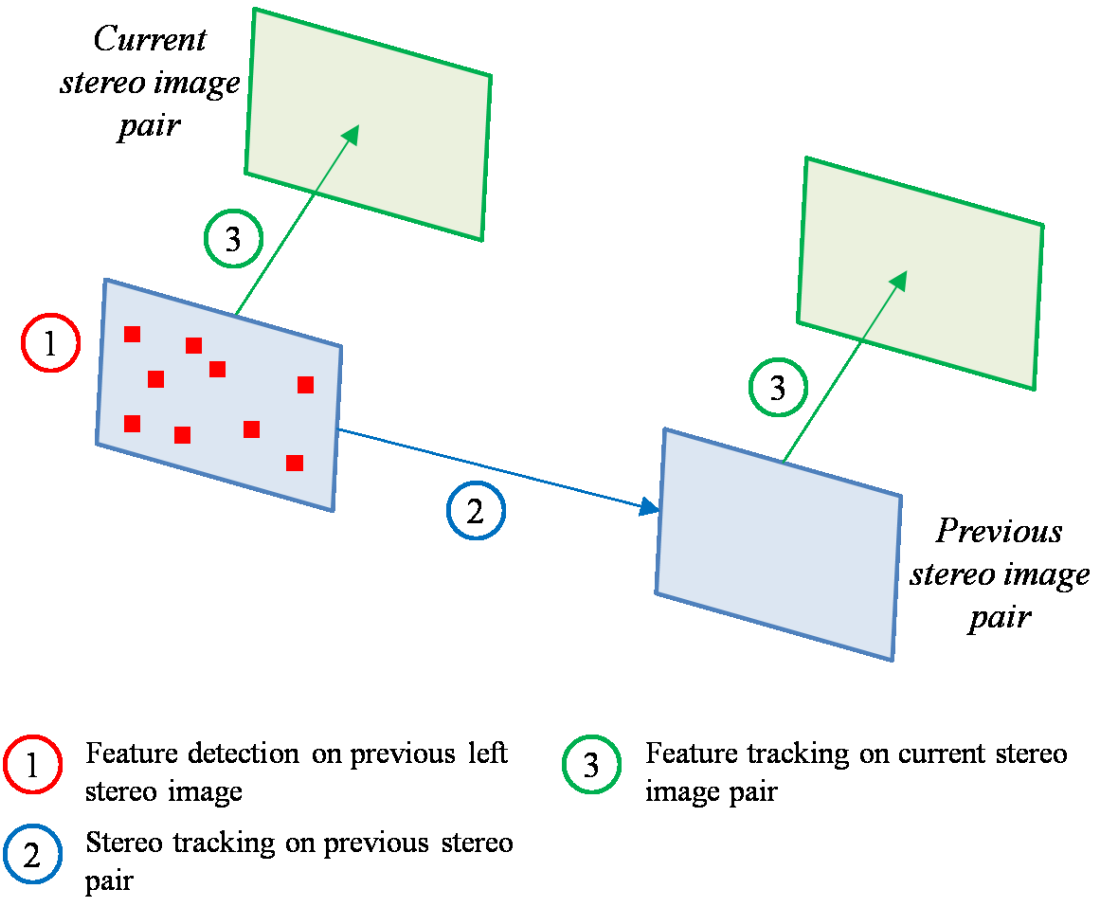


Figure 2.10. *Illustration of the proposed correspondences search strategy based on stereo and feature tracking.*

On the other hand, the affine model (or also called *pyramidal* KLT) gives more options to deal with spatial deformation of the patches. However, it is computationally more expensive than the translational model.

Figure 2.10 illustrates the proposed correspondence search process. Instead of detecting features in both previous and current images, we only detect features on the previous left image. This enables saving computation time that would have been taken if feature detection was carried out on the remaining image set. From this set of detected feature, stereo tracking finds correspondences on the previous right left image.

As the input images are rectified, the epipolar constraint is applied here. The correspondences that do not share the same horizontal image coordinate or present an unreasonable disparity are discarded. From the remaining correspondences of the previous stereo pair, a temporal feature tracking is carried out between left previous and current left images, and between previous and current right images respectively. The epipolar constraint is applied again on correspondences independently found on the current stereo pair. This process ends with a set of filtered correspondences linking the 4 images of two subsequent pairs. Further filtering also called outliers rejection can be used to improve the quality of these correspondences. This process is treated later in Chapter 4. Finally, it has been decided of not tracking over more than two subsequent stereo image pairs. Hence, for each new stereo image a new set of detected features are extracted from the previous left image in order to avoid drift in image feature localization.

2.8 Stereo Visual Odometry Pipeline

This Section introduces the full stereo visual odometry pipeline adopted in the work presented in this thesis. The proposed visual odometry algorithm takes as input intrinsic and extrinsic calibration parameters and four grayscale rectified images representing the previous and the current stereo pairs. Detection is carried out on the previous left image, and stereo tracking gives the correspondences with previous right image. From these correspondences temporal feature tracking is independently run between the previous and current left images and between previous and current right images. The resulting set of correspondences linking the four images serves as input for the motion estimation process. The latter which is also associated with an outliers rejection scheme, gives the

relative inter-frame estimation of motion between the previous and current image pairs. Relative inter-frame motions are accumulated along the camera equipped platform travel. This results in the generation of a 6 DOF trajectory that estimates at best the travelled route. Figure 2.11 illustrates the complete visual odometry pipeline.

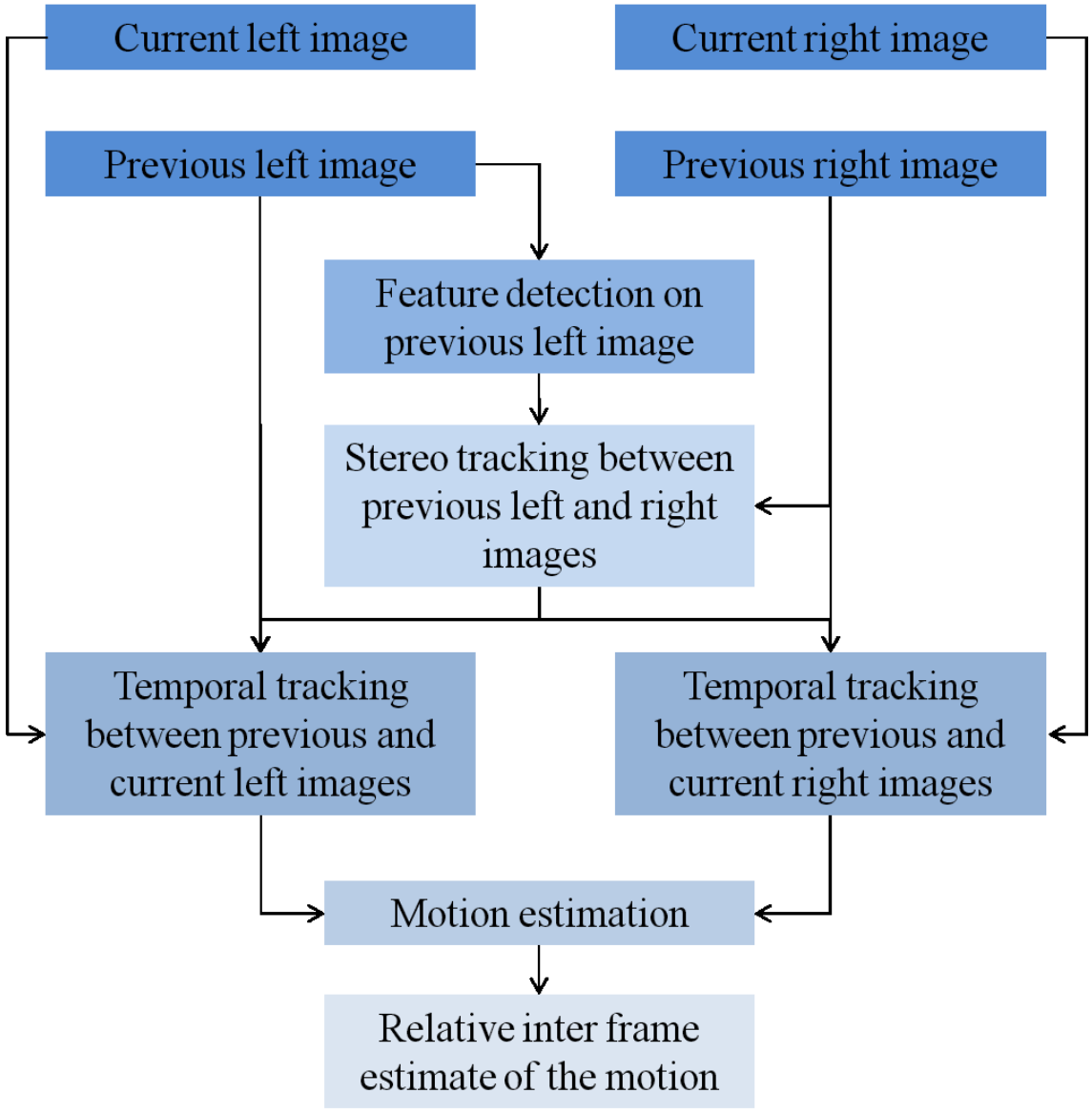


Figure 2.11. *Illustration of the full stereo visual odometry pipeline.*

2.9 Chapter Conclusion

In this chapter, basic knowledge as well as some techniques that compose the visual odometry pipeline have been presented. After describing some principles of image representation, the pinhole camera model giving that relates real world objects and their image plane representation was introduced. After that, the stereo vision properties and how to obtain rectified stereo image pair were explained. Feature detection in general and details of the chosen Shi-Tomasi feature detector technique were given. The correspondence search strategy to link the previous and current stereo pairs was described as well as the KLT feature tracking technique chosen for this purpose. The full stereo visual odometry pipeline adopted in this thesis which leads to platform trajectory estimation was finally depicted.

3 HDR Imaging: Feature Detection and Tracking Application

3.1 Overview

In visual odometry applications, the quality of acquired images is essential to extract meaningful information, which will be analysed by different consecutive processes enable navigation systems to achieve successfully the goal they have been designed for. As we explain in Chapter 2, images are a digital representation of the real world environment surrounding the camera in terms of a set of brightness intensity values associated to elements of grid of pixel locations. Thus, contents of a scene, which is composed of static and dynamic objects form the essential source of information. In navigation system applications, camera motion is estimated from the analysis of correspondences between sets of detected high level features on a sequence of images. Feature tracking is the operation that enables us to find correspondences between these sets of features. Several techniques have been developed to achieve this operation as mentioned in the previous chapter. Among them, optical flow approaches such as Kanade-Lucas and Horn–Schunck methods. Also other techniques such as phase correlation, cross correlation or sum of the absolute differences are used. The objective of feature tracking is to estimate the pixel motion from the apparent change between images. Successfully tracked features associate the same static objects of the scene between subsequent images are called inliers.

On the other hand incorrect correspondences between static object(s) and features belonging to dynamic object(s) are considered as biased information, and are usually called outliers. However, outliers are not the main issue, as there are several efficient outlier removal techniques that have been developed. In fact, one of the major concerns lies in the ability to extract these high level features (corners), which strongly depend on the clarity of acquired images [25], [26].

An over or under exposure might significantly degrade the digital representation of the scene, resulting in a failure of the front end feature detection operation (Figure 3.1). Consequently, the rest of the navigation process is likely to deteriorate especially if it lasts for too long. The reason of such sensitivity towards illumination conditions comes from the limitation in the dynamic range for most image sensors used in these kinds of applications. The majority of them have an 8 bit integer single channel representation, which corresponds to 256 shades per pixel whereas the human eye can distinguish 10 million different colours and real scenes vary in about 10 orders of magnitude [27]. As a consequence, for extreme illumination conditions, details of the scene are either lost in the noise for darker regions or lost in saturation for brighter regions. Both phenomena can even be encountered together especially in backlighting cases such as indoor/outdoor passage or when incident light penetrates inside a building through one or several windows. This results in high illumination contrast with very dark and very bright areas.

Figure 3.1. *Example of pixel saturation due to an under (left) and over (right) exposure with a none HDR image sensor.*

In these conditions, an important part of the information in the image is lost which makes the extraction of any relevant features extremely difficult (Figure 3.1).

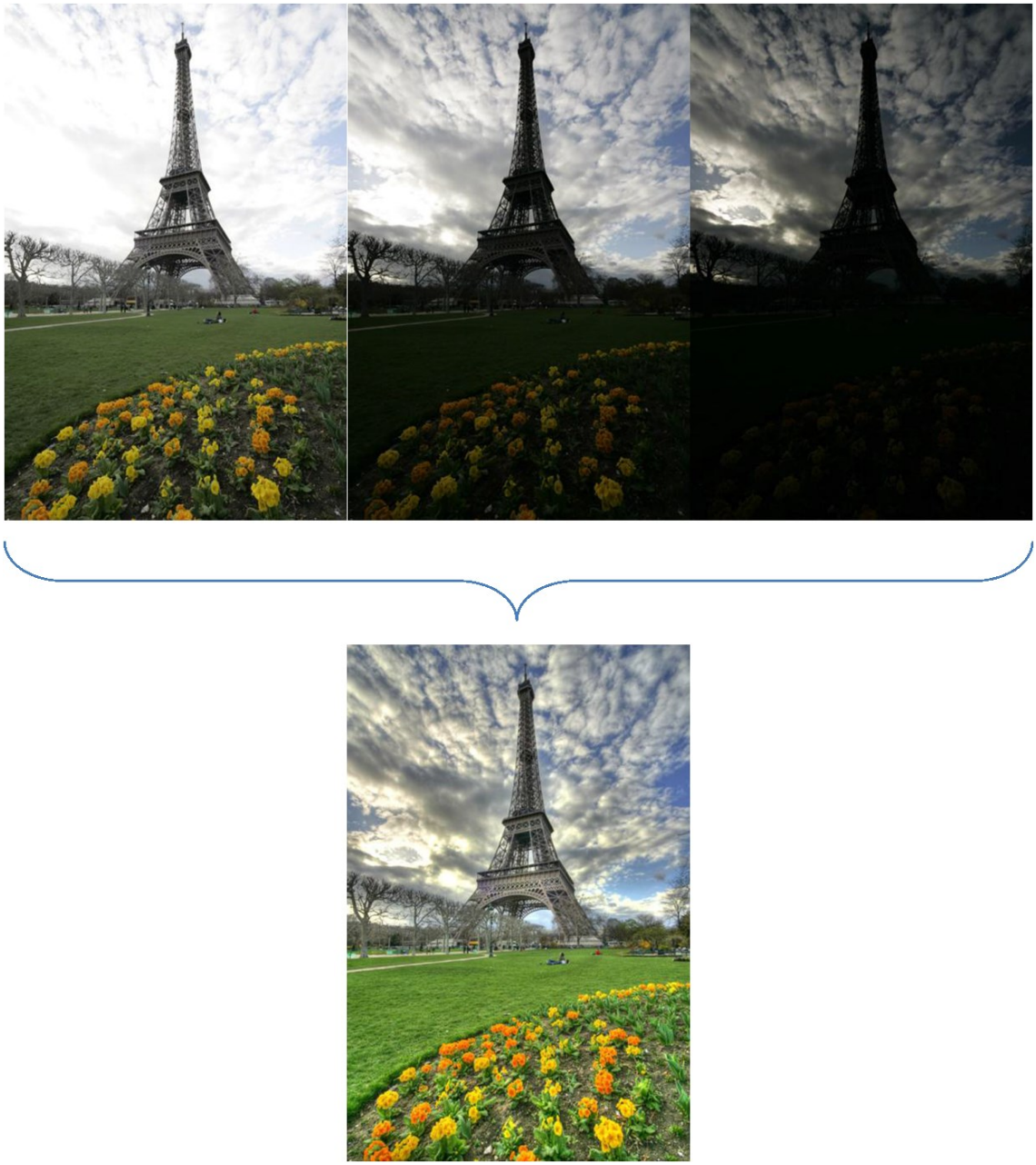


Figure 3.2. *Illustration of HDR image generation with multiple exposure images (top) and tone mapping image (bottom) [109].*

HDR imaging extends the limit of the luminance range of a standard digital camera. It can reach a maximum of 32 bits floating-point single channel representation, which corresponds to 4.29 billion shades per pixel. This corresponds to 9 orders of magnitude. Hence, it is one magnitude unit less than for real world scenes. As a result, extreme illumination is no longer a problem as dedicated space per bits enables capturing higher dynamic range than the eye can discern giving a realistic rendering. This still remains true even after compression operation to convert HDR images to displayable formats (png, bmp, tiff etc...). This process, called tone mapping [28], uses richness of intensity information stored in an HDR image to create a compressed image where all the pixels get the correct exposure (Figure 3.2).

In visual applications such as visual navigation systems, imaging holds an essential role as it bring input system information, and feature detection and tracking contribute to the analysis of the motion through images. The work in this chapter contributes by offering to analyse HDR image sensor and show by using it a way to reduce the limitations of visual navigation systems in extreme illumination conditions. This is essential, because problems of over and under exposure due to changes in lighting conditions are common in machine vision applications**.**

3.2 HDR Imaging

Generation of an HDR image is based on a combination of multiple image captures of the same scene subject to different exposures [29]. Several techniques have been proposed to generate HDR images[30]–[32]. The basic ones use a set of captures with different exposure of the same scene acquired with a single static camera. Later, in order to get rid of constraining static acquisition required to generate HDR images, sensors with techniques using varying pixel exposures were introduced [33]. Then, a video approach using the successive multiple exposed frames has been investigated to produce HDR images [34].

Tone mapping is another important field of HDR imaging that has been extensively investigated in order to improve the rendering quality of displayable format images. It can roughly be divided into two categories: global and local. Global approaches apply the same function for the whole image and are computationally efficient, but not really

effective for extremely dark or bright areas [35], [36]. On the other hand, local techniques are more demanding but much more accurate as each pixel is processed according to the surrounding pixels intensity. Local approaches use a broad range of techniques such as block-based [37], gradient based [38], frequency domain [39], histogram equalization based [40], and even machine learning methods [41] fusing the multiple exposure images in order to generate high quality compressed images for display. A lot of techniques extended this multiple exposure process to stereo cameras by applying stereo matching to obtain a dense disparity map of an acquired scene [42], and also to generate HDR images [43]. Medical imaging also benefited from microscopy HDR imaging [44] to enhance the level of detail for disease diagnostics, or cells specimen analysis.

3.3 Feature Detection with HDR Imaging

Although HDR imaging has become increasingly popular in recent years, only a few works investigate feature detection over HDR images. In [45] a local illumination invariant tone mapping to HDR images is applied, enabling SIFT algorithm to perform better than for traditional HDR images. Feature detection performances comparatively led on HDR and standard images also called Low Dynamic Range (LDR) have also been studied. Pribyl et al. [46] compared LDR images with global and local tone mapped generated HDR images. Images were acquired in a dark room lit with two reflectors to create a high contrast scene and to assess the repeatability of the different feature detection techniques. Globally it has been found that HDR imaging increases repeatability rate of the detector as compared to LDR images. Our recent study uses a larger range of feature detection techniques (SURF, SIFT, Harris, Shi-Tomasi and FAST) as well as feature matching based on descriptors (SURF and SIFT) to compare HDR and LDR images on extreme illumination scenarios [47]. In this work we showed several times higher detection and matching rates on subsequent HDR images compared to subsequent LDR images through different scenes presenting extreme illumination. Additionally, and to the best of our knowledge, there has been no similar work assessing the matching performance comparatively between HDR and LDR images.

3.4 Feature Detection and Tracking Algorithm

The four selected feature detection methods have been chosen from those most used in computer vision [48]. They consist of: SIFT, SURF, GFTT and FAST. Our feature tracking evaluation algorithm selects between these four feature detectors. Notably, SURF and SIFT descriptor properties are not used in this test as feature tracking is not descriptor based (use of KLT). Feature detection is led on the first frame of an image sequence. The detected features are then filtered according to their cornerness response. All the features with a response below a minimum response threshold are discarded. Then Pyramidal KLT feature tracker is run consecutively between the frames composing the image sequence, starting from the set of pre-filtered features. If a tracked feature remains inside a certain radius that is spatially constrained, then tracking the feature is validated. Otherwise, it is discarded. This algorithm enables us to assess the tracking persistence of the KLT. This, according to the image quality provided by an image sensor, is described below.

3.5 Experiments and Results

Datasets consist of four scenarios presenting extreme illumination conditions. The first one took place in an outdoor environment with a direct sun exposure. The second, the third and the fourth took place indoor, respectively facing a window bathed in sunlight, in a dim lit office, and in a dark room. The four feature detectors mentioned earlier are employed to extract key-points. These key-points are then input into the pyramidal KLT algorithm for the tracking operation. In this study we use 3 level pyramids. The algorithm was coded in C++ using opencv library [103] to call the feature detection techniques and the pyramidal KLT method. The material used in these tests consists of an Aptina demo kit and Camelot USB 2 camera from Imagine2d Ltd. The first one is the HDR device based on the Aptina MT9M024 HDR CMOS image sensor with a resolution of 1.2 Megapixel (1280x964). The second one is based on the Aptina MT9P031 CMOS image sensor with a resolution of 5 Megapixel (2592x1944). For more clarity, High Dynamic Range and 5 Megapixel devices will respectively take the abbreviation Aptina HDR and Camelot 5MP.

Algorithm 3.1: *Feature tracking persistence evaluation pseudo code*

Initialise: *N, minfeature, minradius, responselevel*

 # N: number of tracked features

 # minfeature: threshold for minimum tracked features

 # minradius: threshold for minimum tracked feature range location

 # responselevel: threshold for minimum cornerness strength

 # start feature tracking algorithm

while(tracking)

if(N < *minfeature*)

 # run FAST | GFTT | SIFT | SURF feature detector

N = detectfeaure();

 # discard features with a low cornerness response

for i = 1, ..., N

if($p_{p(i)}$ < *responselevel*)

remove $p_{p(i)}$ from N

endif

endfor

endif

 # run the pyramidal KLT on the set of points N

{ $p_{p(i)}$), $p_{c(i)}$ } = pyramidalKLT();

for i = 1, ..., N

if(matched and $p_{p(i)}$ - $p_{c(i)\ i}$ || < *minradius*)

#successful tracking

else *# lost feature*

remove $p_{p(i)}$ from N

endif

endfor

endwhile

Figure 3.3. *Tripod carrying the stereo rig with the two image sensors: left Camelot 5MP, right Aptina HDR.*

Standard cameras usually have a pixel dynamic range of 60dB which means that the image sensor can acquire a contrast of 1:1000 dynamic range intensities. The Camelot 5 Megapixel image sensor extends this amplitude ratio by approximately a factor of three with a pixel dynamic range of 71 dB (1:3199). With a pixel dynamic range of 120dB, the Aptina HDR can capture a contrast of 1:1000000. This is made possible with sequential captures of three exposures by the Aptina demo kit HDR image sensor. These three exposures are then combined to create a linearized 20-bit value for each pixel's response. This 20-bit value is then optionally compressed back to a 12 bit value for output. According to the Aptina HDR image sensor datasheet, 12-bit compression is only affected with a minimal data loss. For our experiments selected 12 bit compression mode. These two image sensors were mounted and bolted close one to each other on the centre of a stereo rig (Figure 3.3). This stereo rig was mounted on a tripod and was hand-carried during the tests so that the image sequences were taken at the same time by the two image sensors.

The different image sequences involve various translations and rotations in order to judge the tracking performance of each device. The number of detected features and tracking robustness will be assessed concurrently. That is the ability of maintaining a reasonable percentage of initial detected features during all the image sequences. The Camelot 5 MP image sensor output images are resized to 1296x972 to match the Aptina HDR images full resolution.

3.5.1 Scene 1: Outdoor Direct Sun Exposure

The first scene takes place outside facing the sunlight. The tripod carrying the two image sensors is moved forward with a slight rotation at the end of this sequence. In this case, illumination conditions are very challenging due to direct sunlight facing the cameras. Indeed, this over exposure has a direct impact on image rendering with pixel saturation.

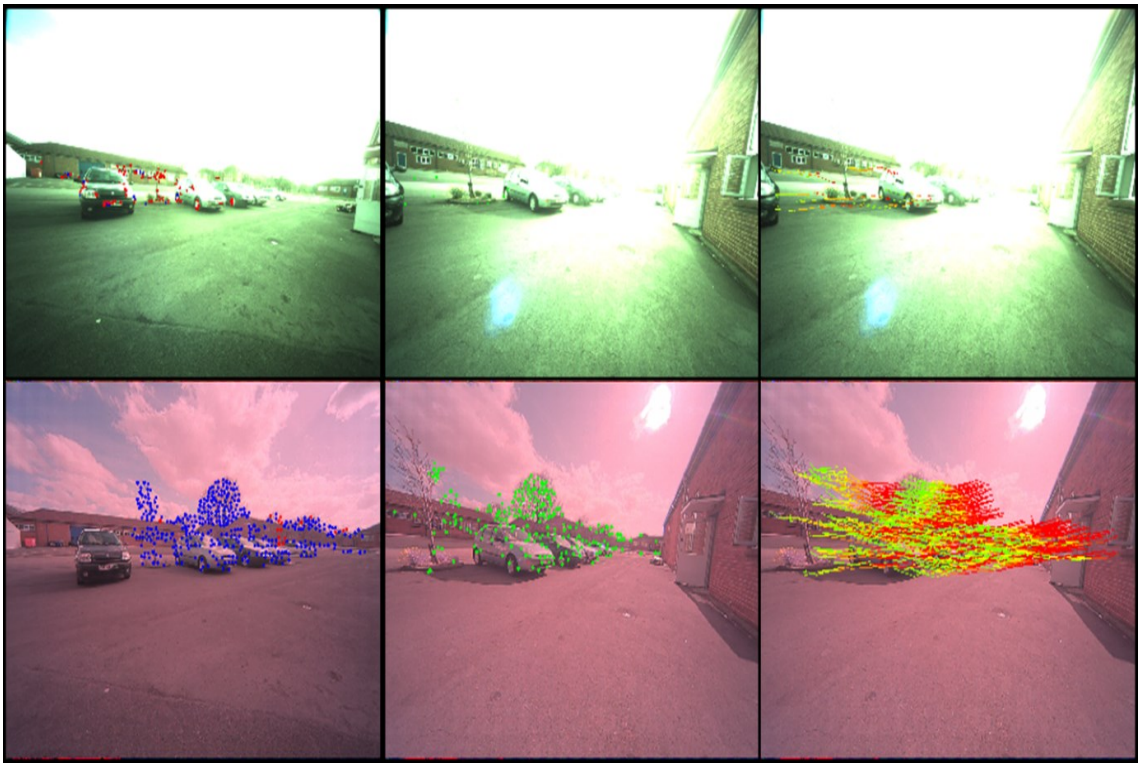


Figure 3.4. *Scene 1 using GFTT feature detector - Top: Camelot 5MP - Bottom: Aptina HDR - From the left to the right: first image, initial tracked features (red: lost, blue tracked); middle image final tracked features (green points); final image with full optical flow (colour shading).*

This is especially true for images acquired with the Camelot 5 MP camera. In Figure 3.4 we can see that the high reflection of the sunlight on the scene has the effect of brightening up the images captured with the Camelot 5 MP. Consequently, only the close surroundings of the camera are clearly distinguishable. Notably, the sky which is significantly present in this sequence accentuates this effect. On the other hand, the HDR images from the Aptina demo kit give a normal rendering of the scene with a matt effect despite direct sun exposure.

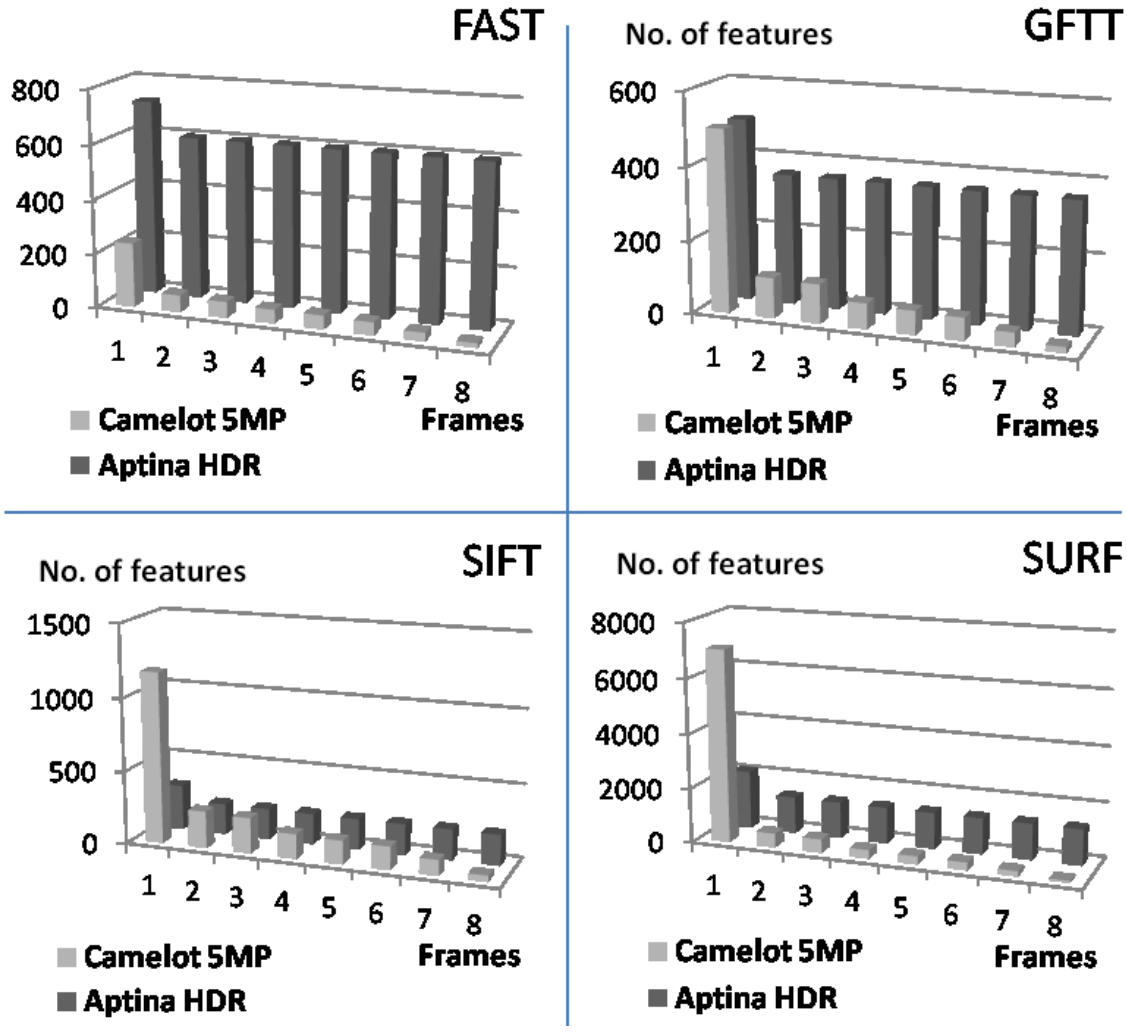


Figure 3.5. *Scene 1: Tracking frame length histogram graphs of Camelot 5MP and Aptina HDR image sequences.*

Indeed, all the contents of the scene, even the clouds are clearly distinguishable. In term of feature detection and tracking performance Camelot's sequence is clearly disadvantaged by the very high illumination of the scene.

Figure 3.5 illustrates tracking performance of the algorithm between the two cameras devices. For the four feature detection techniques we observe the same trend resulting in a continuous decrease in the number of tracked features.

Indeed, as reported in Table 3.1, at the end of the sequence a minimum of a eighth (with SURF) and a maximum of a quarter (with FAST) of initial tracked features are maintained. On the other hand, the algorithm run on Aptina HDR demo kit image sequence enables us to maintain the quasi totality of initially tracked features.

Table 3.1. *Scene 1: Tracking Performance Summary.*

		Detected Features	Initially Tracked	Finally Tracked	
				No.	*%*
FAST	*Camelot 5MP*	237	62	16	25.8
	Aptina HDR	721	599	599	100
GFTT	*Camelot 5MP*	500	110	16	14.5
	Aptina HDR	500	358	358	100
SIFT	*Camelot 5MP*	1173	252	41	16.2
	Aptina HDR	308	210	208	99
SURF	*Camelot 5MP*	7035	506	64	12.6
	Aptina HDR	2132	1335	1308	97.9

Consequently, we have a much higher number of potential inliers on HDR image sequence. Note that, the excessive drop between the number of detected and initially tracked features in Camelot image sequence results from the sensitivity of the feature detector to high illumination. Thus, most of the detected features are irrelevant artefacts localised in majority on the ground.

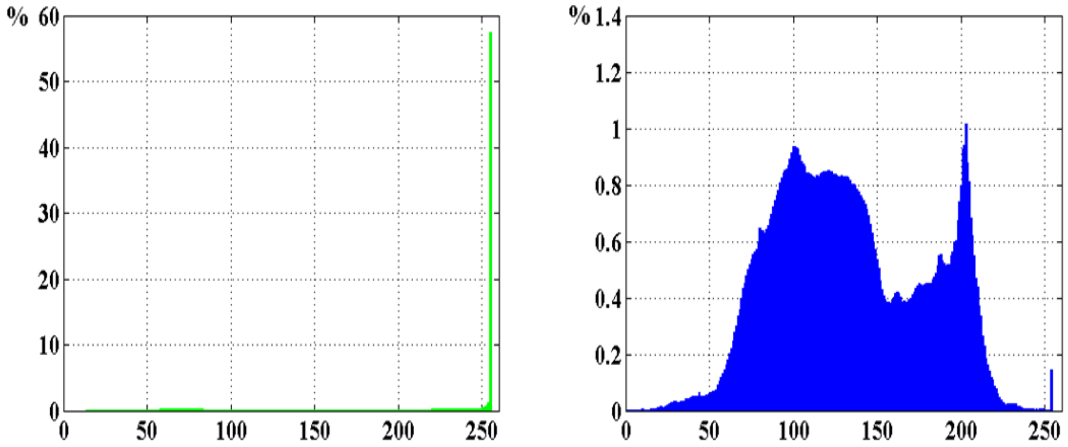


Figure 3.6. *Scene 1: Average greyscale histogram of image sequence Left: Camelot 5MP - Right: Aptina HDR (percentage of pixels/intensity).*

Table 3.2. *Scene 1: Pixel Density Statistics of Average Greyscale Histogram of Image Sequences.*

	Dense interval(s) (>1%)	Pixels per interval(s)		
		%	Mean	Std
Camelot 5MP	[255]	57.48	57.48	-
Apitna HDR	-	-	-	-

In Figure 3.6, an analysis of the average greyscale histogram of the two image sequences gives more information regarding the great difference of results in feature tracking between the two image sensors. Indeed, as reported in Table 3.2, almost 60 % of the pixels of Camelot 5 MP images sequence are unusable because of saturation. In these conditions, the feature tracking operation is obviously really difficult. The average greyscale histogram of Aptina HDR images sequence confirms the visual rendering with a homogenous spread of pixels over pixel intensity bins. It can be seen that only a very small amount of pixels are located on the extremes of the pixel intensity axis despite the very high illumination of scene 1.

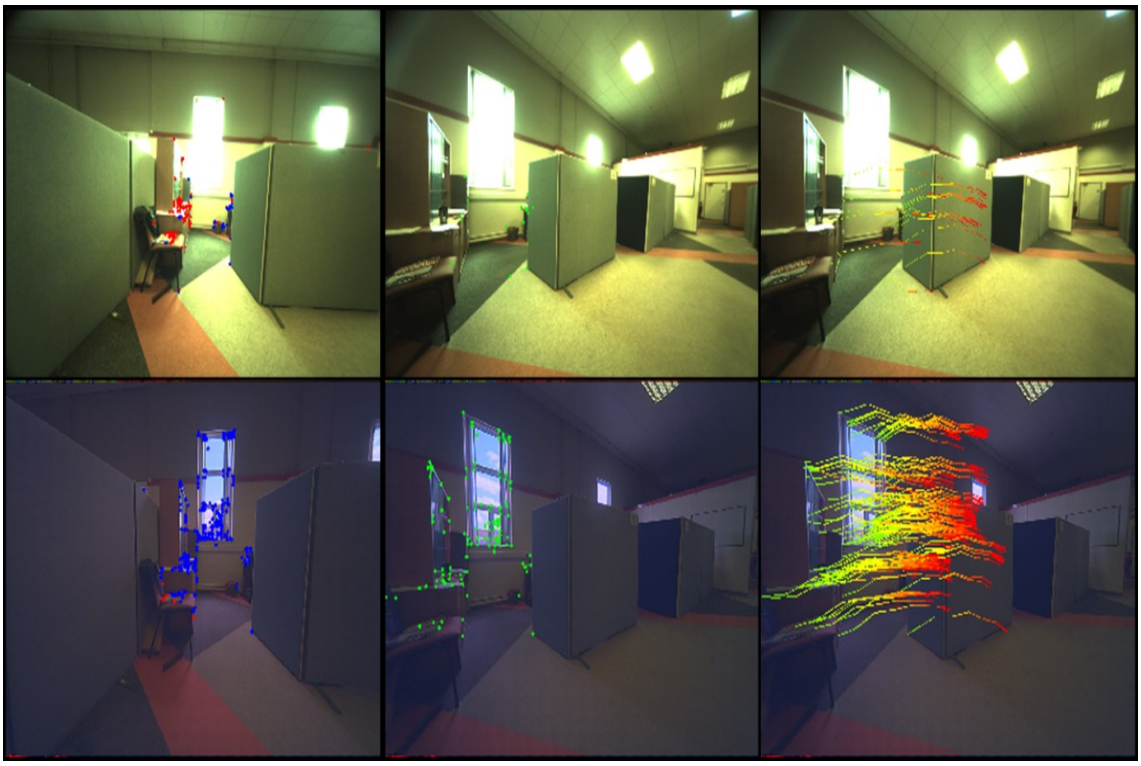


Figure 3.7. *Scene 2 using FAST feature detector - Top: Camelot 5MP - Bottom: Aptina HDR - From the left to the right: first image, initial tracked features (red: lost, blue tracked); middle image final tracked features (green points); final image with full optical flow (colour shading).*

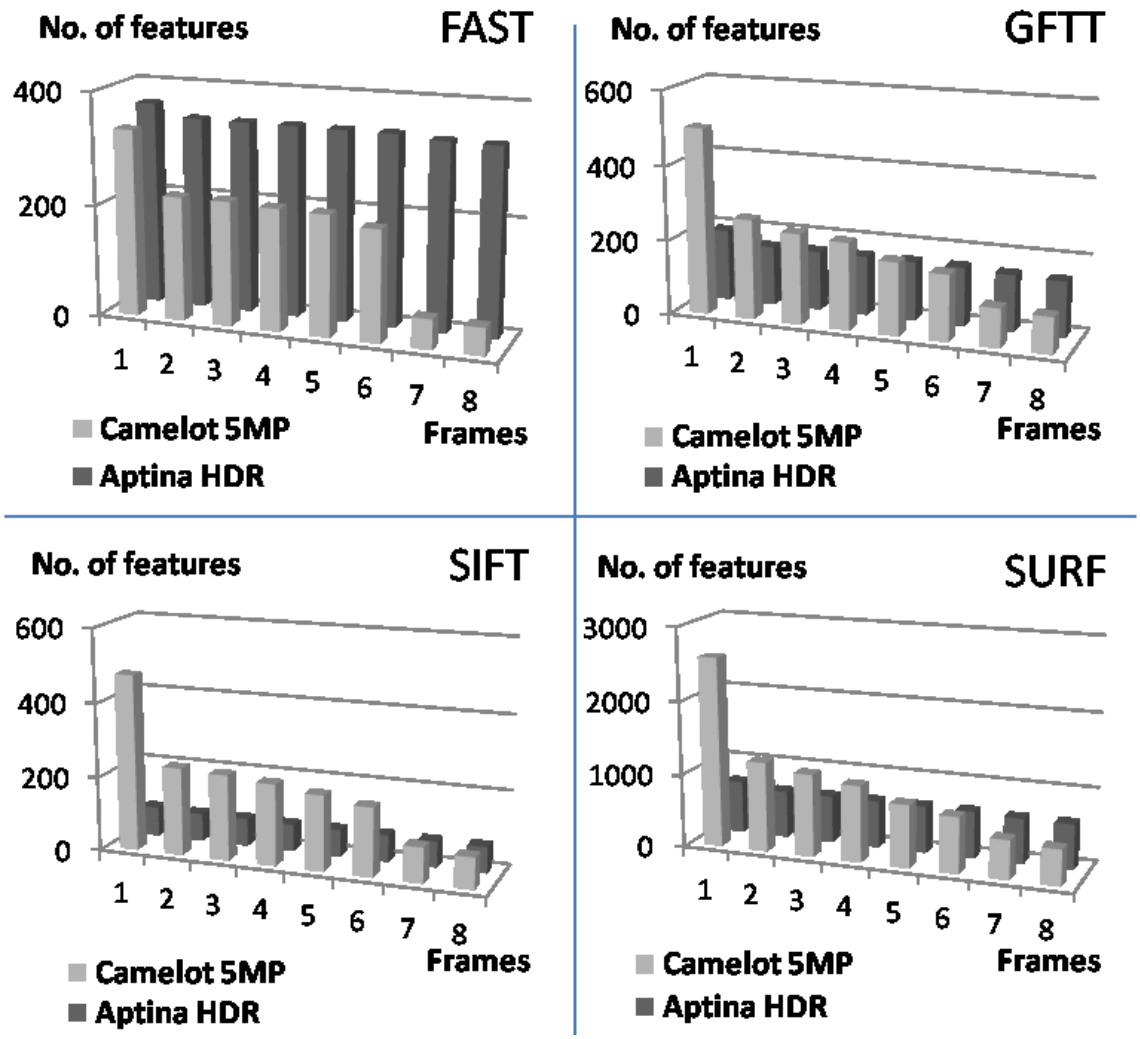


Figure 3.8. *Scene 2: Tracking frame length histogram graphs of Camelot 5MP and Aptina HDR image sequences.*

3.5.2 Scene 2: External Illumination Through a Window

In this second scene we present a typical case of high illumination entering a workspace through a window. The tripod does a clockwise rotation motion during this sequence. In this case, we further investigate the saturation problem highlighted in scene 1 (Figure 3.4). However this and on contrary to scene 1, only concerns a restricted area (the window). Backlighting conditions are quite common in indoor scenario and a handicap in feature tracking as reflection of the light varies from one image to another.

It usually results in pixel saturation of the area where the light comes from. Figure 3.7 illustrates localized saturation phenomenon on the Camelot 5 MP image sequence where the window frame details are lost in high intensity pixels. In this scene, the difficulty lies in the localization of interesting features that are mostly found on the window frame and on furniture around the window. The reflection of the light on the latter gives an extra.

In Figure 3.8, we observe the same continuous decreasing trend as in scene 1 except that the final tracked feature rates are slightly higher. In Table 3.3 in the case of Camelot 5MP, they oscillate approximately between 20% and 40% of the initial tracked features.

Figure 3.9 illustrates the saturation phenomenon around the window area shown by a peak of 4 % at pixel bin 255. Except for this peak, the distribution of pixels in the intensity bins is coherent.

Table 3.3. *Scene 2: Tracking Performance Summary.*

		Detected Features	Initially Tracked	Finally Tracked	
				No.	*%*
FAST	*Camelot 5MP*	333	221	49	22.2
	Aptina HDR	363	341	333	97.6
GFTT	*Camelot 5MP*	500	268	97	36.1
	Aptina HDR	192	159	148	93
SIFT	*Camelot 5MP*	475	236	84	35.6
	Aptina HDR	79	74	70	94.6
SURF	*Camelot 5MP*	2585	1221	486	39.8
	Aptina HDR	712	640	611	95.5

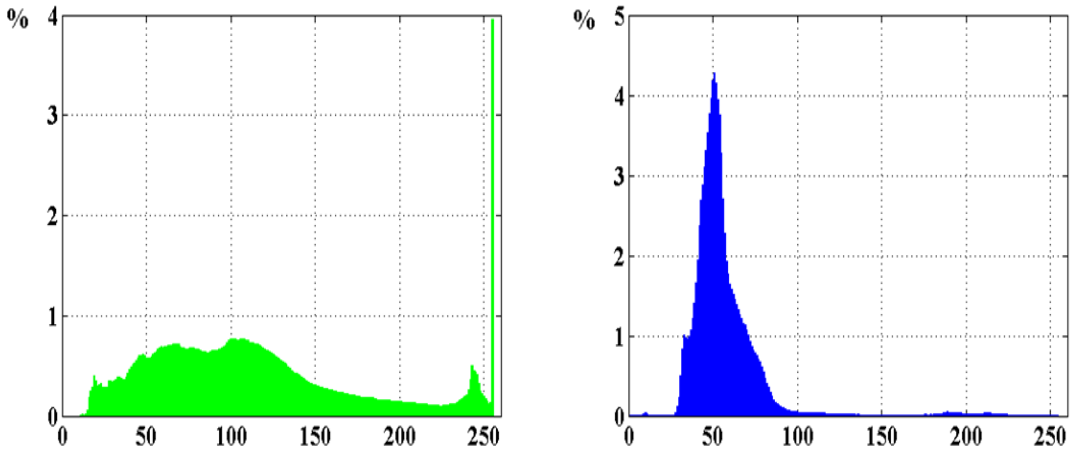


Figure 3.9. *Scene 2: Average greyscale histogram of image sequence Left: Camelot 5MP - Right: Aptina HDR (percentage of pixels/intensity).*

Results obtained on the Aptina HDR image sequence highlight the number of tracked feature opportunities that have been lost because of the saturation problem on the Camelot 5MP image sequence. Indeed, the majority of tracked features belong to the window frame and its closest areas.

Furthermore, as reported in Table 3.3 almost the entire set of initially tracked features are maintained until the end the sequence. Table 3.4 highlights the interval, which gathers around 80% of the pixels. This interval is localized more on the dark side the of pixel intensity axis. The matt effect of Aptina HDR images, however, has no repercussion on the algorithm performance.

Table 3.4. *Scene 2: Pixel Density Statistics of Average Greyscale Histogram of Image Sequences.*

	Dense interval(s) (>1%)	Pixels per interval(s)		
		%	Mean	Std
Camelot 5MP	[255]	3.95	3.95	-
Apitna HDR	[31, 71]	80.94	2.31	1.11

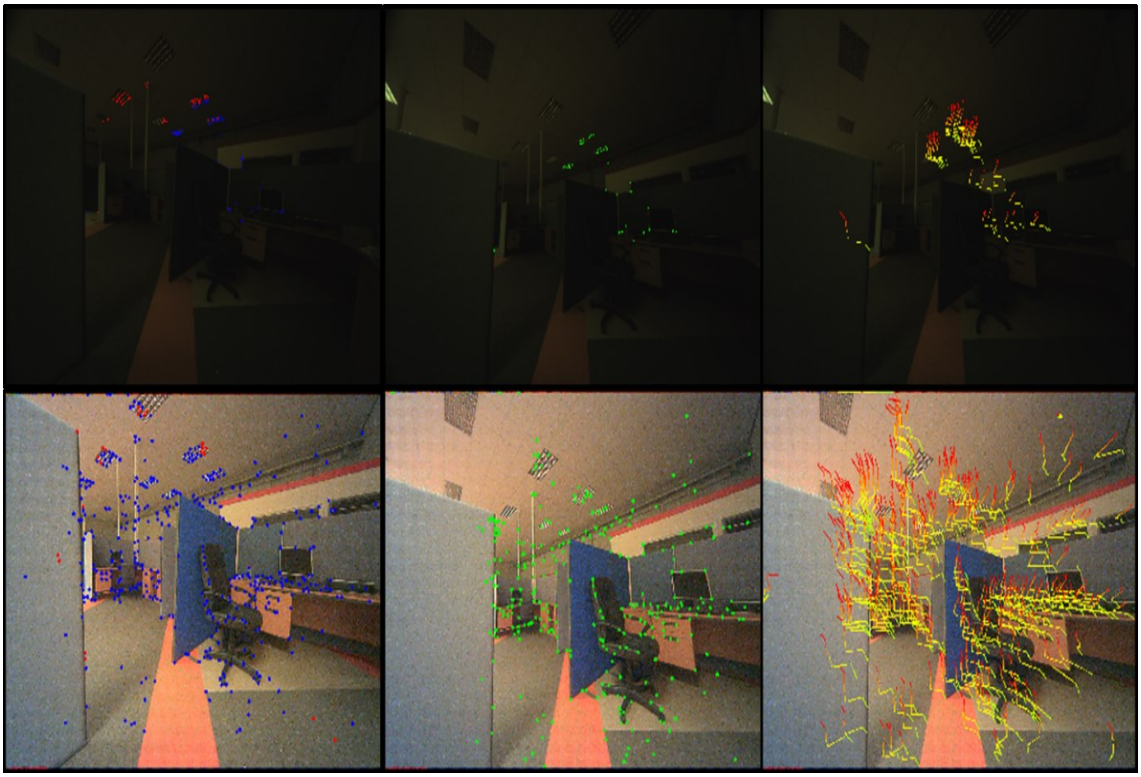


Figure 3.10. *Scene 3 using GFTT feature detector - Top: Camelot 5MP - Bottom: Aptina HDR - From the left to the right: first image, initial tracked features (red: lost, blue tracked); middle image final tracked features (green points); final image with full optical flow (colour shading).*

3.5.3 Scene 3: Dim Lighted Office

The third scene takes place in a dim lit office. The tripod structure is gradually elevated in a circular manner. For this test, the objective is to assess tracking performance in a low definition context due to the reduction of the source of illumination with only out of three rows of ceiling lights switched on. In this scene illustrated in Figure 3.10, despite the darkness and a pronounced uniform matt effect in the Camelot 5MP image sequence, the contents of the office are still discernible.

The reduced contrast of the scene results in limited sets of detected features in the case of Camelot 5MP. However, and in contrast to the two previous scenes, the quasi totality of the few initially tracked features is maintained until the end of the sequence except for SIFT, which is very sensitive to illumination (Table 3.5 and Figure 3.11).

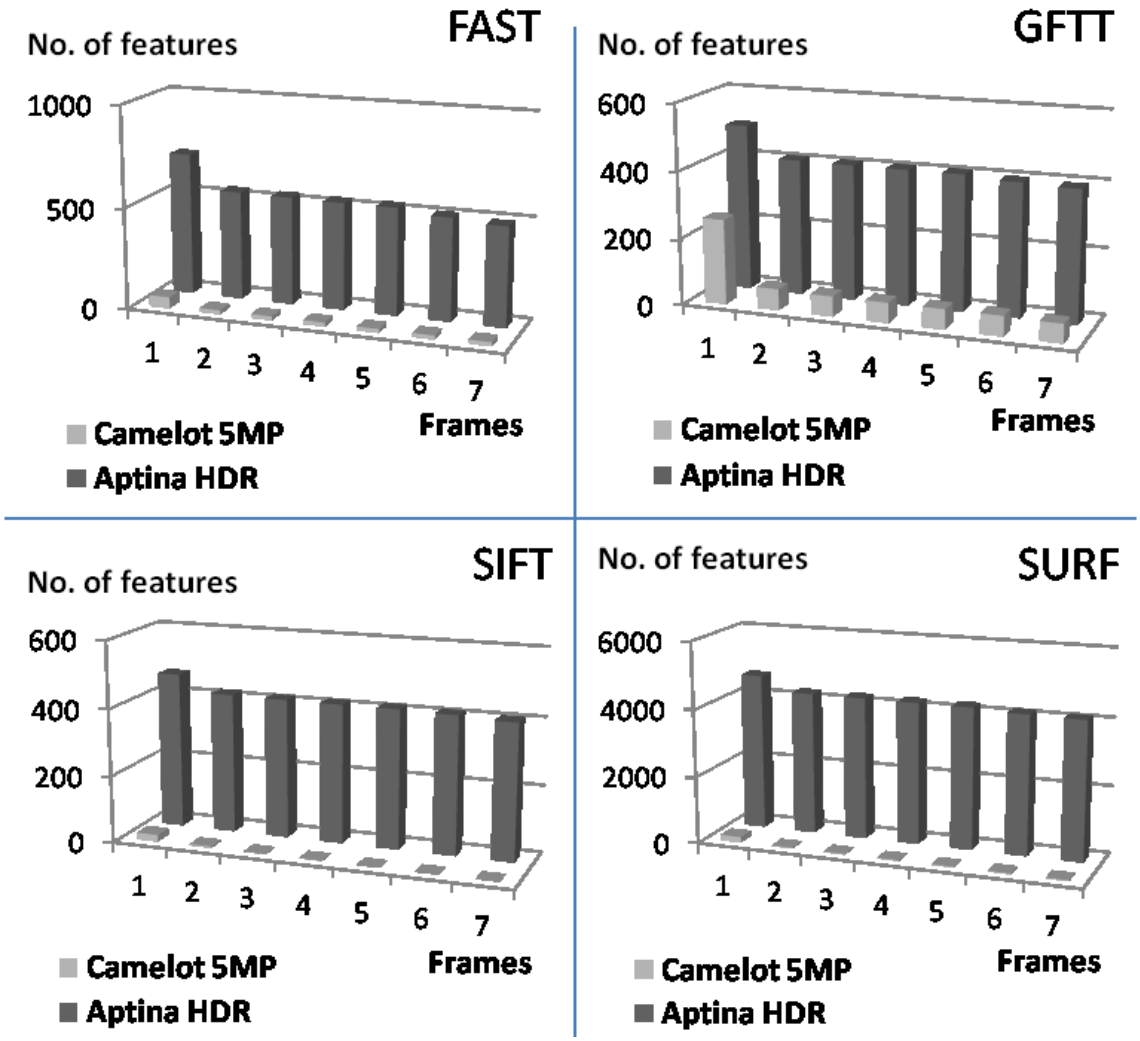


Figure 3.11. *Scene 2: Tracking frame length histogram graphs of Camelot 5MP and Aptina HDR image sequences.*

As reported in Table 3.6, 95 % of the pixels are located in the early start of the pixel intensity axis with a peak at intensity 11 highlighted in Figure 3.12. On the other hand, the contents of the scene are clearly distinguishable in the Aptina HDR image sequence although we notice noise appearing in the images. This phenomenon is especially noticeable in the SURF feature detection score. Indeed, in this case a certain number of residual noises were considered as features.

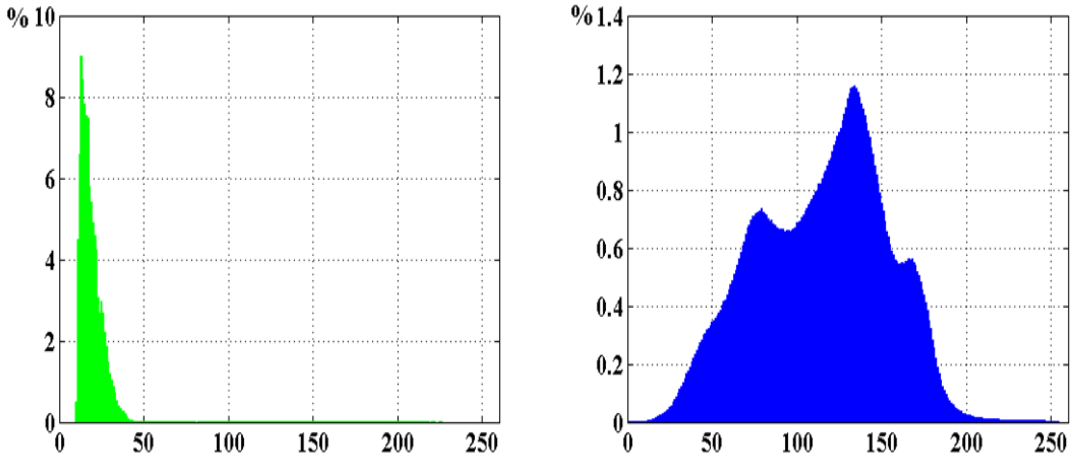


Figure 3.12. *Scene 3: Average greyscale histogram of image sequence Left: Camelot 5MP - Right: Aptina HDR (percentage of pixels/intensity)*

Table 3.5. *Scene 3: Tracking Performance Summary.*

		Detected Features	Initially Tracked	Finally Tracked	
				No.	*%*
FAST	*Camelot 5MP*	55	19	19	100
	Aptina HDR	704	536	489	91.1
GFTT	*Camelot 5MP*	255	63	57	90.5
	Aptina HDR	500	408	393	96.3
SIFT	*Camelot 5MP*	20	4	0	0
	Aptina HDR	466	417	405	98.8
SURF	*Camelot 5MP*	162	22	21	95.4
	Aptina HDR	4654	4230	4128	97.6

However, this does not affect the general trend constituted mainly with correct features, which provides a coherent optical flow (Figure 3.10) while conserving a high and stable percentage of initially tracked features during the full sequence (Figure 3.11, Table 3.5). It also conserves a homogenous distribution of the pixels in the average histogram of its image sequence Figure 3.11.

Table 3.6. *Scene 4: Pixel Density Statistics of Average Greyscale Histogram of Image Sequences.*

	Dense interval(s) (>1%)	Pixels per interval(s)		
		%	Mean	Std
Camelot 5MP	[10, 15]	94.93	15.82	13.64
Apitna HDR	[32, 87]	65.02	1.16	0.089

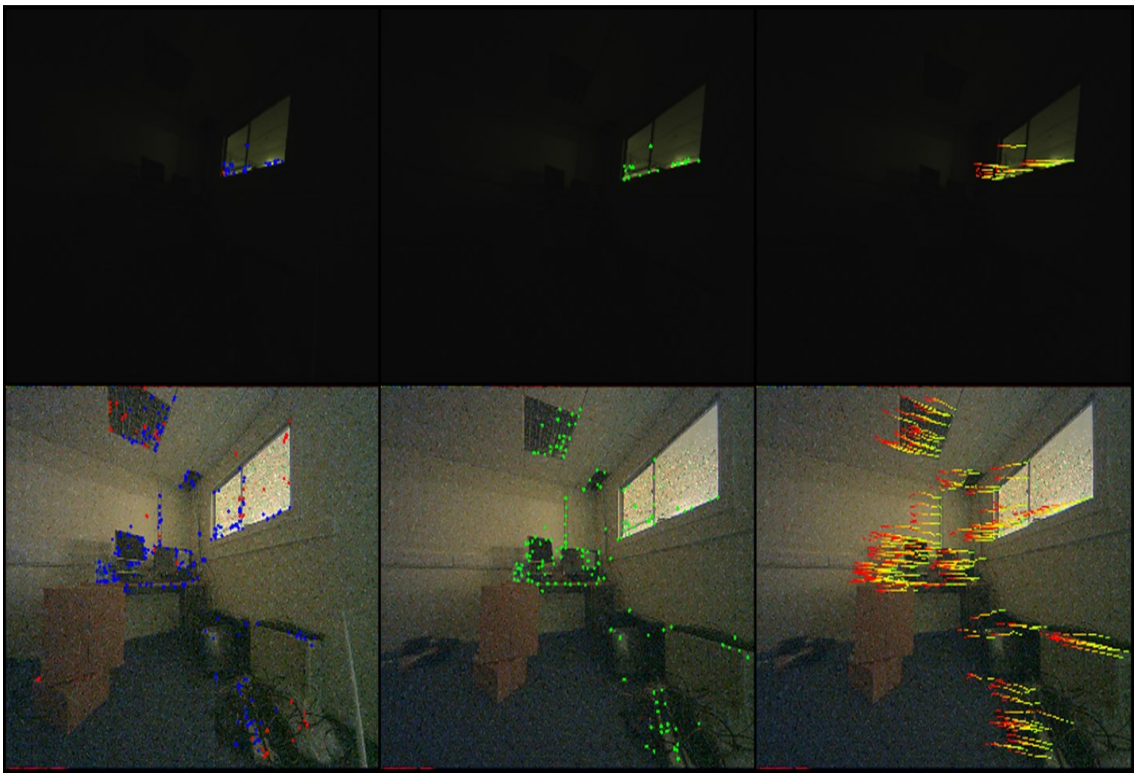


Figure 3.13. *Scene 4: using SIFT feature detector - Top: Camelot 5MP - Bottom: Aptina HDR - From the left to the right: first image, initial tracked features (red: lost, blue tracked); middle image final tracked features (green points); final image with full optical flow (colour shading).*

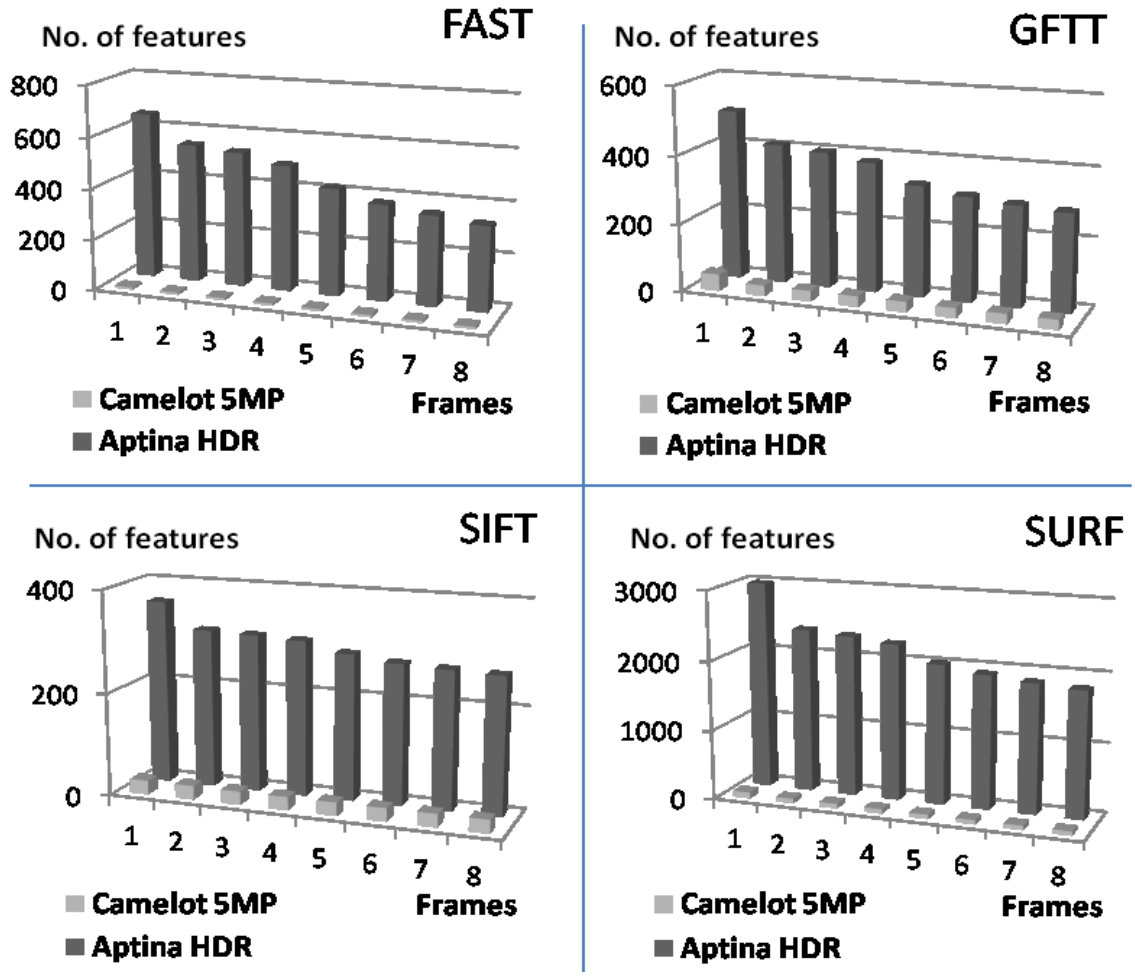


Figure 3.14. *Scene 2: Tracking frame length histogram graphs of Camelot 5MP and Aptina HDR image sequences.*

3.5.4 Scene 4: Dark Room

In this scenario, darkness conditions are pushed further by acquiring an image sequence in a dark room where the only source of light penetrates through a window. Here, the tripod is moved following a pure anti-clockwise rotation motion. Figure 3.13 illustrates the difference between the two image sequences in the scene representation, which is even more obvious when compared to the previous tests.

Hence, only the window is distinguishable in the Camelot 5 MP image sequence. Yet, similarly to scene 3 the only few initially tracked features are maintained with a high percentage until the end of the sequence (see Figure 3.14, Table 3.7). As illustrated in

Figure 3.15 in the case of Camelot 5MP the pixel density is greater than in scene 3 with 94 % distributed only an interval of 5 pixel intensity bins (Table 3.8). The Aptina HDR image sequence gives a better perception of the scene, although images are more affected by noise compared to scene 3 (see Figure 3.13). Consequently a high percentage of detected features consists of noise especially with SURF. However, these false positive features are gradually discarded along the sequence. Hence, feature tracking behaves as a kind of progressive filter here. This is noticeable with the homogenous trend of the optical flow with all feature detectors like in Figure 3.13 for instance. This phenomenon consequently affects the overall rate of finally tracked features which still remains reasonable.

Additionally, the number of finally tracked features with the Aptina HDR device remains much higher and spread over images than with the Camelot 5 MP camera. Despite the darkness of the room, we remark in Figure 3.15, that the overall pixel distribution remains homogenous even if it is on a smaller interval than usual.

Table 3.7. *Scene 4: Tracking Performance Summary.*

		Detected Features	Initially Tracked	Finally Tracked	
				No.	*%*
FAST	*Camelot 5MP*	8	8	8	100
	Aptina HDR	652	543	330	60.8
GFTT	*Camelot 5MP*	49	31	29	93.5
	Aptina HDR	500	411	286	69.6
SIFT	*Camelot 5MP*	27	27	26	96.3
	Aptina HDR	360	310	267	86.1
SURF	*Camelot 5MP*	85	65	58	89.2
	Aptina HDR	2984	2357	1818	77.1

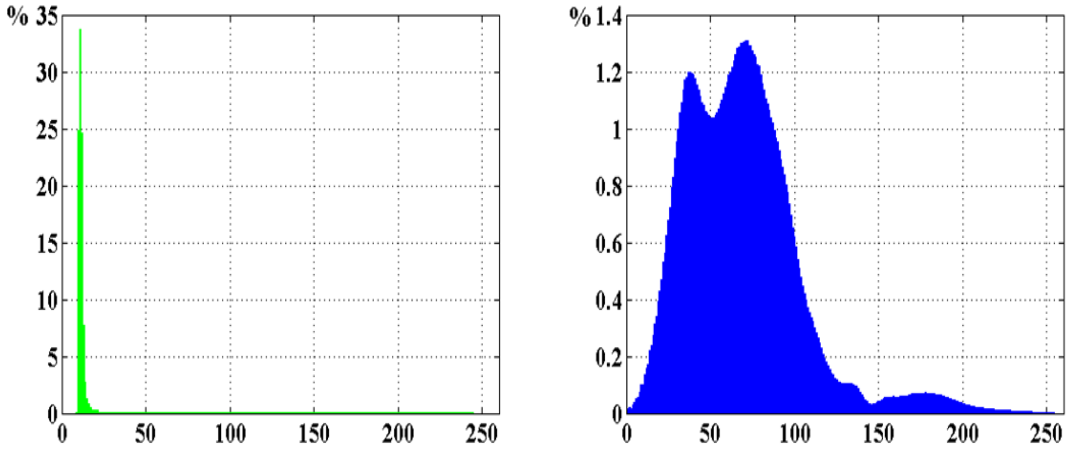


Figure 3.15. *Scene 4: Average greyscale histogram of image sequence Left: Camelot 5MP - Right: Aptina HDR (percentage of pixels/intensity).*

Table 3.8. *Scene 4: Pixel Density Statistics of Average Greyscale Histogram of Image Sequences.*

	Dense interval(s) (>1%)	Pixels per interval(s)		
		%	Mean	Std
Camelot 5MP	[10, 15]	94.93	15.82	13.64
Apitna HDR	[32, 87]	65.02	1.16	0.089

3.5.5 Discussion

HDR imaging solution for feature detection and tracking to cope relatively well with extreme illumination conditions. As reported in Table 3.9 ratios of number of finally tracked features between the Aptina HDR and the Camelot 5 MP image sequences are always greater than one. This remains true if we analyse performance either by feature detection techniques or by scenes. Indeed, using the algorithm on the Aptina HDR images provides several times more features than with the Camelot 5 MP. Having said this, the overall score looks of 51.41 times more features on average for the Aptina HDR image sequence should be tempered. Indeed, results obtained on scene 3 and especially on scene 4 (dim and dark conditions) with SURF are certainly exaggerated because of its sensitivity towards noise.

Table 3.9. *Final Tracked Features Number Ratio Summary.*

	Aptina HDR / Camelot 5MP (No.)				
	Scene 1	Scene 2	Scene 3	Scene 4	Mean
FAST	37.44	6.79	25.73	41.25	27.80
GFTT	22.37	1.52	6.89	9.86	10.16
SIFT	5.07	0.83	405	10.27	105.29
SURF	20.44	1.25	196.57	31.34	62.4
Mean	21.33	2.60	158.55	23.15	51.41

Table 3.10. *Final Tracked Features Percentage Difference Summary.*

	Aptina HDR - Camelot 5MP (%)				
	Scene 1	Scene 2	Scene 3	Scene 4	Mean
FAST	72.4	75.4	-8.8	-39.2	25.4
GFTT	85.5	56.9	5.8	-23.9	31.07
SIFT	82.8	59	98.8	-10.2	57.6
SURF	85.3	55.7	2.2	-12.1	32.77
Mean	81.95	61.75	24.5	-21.35	36.71

In Table 3.10, this time, it is the difference of percentage of finally tracked features between Aptina HDR and Camelot 5 MP image sequences, which is analysed by scenes and by feature detection techniques used. Here as well we remark high differences with an overall difference of 36.71 % more finally tracked features with the Aptina HDR images. The only exception is on scene 4 where percentage of finally tracked features on Camelot 5 MP images is higher. However, as it has been explained in section 3.4.4 this is the consequence of noisy features progressively discarded along the tracking process and which lowers the rate of finally tracked features with the Aptina HDR images. Moreover the ratio of respective finally tracked features in Table 3.9 shows 23.15 times more features with the Aptina HDR than with the Camelot 5 MP images.

The results show clearly the superiority of the Aptina HDR over the Camelot 5 MP, in terms of successful rendering of extreme illumination conditions. This has been confirmed through the experiments with a much higher number and a higher percentage of finally tracked features in comparison to the Camelot 5 MP (Table 3.9 and 3.10). The positive contribution of HDR technology is real and its impact in enhancing detection and tracking in scenarios under extreme illumination is certain.

3.6 Chapter Conclusion

This chapter presented a comparative evaluation of an HDR imaging sensor with a 5 Megapixel image sensor. This was done in four different environments, each presenting specific challenging illumination conditions. The quality of the image sequences acquired with the two devices were assessed using four feature detection techniques and a feature tracking algorithm based on the pyramidal implementation of the KLT feature tracker. It has been shown that under challenging visibility conditions images from the HDR image sensor provide a large number and a high rate of final tracked features which are several times higher compared to images acquired with the 5 Megapixel image sensor.

4 Motion Estimation

4.1 Overview

Motion estimation is the central part of visual odometry algorithms. This process computes ego-motion of a camera equipped platform relatively to a pair of subsequent frames. To do so, motion estimation algorithms use correspondences between consecutive images. The successive accumulation of these relative inter-frame motions provides the full trajectory that a camera equipped platform has followed. In Chapter 2, emphasis was put on feature detection and feature tracking. The reasons behind this are related to the critical role and influence that feature correspondences have on the motion estimation process.

Notably, the feature detection, stereo tracking, and feature tracking strategy explained in Chapter 2 forms the same base for all the motion estimation algorithms presented in this chapter.

4.2 Motion Estimation Variants

In Chapter 2, we explained that the most suitable definition of features to adopt is points. This derives from the chosen sparse approach and also because of unstructured scene context that shall be encountered. Based on a two-dimensional or three-dimensional representation of feature point correspondences, motion estimation can be solved following three general methods.

The first general method consists in solving motion estimation only from the 2D image coordinates of the correspondences between subsequent images I_p and I_c (where $_p$ stands for the previous image and $_c$ stands for the current image). The essential matrix E, which contains the camera motion parameters of a calibrated camera up to a scale factor, is computed from the 2D correspondences with a minimum of 5 [49]. In the case of stereo vision, the essential matrix is calculated from the Fundamental matrix F and the cameras calibration matrix K. There are also other efficient variants using 8 or more correspondences [50]–[53]. Having more features increases the robustness against noise and also provides the advantage of solving the over-determined system. Then, the inter-frame rotation matrix R and translation vector t are extracted from the essential matrix using SVD (Singular Value Decomposition). Finally, relative scale for the translation parameters can be calculated with the ratio of triangulated 3D correspondences between consecutives images or using the trifocal constraint for 2D correspondences over 3 consecutive frames [7]. This general method is just briefly reviewed here and is not studied in this manuscript. More emphasis will be given on the two remaining general methods which will form respectively the two main sections (Sections 4.3 and 4.4) of this chapter.

The second general method uses feature point correspondences defined in 3D only, and deal with the motion estimation problem following a spatial approach instead of a planar approach as used in the first general method Among the three general methods, 3D registration problems were the earliest studied and several solutions were proposed [54]–[59]. This method is very suitable for stereo visual odometry scheme where 3D structure correspondences that can be easily obtained by triangulation from the 2D images coordinates using stereoscopy properties for calibrated cameras [11]. The inter-frame motion composed of the rotation matrix R and translation vector t are computed by

determining the aligning transform, which minimises the Euclidean distance (L_2) between the two 3D clouds of points respectively belonging to previous and current images I_p and I_c.

The third general method uses both 2D and 3D representations of feature point correspondences. This approach takes advantage of both planar and spatial representations. This method follows the same process as the second method. The main difference, however, with the latter lies in the minimisation of the image re-projection error instead of the Euclidean distance between the 3D positions of the feature point correspondences. Re-projecting feature correspondences in the image plane is more accurate than dealing only with 3D points that might induce a lot more uncertainty especially in depth estimation according to the quality of triangulation as mentioned by [49].

4.3 3D Correspondences Based Motion Estimation

This section details the method, which solves the motion estimation problem using only the 3D representation of the feature point correspondences between consecutive stereo images. Formulation of the motion estimation problem is presented, along with the technique that has been chosen to solve it. Improvements using outlier rejection scheme are also explained. Finally, this section ends with the presentation of the results obtained following this method.

4.3.1 Formulation of the Motion Problem

Considering two sets of rigid 3D feature point correspondences P_p and P_c, the motion equation is decoupled into a rotation and a translation as follows:

$$P_{c(i)} = RP_{p(i)} + t \tag{4.1}$$

where R and t represent the 4x4 homogeneous rotation matrix and the 4x1 homogeneous translation vector respectively. These are the two unknown elements. Additionally, $P_{p(i)}$ and $P_{c(i)}$ are respectively the 4x1 homogeneous vectors of the 3D point positions in the previous and the current stereo image. In reality, this identity is almost impossible to be validated because of noise and approximation errors. Thus equation (4.1) can be re-formulated as:

$$P_{c(i)} = RP_{p(i)} + t + o \tag{4.2}$$

or

$$P^*_{c(i)} = RP_{p(i)} + t \tag{4.3}$$

Where $P^*_{c(i)}$ is the 4x1 homogeneous approximated post-motion vector calculated with R, t and previous 3D point $P_{p(i)}$ and o represent the Gaussian noise.

4.3.2 Quaternion Based Method for Motion Estimation

The method that has been chosen to solve the equation of motion (4.1) is a closed form solution of absolute orientation proposed by Horn [54]. This method presents the advantage of using quaternion representation for the orientation instead of an Euler angle representation. The later can be affected by a loss of a degree of freedom also called gimbal-lock. A quaternion is composed of 4 elements, which consist of one scalar and three imaginary parts that are also orthogonal [60] and it is expressed as follows:

$$q = q_0 + q_1\vec{\imath} + q_2\vec{\jmath} + q_3\vec{k} \tag{4.4}$$

where $\vec{\imath}$, $\vec{\jmath}$,and $\vec{k}$ are unit vectors fulfilling these following conditions:

$$\begin{cases} \vec{\imath}^2 = \vec{\jmath}^2 = \vec{k}^2 = -1 \\ and \\ \vec{\imath}\vec{\jmath} = \vec{k},\ \vec{\jmath}\vec{k} = \vec{\imath}\ and\ \vec{k}\vec{\imath} = \vec{\jmath} \end{cases} \tag{4.5}$$

In this method, the first element to be calculated is the quaternion vector q from which the rotation matrix will be computed. Considering a set of n (n ≥ 3) non co-planar correspondences of rigid feature points $P_{p(i)}$ and $P_{c(i)}$, the quaternion vector takes the values of the eigen-vector containing the largest eigen-value of the 4x4 matrix Q computed as follows:

$$Q = \begin{bmatrix} trace\left(\sum_{P_cP_p}\right) & \Delta^T \\ \Delta & \sum_{P_cP_p} + \sum_{P_cP_p}{}^T - trace\left(\sum_{P_cP_p}\right) \cdot I_3 \end{bmatrix} \tag{4.6}$$

Where I_3 is a 3x3 identity matrix and $\sum_{P_cP_p}$ is the cross covariance matrix defined as:

$$\sum_{P_cP_p} = \frac{1}{n}\sum_{i=1}^{n}\left(P_{c(i)}P_{p(i)}^{T}\right) - \left(\overline{P_c}\ \overline{P_p}^{T}\right) \tag{4.7}$$

with $\overline{P_p}$ and $\overline{P_c}$ respectively the 3x1 mean vectors of the previous and the current 3D set of feature points:

$$\overline{P_p} = \frac{1}{n}\sum_{i=1}^{n} P_{p(i)} \tag{4.8}$$

$$\overline{P_c} = \frac{1}{n}\sum_{i=1}^{n} P_{c(i)} \tag{4.9}$$

The 3x1 vector Δ is composed of three elements of the 3x3 matrix D which is defined as the cross covariance matrix (4.7) and its transpose:

$$\Delta = [D_{23} \quad D_{31} \quad D_{21}] \tag{4.10}$$

with

$$D = \sum_{P_cP_p} - \sum_{P_cP_p}{}^{T} \tag{4.11}$$

As mentioned, above the eigen-vector holding the largest value of the matrix Q gives the quaternion vector q (4.4). The rotation matrix is then obtained from the elements of the quaternion vector as follows:

$$R = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 + q_2^2 - q_1^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 + q_3^2 - q_1^2 - q_2^2) \end{bmatrix} \tag{4.12}$$

In the case of $n \geq 3$, the translational vector t defined in (4.1) can be computed as the difference between the centroids of the two sets of corresponding points P_p and P_c (4.8 and 4.9) and the calculated rotation matrix R (4.12) as:

$$t = \overline{P_p} - R\overline{P_c} \tag{4.13}$$

For any motion estimation technique in general and for the quaternion based motion estimation algorithm in particular, the computed solution is highly dependent on the quality of the correspondences.

Preceding stages consisting of feature detection, stereo matching and feature tracking form the base of the visual motion estimation algorithm, and serve as input data. Although the use of advanced image processing techniques and the establishment of strict selection criteria help to increase the confidence in the set of selected pair of points, it remains difficult to avoid mismatches completely. These are sources of ambiguity and might lead the motion algorithm to fail according to the rigid body assumption for motion estimation algorithm considering all the correspondences are good.

4.3.3 Outliers Rejection

The input of motion estimation algorithm is composed of two sets of 3D correspondences. As it was shown in Chapter 2 (Section 2.5 and Section 2.7), epipolar constraints from the stereoscopy property as well as the distance criterion contribute to pre-filter the correspondences along the visual odometry process until reaching the motion estimation stage. Despite the fact that these constrain the correspondences linking previous and current stereo images they might still contain a certain number of outliers. These are mainly resulting from mismatching either on the stereo matching or on feature tracking stages.

It has been seen in subsection 4.3.1 that the motion estimation algorithm assumes perfect correspondences between feature points belonging to previous and current stereo images. In order to get closer to this assumption, using an outliers rejection scheme that contributes to remove wrong correspondences is necessary. These later affect the accuracy of the motion estimation algorithm and may even lead to a complete failure. In the visual odometry process, which accumulates inter-frame motion estimations these failures and subsequently degraded estimations can have a dramatic impact on the generated trajectory even if it happens only once (between two stereo frames).

Outliers rejection problem is usually tackled using robust statistical methods, alone or with a RANdom SAmple Consensus (RANSAC) [25], [57], weighted least squares methods[61], or other iterative approaches [62]–[66] . Based on the work of [66], the adopted approach uses only 3D geometric properties of the corresponding set of points. The reason behind this choice is motivated by the strength of geometrical constraints. Indeed, the 3D position of features is theoretically the same before and after the motion.

Consequently the distances between corresponding 3D points should remain the same during the frame regardless of the camera positioning and orientation. Thus, by comparing the distance between a 3D point to the other elements of its own set on both camera coordinate systems should validate the initial guess if the distances remain the same. If not, this means that at least one of the correspondences is wrong. The robustness of this approach enables the detection of almost all the outliers even in cases where their amount is significantly higher than the amount of inliers. This is a major advantage in comparison to statistical approaches that can be misleading when correspondences are dominated by outliers. Additionally, in the case that eventual outliers pass through the outlier removal stage [66], their influence would be minimal as they are restrained by the filtering algorithm to lie within the close neighbourhood of their correct position.

Considering two sets of n previous and current feature correspondences P_p and P_c, the condition of preserved relative distances between 3D points for a same set within the two stereo camera coordinate systems is expressed as:

$$\left\|P_{p(i)} - P_{p(j)}\right\| = \left\|P_{c(i)} - P_{c(j)}\right\| \tag{4.14}$$

In reality this condition is impossible to realise due to the noisy nature of the 3D points that mainly results from feature localization and 3D reconstruction approximation errors. Consequently, equation (4.14) is re-formulated in an inequality as follows:

$$e = \left\|P_{p(i)} - P_{p(j)}\right\| - \left\|P_{c(i)} - P_{c(j)}\right\| < \varepsilon \tag{4.15}$$

where e is the difference of the relative differences between two relative points in each set and ε is a distance threshold set to 0.5. Figure 4.1 gives an illustration of this geometric constraint. Indeed adding weight related to the distance of these 3D points and normalise equation (4.15) would certainly strengthen this condition.

In addition to this constraint, authors in [66] give another constraint based on rotational property that restrain the maximum camera movement between two subsequent frames under a certain angle θ expressed below as:

$$\left(P_{p(i)} - P_{p(j)}\right)\left(P_{c(i)} - P_{c(j)}\right) > cos\theta \tag{4.16}$$

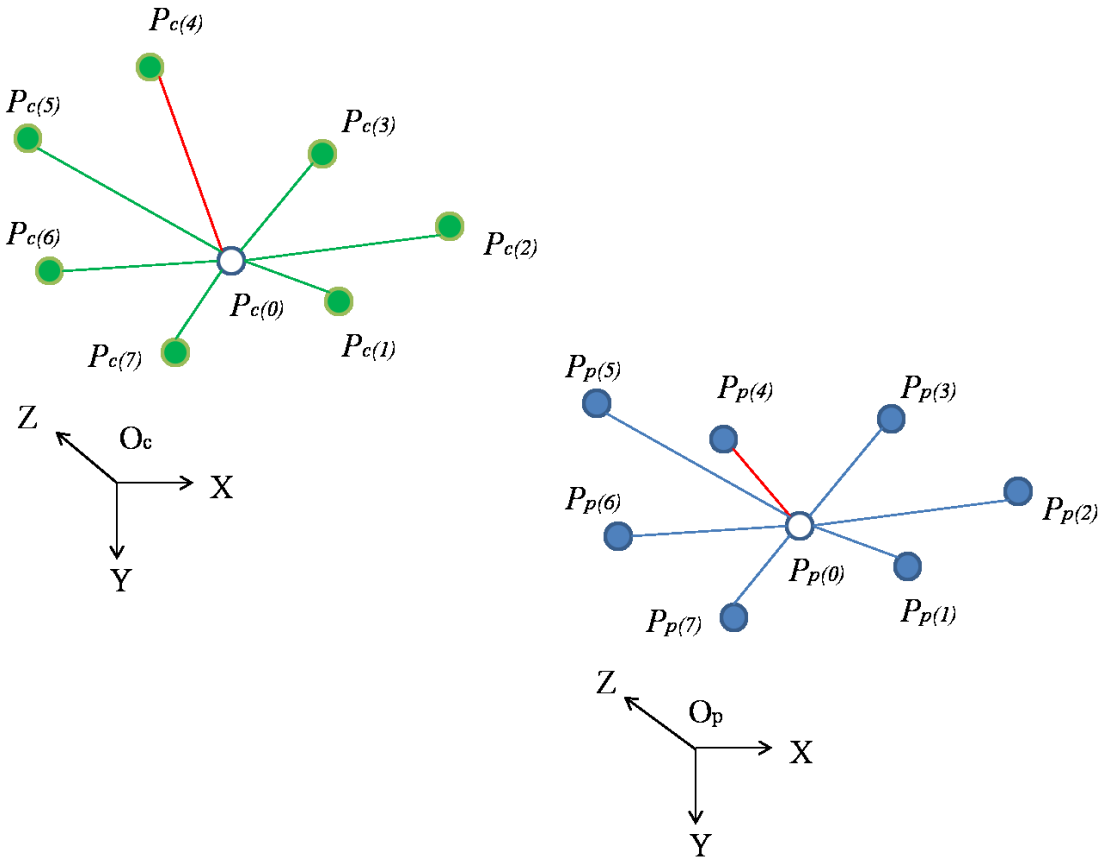


Figure 4.1. *Illustration of the geometric constraint based on distance preservation between two sets of 3D point correspondences. The red line highlights a wrong correspondence because of the non-preserved distance.*

It is proposed in [66] that θ equals to π/4. However this value was found too restrictive (as it will be demonstrated in section 4.3.4). A more reasonable value was set to be π/2 instead.

4.3.4 Results

In this section, results of visual odometry algorithm including outliers rejection based on the quaternion motion estimation method are presented. The algorithm is run on an outdoor urban environment dataset [107]. In this dataset, a car is equipped with stereo cameras (Point Grey® Flea2) mounted on its roof with a baseline of 0.57m. Visual data consist of 1344 x 372 rectified stereo images acquired at 30Hz. GPS/INS data are also

provided with this dataset and will serve as a positioning reference in the visual odometry trajectory comparison test.

In this dataset the vehicle is driven following a straight line at relatively high speed for about 65 m then the vehicle initiates a sharp turn to the right and continues straight on for about 10 m at low speed. This gives two angles of analysis during the comparison tests. The first part of the route presents a certain challenge due to the speed of the car causing a low persistency of the scene's content. This makes tracking of features close to the cameras harder while they represent the most meaningful data (i.e. presenting the lowest probability of localisation errors). In the second part of the route, persistency of the content of the scene is high but the car ego-motion in the turn leads to subsequent rotation changes between consecutive stereo images.

In the first instance, results that contributed to set the criteria values for equations (4.15) and (4.16) are presented including feature tracking assessment and visual odometry trajectories comparison.

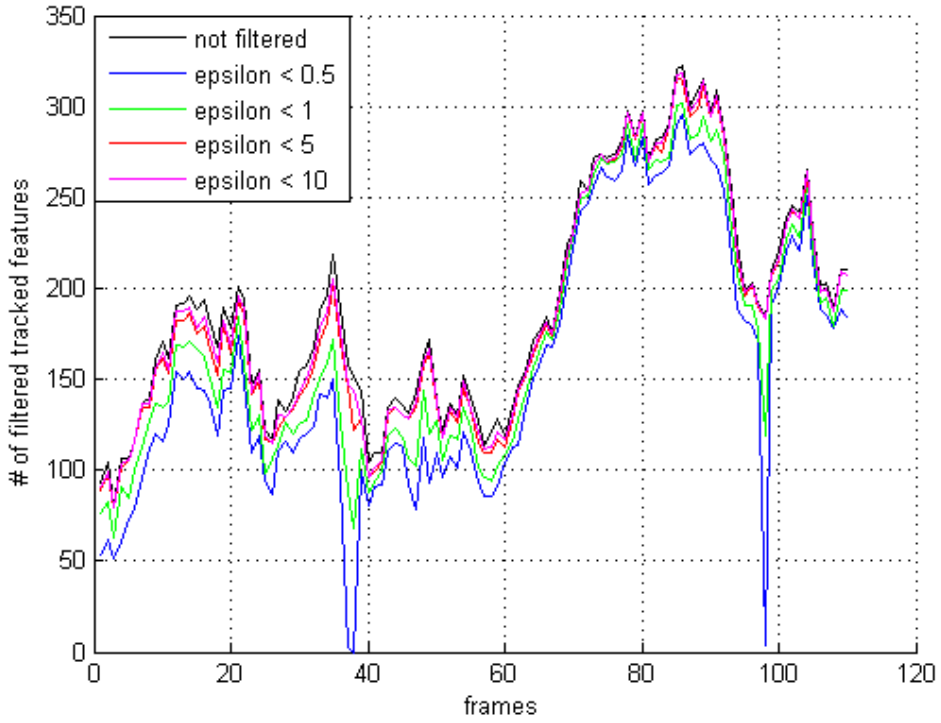


Figure 4.2. *Evolution of the number of features considered as inliers along the full sequence varying parameter ε.*

Then, comparison between the visual odometry trajectories with and without outliers rejection is given in order to highlight its importance in this process. Notably, all the visual odometry trajectories as well as the GPS/INS reference are projected on the horizontal plane and superimposed to provide the fairest comparison possible. This also makes possible the 3D visualisation of the presented results. Firstly, the assessment of the restriction level for rigid transform property expressed in equation (4.15) by varying parameter ε will be the focus of this section.

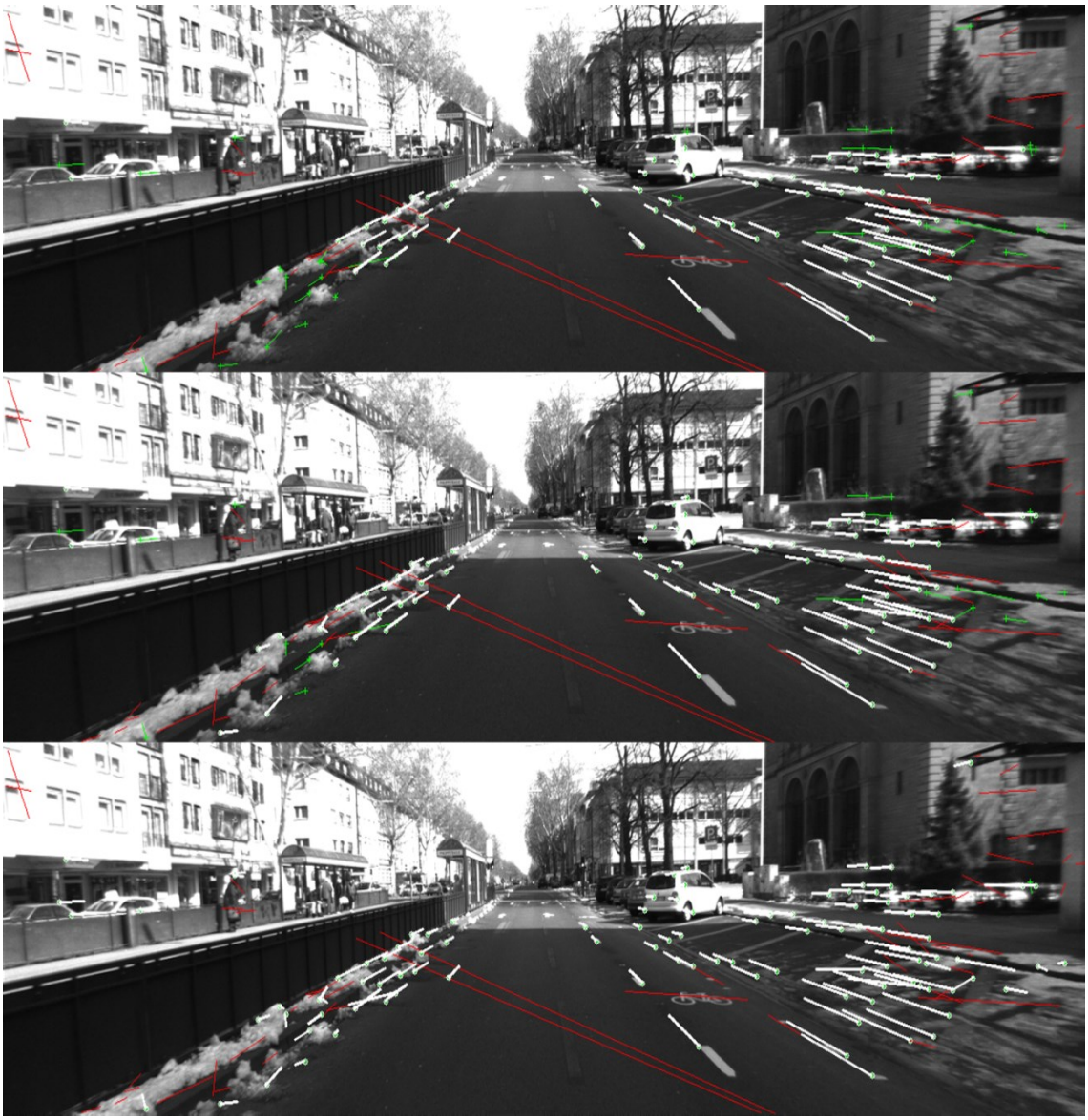


Figure 4.3. *Illustration on the current left image of the outlier rejection quality varying parameter ε. From top to bottom: ε = 0.5 ε = 1, ε = 5.*

Notably for this test, parameter θ for condition (4.16) is fixed to π/2. In Figure 4.2, results of remaining inliers from outliers rejection stage are showed with ε = {0.5; 1; 5; 10}.

It can be noticed that for all the data, the average number of features is lower in the first part than in the second part. This corroborates what has been mentioned above, regarding the nature of the full sequence. Obviously this curve with the lowest ε (0.5) is the most restrictive. For the curve, two peaks are noticeable where the outlier rejection algorithm only results with a small but sufficient number of filtered feature correspondences (more than 3) to achieve correct motion estimation. Figure 4.3 illustrates the quality of the outlier rejection on the straight-line part of the dataset and using the condition (4.15). Red lines represent the optical flows of originally tracked features with KLT. The green lines represent the optical flows of pre-filtered features after KLT stage according to the conditions cited in Chapter 2 (Section 2.7). White lines are the optical flows of filtered features after outliers rejection stage.

With ε equals to 0.5, only consistent correspondences are kept while with a higher value of ε, more outliers are remaining as it can be noticed on the bottom left as well as on the right side of the images (Middle and bottom). Figure 4.4 illustrates the quality of outlier rejection at the turning part of the dataset for condition (4.15). For this part of the trajectory, there is much less ambiguity as the speed of the car is more reasonable but also because the content of the scene is closer to the cameras.

However, it can be noticed that with ε set to 0.5, the few outliers present in images with a higher value of ε are avoided.

Figure 4.5 and 4.6 show 2D and 3D plots of the generated trajectories for the different values of ε respectively. In Figure 4.5, the trajectory with a value of ε greater than 0.5 deviates from the GPS/INS trajectory reference (black) around the middle of the straight line when the speed of the car is the highest, which affects the quality of initial set of tracked features with KLT.

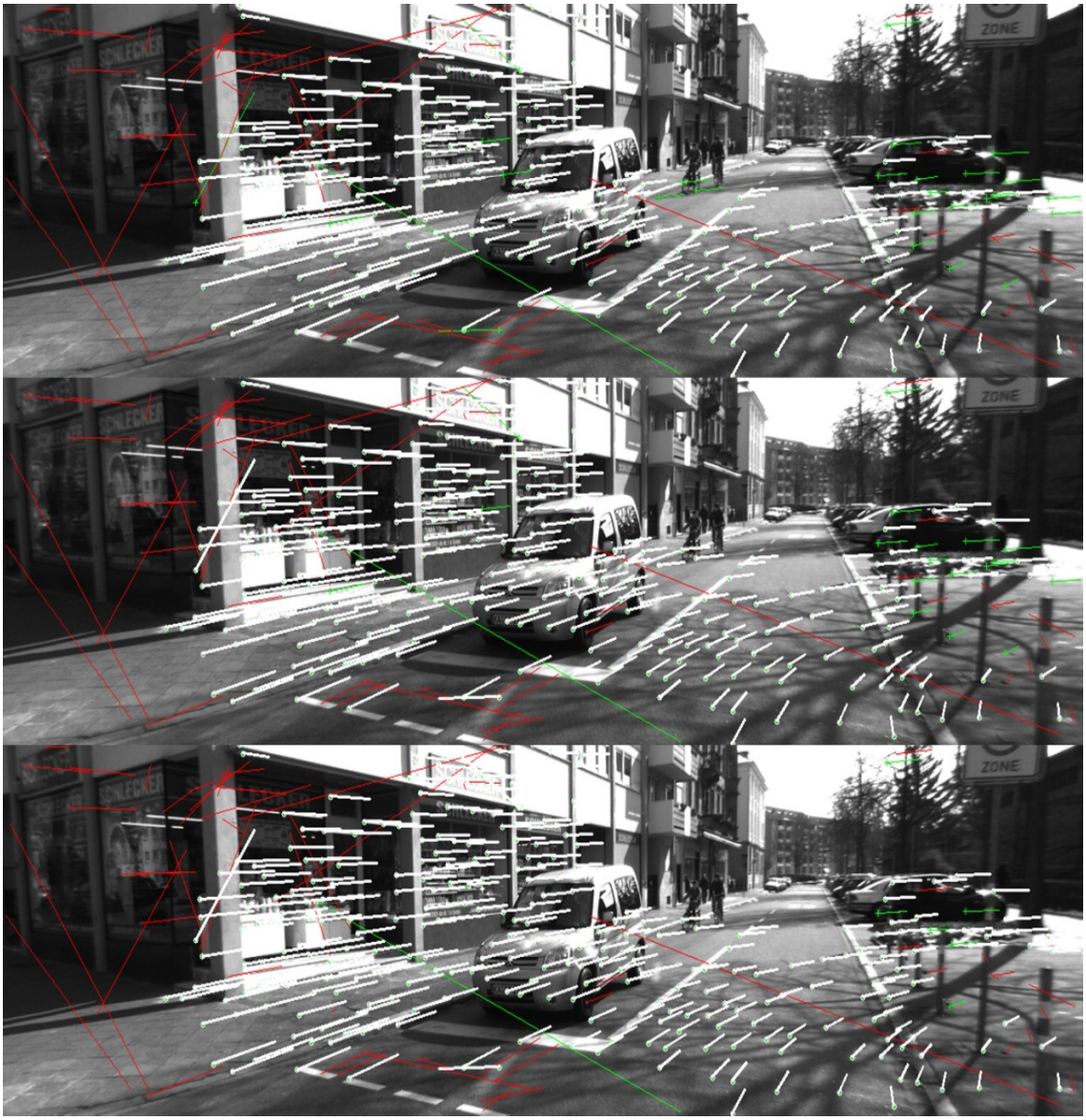


Figure 4.4. *Illustration on the current left image of the outlier rejection quality varying parameter ε. From top to bottom: ε = 0.5, ε = 1, ε = 5.*

Figure 4.6 shows that the estimation of height is getting worse while the value of ε is larger especially for the straight-line part of the trajectory. For the turning part, all the trajectories are affected in the same way in the height estimation. This is more a general problem of the motion estimation algorithm to deal with height estimation in specific conditions like for sharp turns.

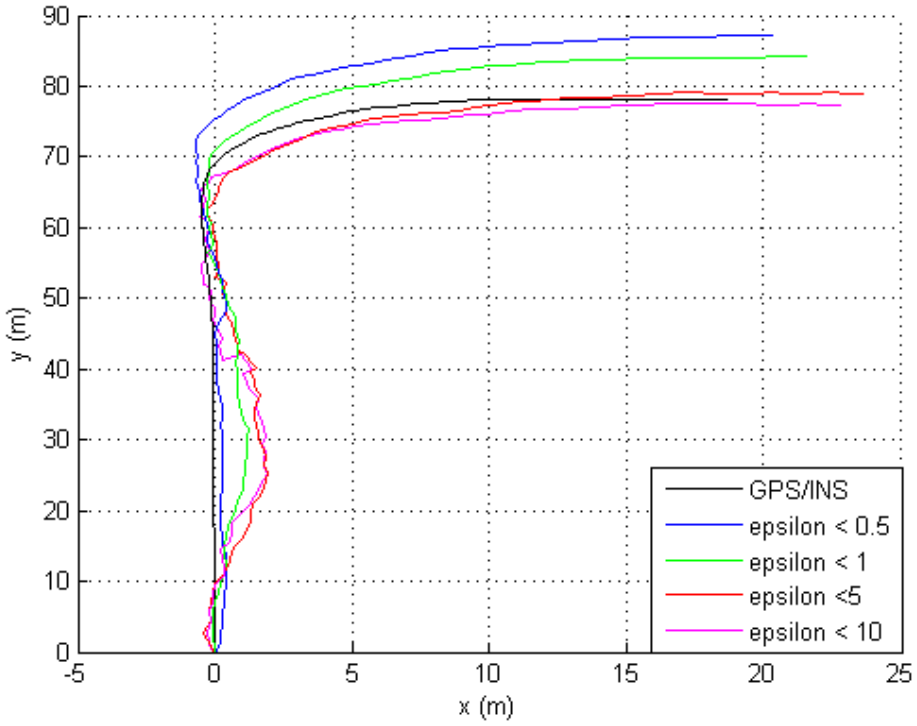


Figure 4.5. *Two dimensional plot of generated trajectories varying parameter ε.*

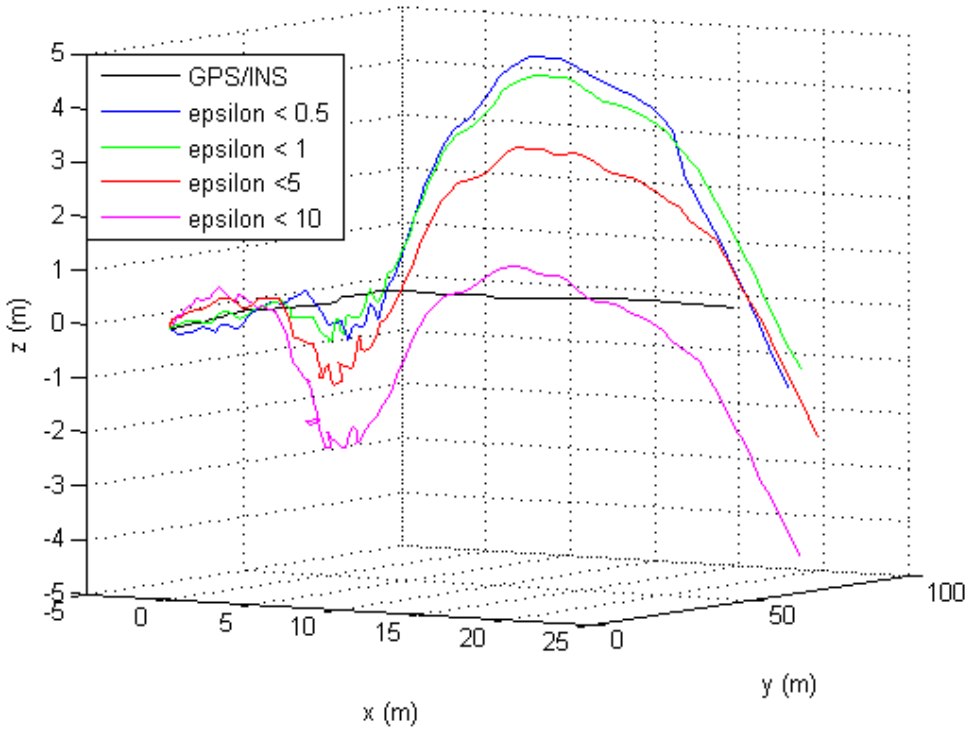


Figure 4.6. *Three dimensional plot of generated trajectories varying parameter ε.*

Setting ε to 0.5 is found to be the most acceptable restriction. Indeed, above this value, the generated trajectories deviate and are permanently affected. Below this value, the restriction is so constraining that for many frames motion estimation cannot be achieved because of the lack of features (lower than 3) as after the outliers rejection stage we are missing feature information.

In the same way as presented above, insights leading to the choice for the value θ regarding the second condition for rigid transformation expressed in (4.16) are given. Notably for this test, the parameter ε is fixed to 0.5 following the conclusion of what has been explained earlier. In Figure 4.7, results of remaining inliers from outliers rejection stage are showed with θ = {π/2; π/2.25; π/2.75; π/3}. The curve with θ set to π/2 is the less restrictive. For the other values of θ, a significant drop of the number of feature considered as inliers, especially in the straight-line part of the route, is noticed. This leads at several times to cases where not enough feature correspondences are available for the motion estimation to be conducted.

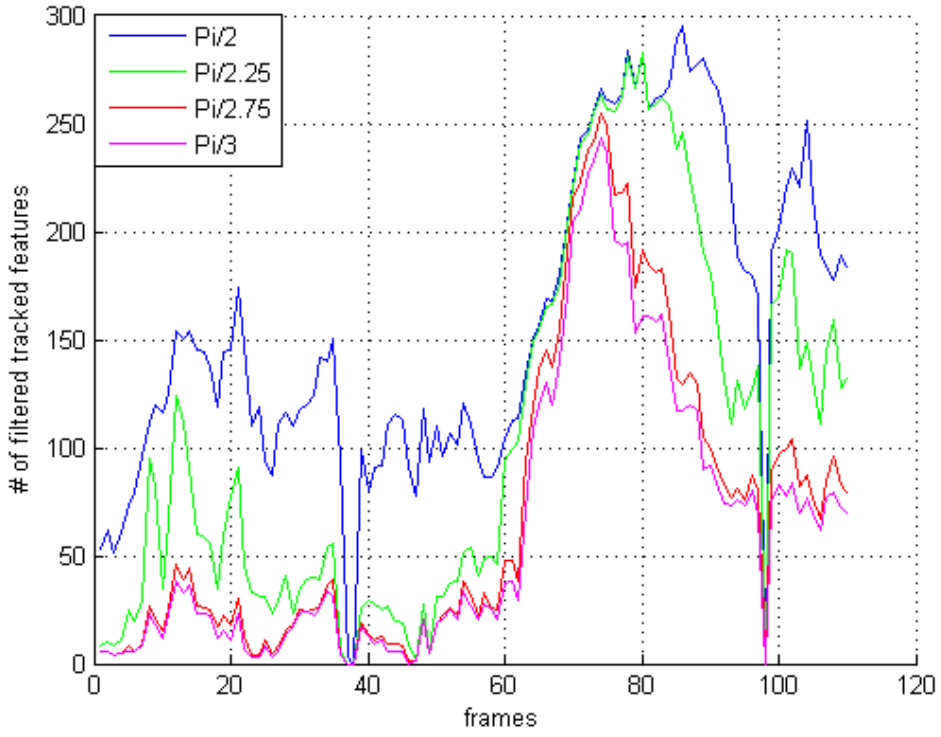


Figure 4.7. *Evolution of the number of features considered as inliers along the full sequence varying parameter θ.*

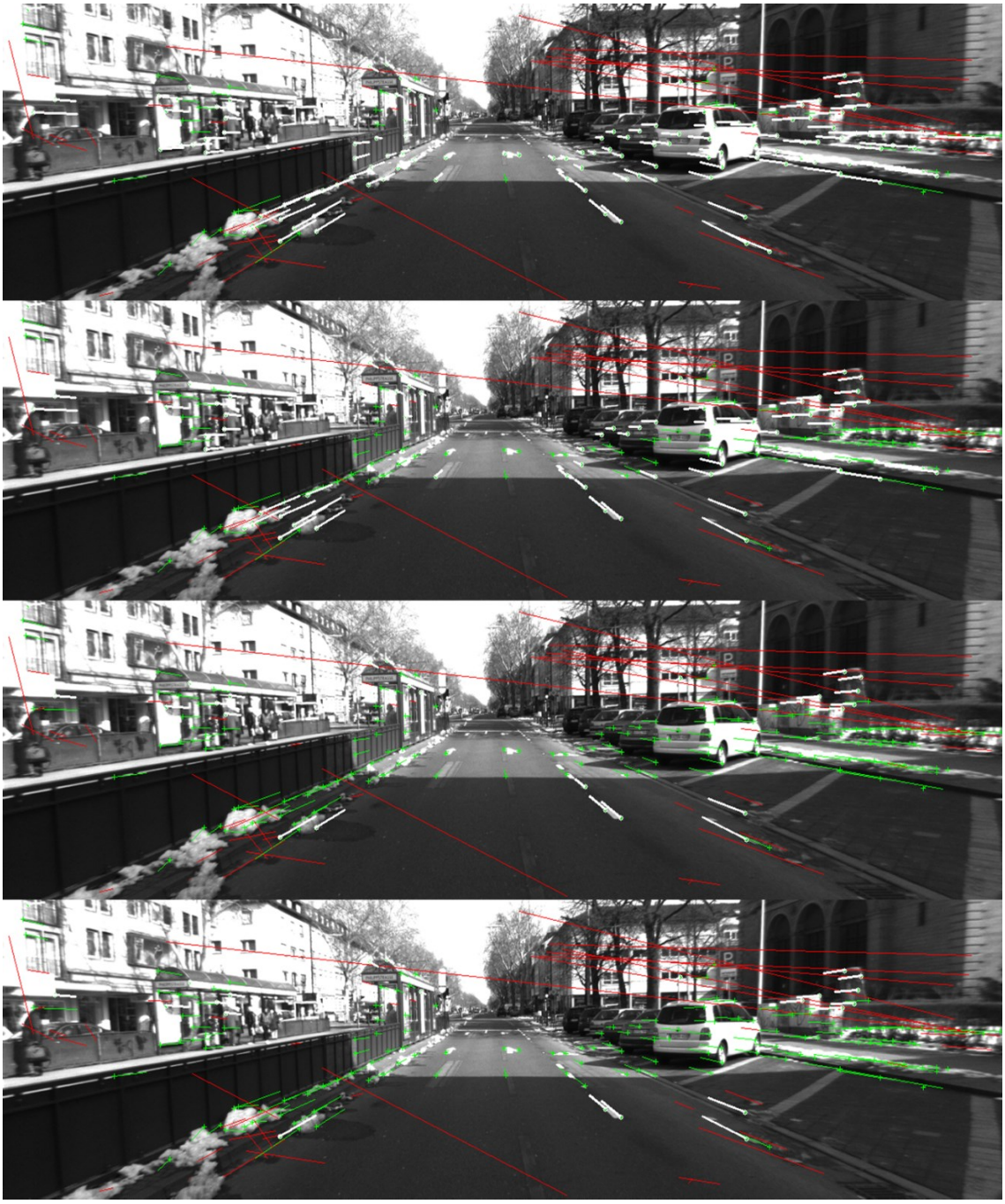


Figure 4.8. *Illustration on the current left image of the outlier rejection quality varying parameter θ. From top to bottom: θ = π/2, θ = π/2 .25, θ = π/2 75, θ = π/3.*

To cope with this eventual case, rotation *R* and translation *t* resulting from previous inter-frame (that were stored) are taken again to preserve certain continuity in the estimation. This is only an exceptional measure to be used in order to preserve the continuity of the trajectory estimation.

In Figure 4.8, it can be noticed that condition (4.16) is more restrictive even though correspondences look coherent for the large majority of them. However, as it can be seen in Figure 4.9 and 4.10, this high restriction criterion does not necessarily guarantee a better quality of the generated trajectory. In [66], authors proposed a value of $\pi/4$ for θ. However, in their case the outlier rejection algorithm was applied for a camera rotational motion of 360 °. Here, this value would be too restrictive as we can see that even with a value of θ set to $\pi/2.75$ (65.45°), motion estimation degenerates in the straight-line trajectory part. Indeed, these restrictions, ((4.15) and (4.16)), can be penalising in this case.

On the other hand, it seems that they are not penalising when the motion is slower or mainly rotational based as it was highlighted in the second part of the route. Thus, the best balance found between these two criteria in a context that try to handle all types of motion at various speeds is to fix ε to 0.5 and θ to $\pi/2$. Finally, Figure 4.11 and 4.12 show the necessity of using outliers rejection algorithm as the presence of a certain amount of them can result in a completely zigzagging trajectory.

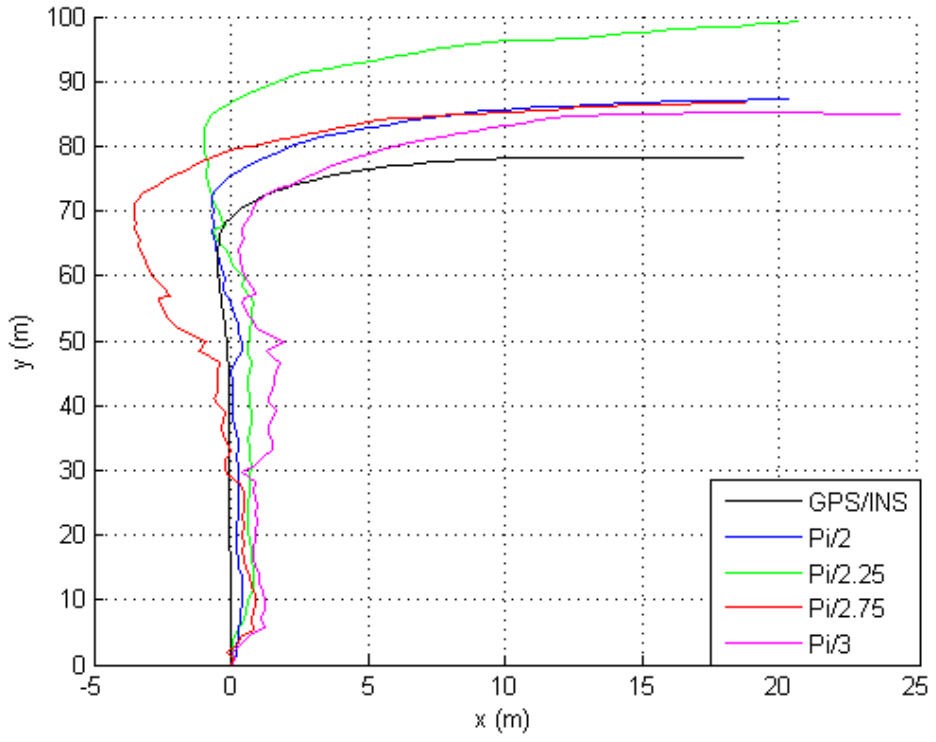


Figure 4.9. *Two dimensional plot of generated trajectories varying parameter θ.*

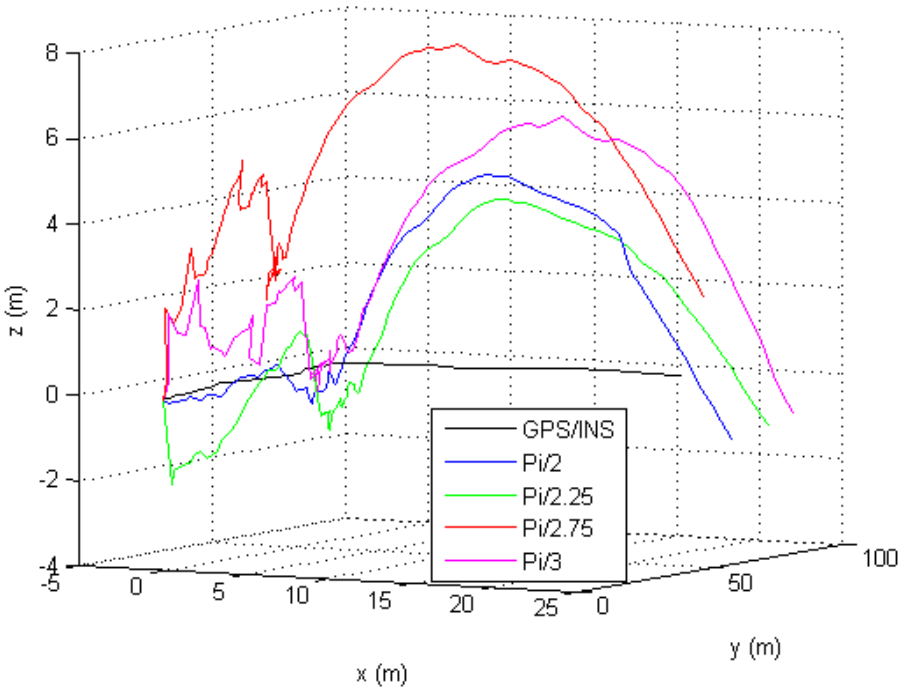


Figure 4.10. *Three dimensional plot of generated trajectories varying parameter θ.*

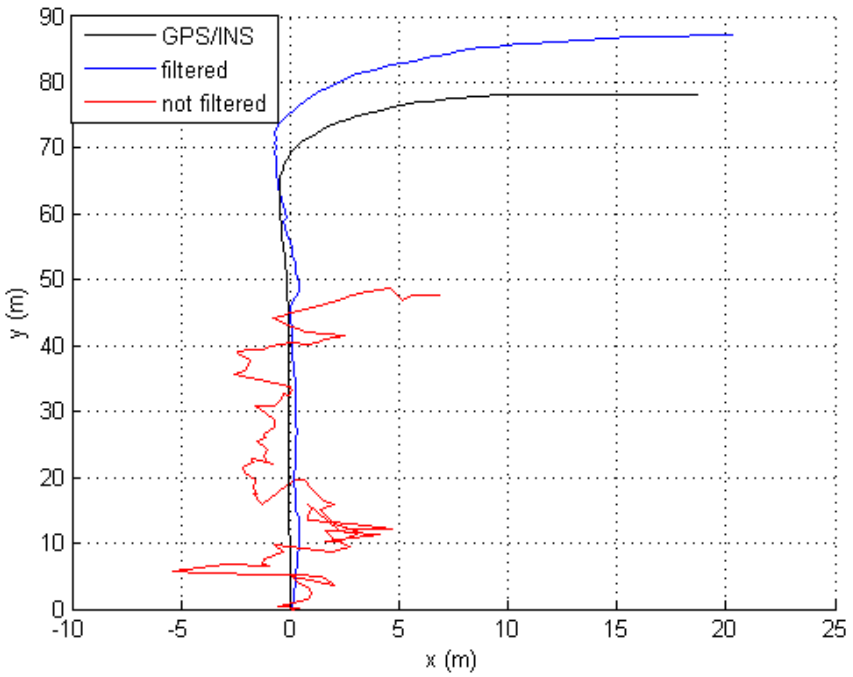


Figure 4.11. Two *dimensional plot of filtered and not filtered generated trajectories.*

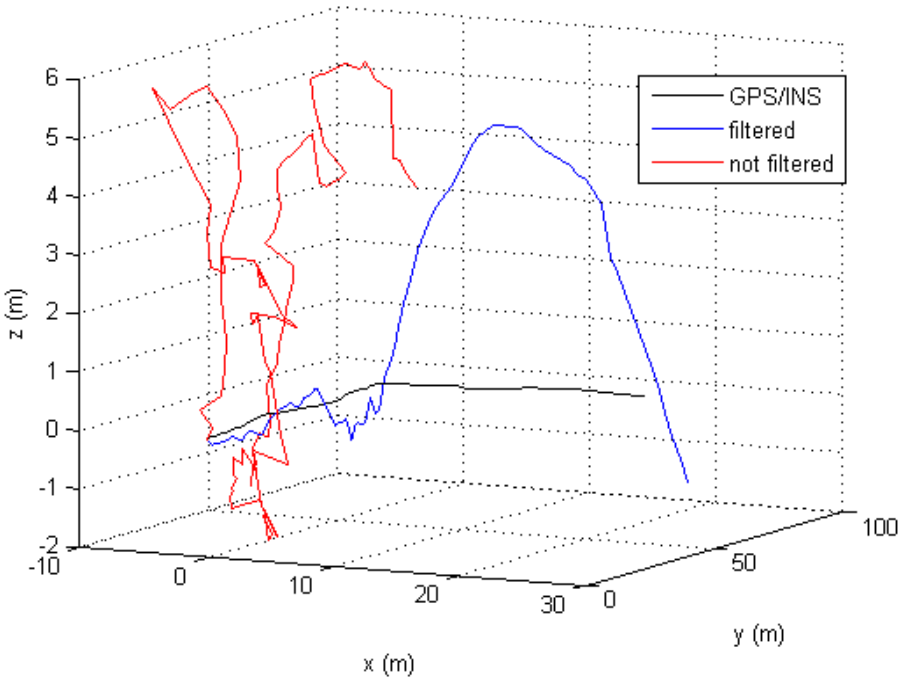


Figure 4.12. *Three dimensional plot of filtered and not filtered generated trajectories.*

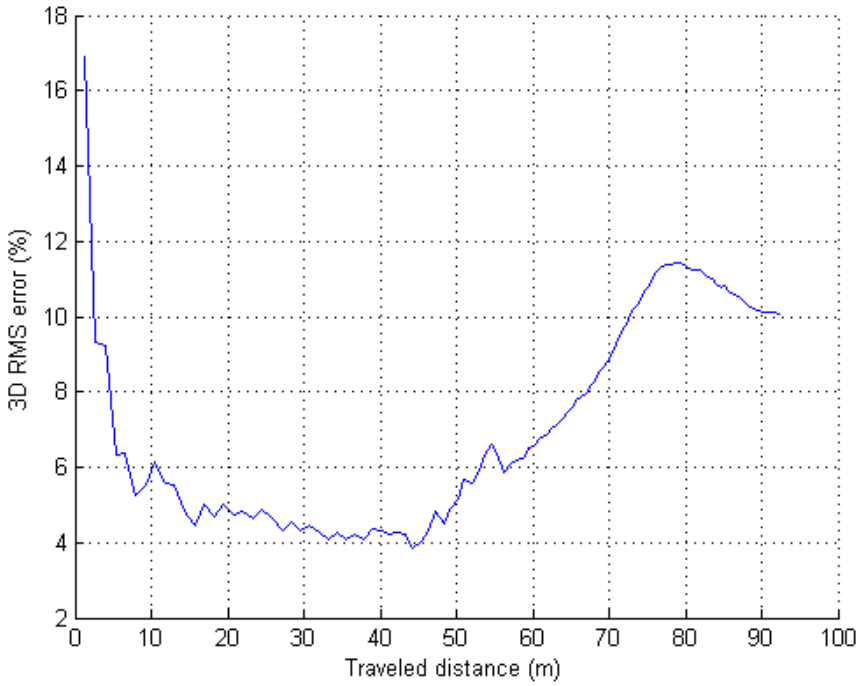


Figure 4.13. *Three dimensional relative error over the travelled distance for the filtered solution based on quaternion motion estimation algorithm.*

Motion estimation using quaternion method and based on 3D correspondences offers almost a reasonable solution, which is quite remarkable with regard to the challenging conditions of dataset [107]. Figure 4.13 shows that this solution ends with approximately 10% of 3D RMS error on a 92.2 m route which means that at the final position the error represent 10% of the distance travelled. This amount of error is mainly due to height estimation, which accumulates at the turn but also due to over estimation of forward distance in the straight line. The latter makes the trajectory turning few meters after it should have turned (see GPS/INS reference in black).

Regarding the goals fixed in this work the solution presented here is of a good value. However, it is clearly not accurate enough. Indeed, 3D point's representation suffer from relative uncertainties resulting from noise and accumulated approximations along the different stages (feature detection, feature tracking and 3D reconstruction). This is why in the next section (Section 4.4), a solution based on 3D structure to 2D image feature correspondences, which greatly decreases approximation errors is be presented.

4.4 3D Structure to 2D Image Feature Based Motion Estimation

This section details the method, which solves the motion estimation problem using spatial and image plane representations of the feature point correspondences between consecutive stereo images. In the previous Section, motion estimation, computed from 3D reconstructed correspondences from stereovision properties, showed limitations. These are mainly due to the difficulty in obtaining a precise 3D representation of the features through triangulation. In order to reach a finer level of accuracy in the representation of feature correspondences, an additional re-projection operation have to be adopted. As mentioned by Nister and al. [2], dealing with 3D structure to 2D image coordinates is more accurate than dealing with 3D structure only. Indeed, in the first case it is the image re-projection error that has to be handled while in the latter it is the 3D position error, which is more ambiguous. The process of refining the re-projection error from 3D points is called bundle adjustment (BA). BA is widely used in 3D computer vision applications and especially for structure from motion problems. Triggs et al. provided an exhaustive survey of the different utilisations of BA [67].

This Section is organised as follow: Theoretical explanation of using BA in the visual odometry context is firstly presented. Non-linear minimisation approaches based on trust region method for solving BA are then introduced. Finally, the presentation of the performances achieved on long-range outdoors routes with this new implementation is given.

4.4.1 Local Bundle Adjustment

Bundle adjustment is a large non-linear least squares problem that tries to simultaneously optimise the 3D positions, the parameters of the relative motion between the different views, and the parameters of the cameras in accordance to the image re-projection of all the feature points. This implies a very high computational cost considering all the parameters to be optimised. A lighter version of BA called local (windowed) bundle adjustment [67] proposes to take into account only *N* number of subsequent views instead of all the views in the original method. In order to further reduce the BA complexity, it is also proposed to only focus on the optimisation of the camera and relative motion parameters, while keeping fixed the 3D points.

In this section, the presented solution for motion estimation is a minimalist version of the local bundle adjustment involving only two consecutive stereo pair images. It only focuses on the optimisation of the relative motion parameters between these two views. This is in the continuity of what has been explained in the previous chapters with the target of reducing the computational burden in order to achieve real time performance. Additionally, the two views scheme fits perfectly our feature tracking strategy, which consists in not tracking features over more than two consecutive frames in order to avoid drift in image feature localization as explained in Section 2.6. The non-linear objective function to minimise is the image re-projection error function of the motion parameter vector κ expressed as follows:

$$\min \sum_{i=1}^{N} \left\| p_{cL(i)} - f(P_{p(i)}; \kappa) \right\|^2 + \left\| p_{cR(i)} - f(P_{p(i)} - B; \kappa) \right\|^2 \tag{4.17}$$

and

$$\kappa = \left[q_0 \; q_1 \; q_2 \; q_3 \; t_x \; t_y \; t_z \right]^T \tag{4.18}$$

This motion parameter vector κ to be optimised is a *1*x*7* vector which consists of the four quaternion elements for the orientation part representing $R_{(q)}$ (4.12) and the three remaining elements represent *t* for the translational part.

Non-linear re-projection function defined by *f* in equation (4.17) takes as input, P_p a 3D triangulated features from the previous stereo pair and the motion parameter κ. The link between spatial and planar representations is obtained with the help of the rectified camera matrix K_{rect} derived from (2.5) as follows:

$$\begin{cases} \begin{bmatrix} u^*_{cL} \\ v^*_{cL} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^*_c/Z^*_c \\ Y^*_c/Z^*_c \\ 1 \end{bmatrix} \\ and \\ \begin{bmatrix} u^*_{cR} \\ v^*_{cR} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (X^*_c - B)/Z^*_c \\ Y^*_c/Z^*_c \\ 1 \end{bmatrix} \end{cases} \quad \text{with} \quad \begin{cases} p^*_{cL} = \begin{bmatrix} u^*_{cL} \\ v^*_{cL} \end{bmatrix} \\ and \\ p^*_{cR} = \begin{bmatrix} u^*_{cR} \\ v^*_{cR} \end{bmatrix} \end{cases} \tag{4.19}$$

Where p^*_{cL} and p^*_{cR} are respectively the projected features in the left and right current stereo pair. The rectified camera parameters *B*, *f*, u_0 and v_0 are respectively the baseline, the focal length and the central pixel coordinates as defined in (2.3). The post motion point $P^*_c = \{X^*_c, Y^*_c, Z^*_c\}$ is computed from P_p, $R_{(q)}$ and *t* following (4.3):

$$P^*_c = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 + q_2^2 - q_1^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 + q_3^2 - q_1^2 - q_2^2) \end{bmatrix} P_p + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.20}$$

Figure 4.14 illustrates the re-projection process where (4.17) can be re-formulated as:

$$\min \sum_{i=1}^{N} \left\| \begin{bmatrix} u_{cL(i)} \\ v_{cL(i)} \end{bmatrix} - \begin{bmatrix} u^*_{cL(i)} \\ v^*_{cL(i)} \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} u_{cR(i)} \\ v_{cR(i)} \end{bmatrix} - \begin{bmatrix} u^*_{cR(i)} \\ v^*_{cR(i)} \end{bmatrix} \right\|^2 \tag{4.21}$$

As it can be seen from Figure 4.14, the objective is to reduce the pixel distance between the tracked features and their relative re-projected features.

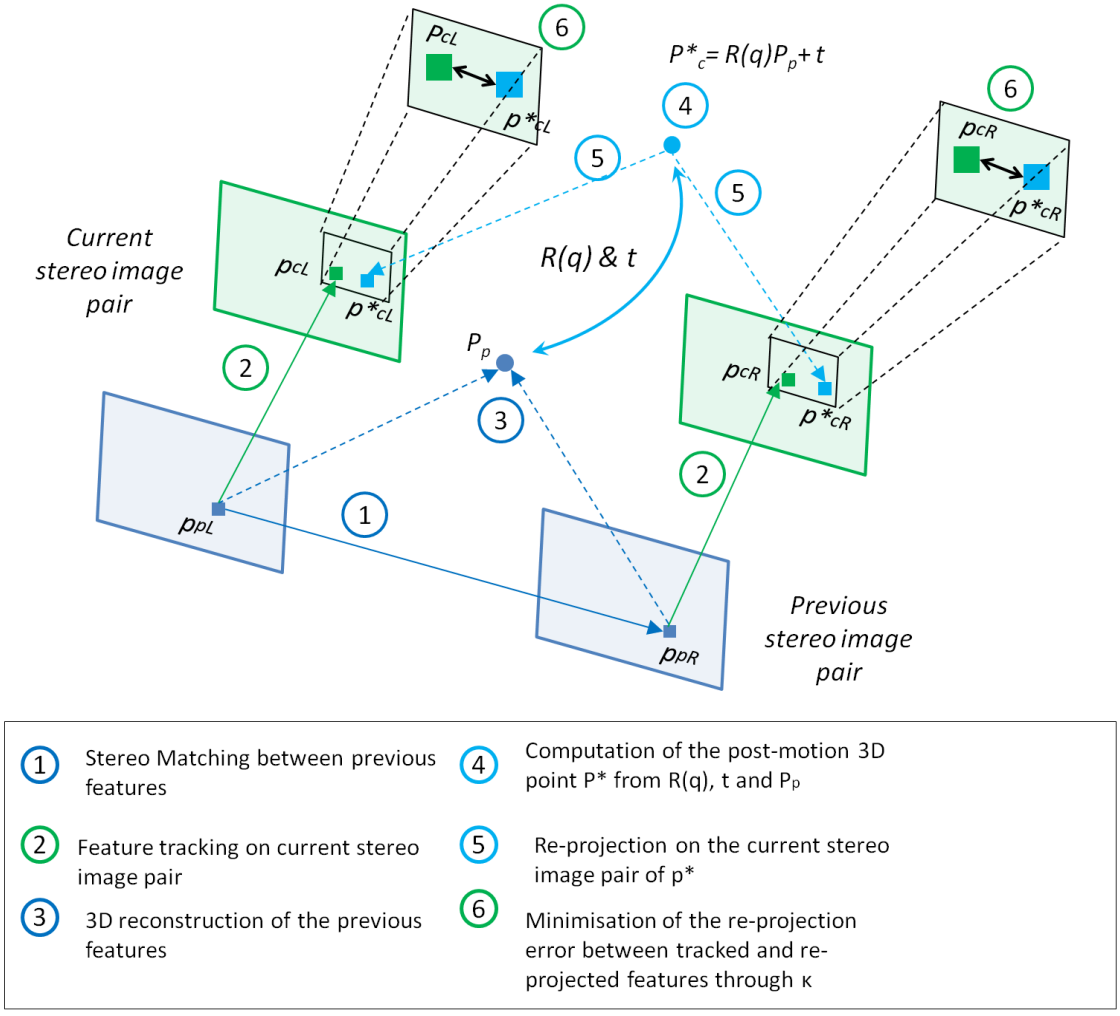


Figure 4.14. *Illustration of the re-projection process for the adopted two views local bundle adjustment scheme.*

This is done using non-linear minimisation techniques optimising the motion parameter κ, which is used to compute the rotation matrix $R_{(q)}$ and the translation vector t. By refining κ, the re-projection quality is getting better, and in the meantime the relative motion estimation becomes more accurate.

4.4.2 Minimisation

In order to reduce the approximation resulting from noisy measurements, localisation errors and objective function non linearity, κ minimises $Y(\kappa)$ defined as:

$$Y(\kappa) = \sum_{i=1}^{N} \left\| x - f(\kappa, P_p) \right\|^2 \qquad (4.22)$$

Where x refers to the tracked features, $f(\kappa)$ refers to the re-projected features as expressed in equation (4.19) and P_p refers to the triangulated 3D pervious features.

4.4.2.1 Levenberg-Marquardt

Levenberg-Marquardt algorithm is widely used to solve bundle adjustment problem [67]–[69]. This technique that was first proposed by Levenberg [70] and then Marquardt [71], is widely adopted to solve nonlinear least square problems thanks to its ease of implementation and its effectiveness. It also belongs to the trust region methods family that guaranty global convergence while avoiding the non-positive definite Hessian problem in contrary to Newton's method.

In contrary to line search optimisation methods, trust region approaches set first a maximum distance before choosing a direction. Hence, the model is trusted around a restricted area Δ, which is adjusted along iterations. If the model matches the objective function then Δ is increased, whereas it decreases if the approximation is poor.

Trust region sub-problem follows a quadratic model $\varrho(\delta)$ that approximates the objective function f (4.17) in such a way that the model is trusted within a limited region Δ_k around the current motion parameter vector κ_k:

$$\varrho(\delta) = f(\kappa) + (J^T r)^T \delta + \frac{1}{2}\delta^T (J^T J)\delta \tag{4.23}$$

with $$r = x - f(\kappa) \qquad \text{and} \qquad J = \frac{\partial f(\kappa)}{\partial \kappa} \tag{4.24}$$

Where r is the nx1 residual vector, J is the nx7 Jacobian matrix of the function $f(\kappa)$ and J^TJ is the nxn approximation of the Hessian matrix, and I the nxn identity matrix (n the number of correspondences). The solution of the objective function (4.14) is thus formulated as:

$$(J^T J - \mu I)\delta_{LM} = J^T r \tag{4.25}$$

By mean of a damping factor μ, LM algorithm switches between the steepest descent method when the current solution is far and the Gauss-Newton method if the current solution is approaching the local minimum. Algorithm 4.1 gives a description of the LM algorithm inspired from the work of [72] that was adapted to the two views local bundle adjustment scheme introduced here.

Algorithm 4.1: *Levenberg-Marquardt minimisation process*

Input: $\{p_{cL}, p_{cR}\}$, $\{P_p\}$, κ_0
Output: κ_{optim}

Set: $\upsilon = 2$, $\xi = 1e^{-3}$, $k = 0$, $\rho = 1$, $\mu = \xi * max(A_{ii})$, $\chi = 1e^{-3}$, $maxiter = 50$;

 # start of optimization task
while(not converged and not *maxiter*)
if($\rho > 0$)
Compute:
$A = J^T J$, $b = J^T r$, $||b||$, $||\kappa_k||$ and $||x - f(\kappa_k, P_p)||$

if($||b||_\infty < \chi$ or $||r|| < \chi$)
$\kappa_{optim} = \kappa_k$
converged
endif

$\mu = \mu * max(1/3, 1-(2\rho-1)^3)$
$\upsilon = 2$
else
$\mu = \mu * \upsilon$
$\upsilon = 2 * \upsilon$
endif

Compute δ_{LM} (4.25)

if($||\delta_{LM}|| < \chi ||\kappa_k||$)
$\kappa_{optim} = \kappa_k$
converged
else
Compute:
$\kappa_{new} = \kappa_k + \delta_{LM}$
$A = J^T J$, $b = J^T(x - f(\kappa_{new}, P_p))$
$\rho = (||r||^2 - || x - f(\kappa_{new}, P_p)||^2)/(\delta^T_{LM} (\mu \delta_{LM} + b))$

if($\rho > 0$)
$\kappa_k = \kappa_{new}$
$k=k+1$
endif
endif

endwhile

4.4.2.2 Double Dogleg

In this work, instead of using Levenberg-Marquardt algorithm, we decided to adapt double Dogleg trust region method [73], which is a variant of the Dogleg algorithm to solve the bundle adjustment for motion estimation. In [74], it has already been shown that the use of Dogleg [75] trust region technique presents advantages in term of computational cost compared to Levenberg-Marquardt methods for full bundle adjustment applied to 3D structure reconstruction only.

Dogleg algorithm is delineated by two lines composed of the steepest descent direction and the Newton point direction (see Figure 4.15). The optimal trajectory follows the steepest descent direction until reaching the Cauchy point (C.P) then converges to the Newton point passing by the Dogleg step. This later should be intersecting with the trust region boundary Δ. By introducing an intermediate Newton step N between the Cauchy Point and the actual Newton point, the behaviour of the Double Dogleg algorithm presents a further improvement. Indeed, the optimal curve trajectory crosses the trust region before original Dogleg. This direct control between these two lines (Steepest descent and Newton) by the mean of trust region (characterised by Δ) gives a faster optimisation to the algorithm and is also the main difference with Levenberg-Marquardt algorithm [76].

Solving (4.23) is not straightforward. In the case where the unconstrained solution δ_N (Gauss-Newton step) is too long, the convergence trajectories are ruled out by the following equations:

For Dogleg:

$$\delta_{DL} = \delta_{CP} + \lambda(\delta_{CP} - \delta_N) \tag{4.26}$$

For double Dogleg:

$$\delta_{DDL} = \delta_{CP} + \lambda(\beta\delta_{CP} - \delta_N) \tag{4.27}$$

with $\beta = 0.8\gamma + 0.2$ ($\gamma \in [0,1]$) is an adjusting parameter that fixes the position of the intermediate Newton step N in the Newton direction for the double Dogleg, (see Figure 4.15).

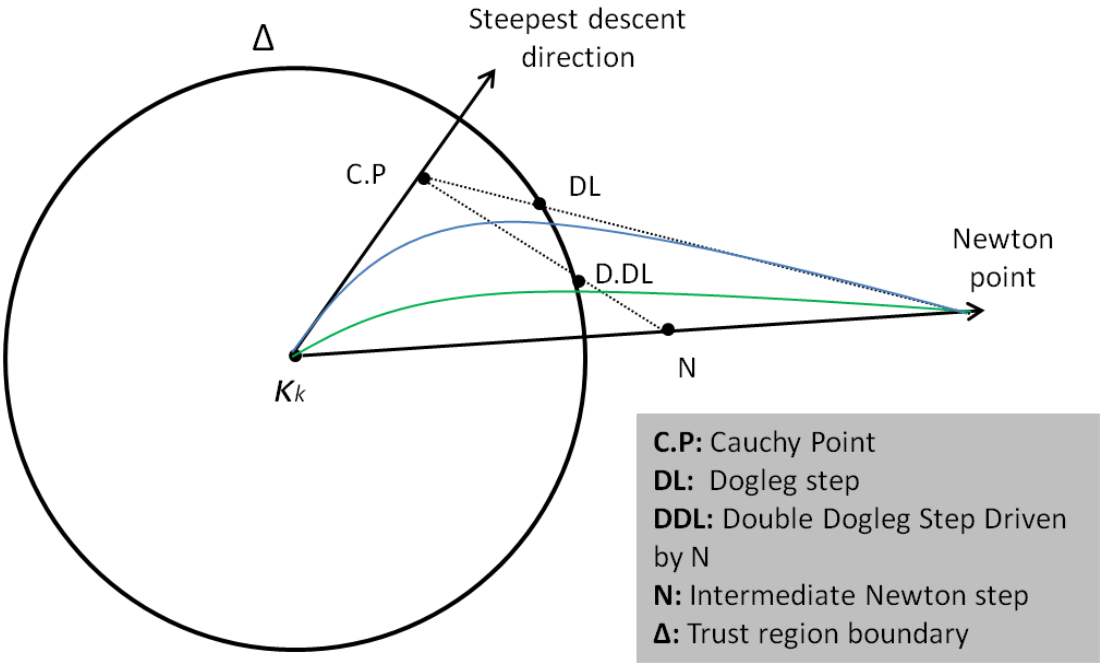


Figure 4.15. *Illustration of Dogleg and double Dogleg convergence curves.*

The minimiser for the Cauchy point δ_{cp} and the Gauss-Newton step δ_N are obtained using these equations:

$$\begin{cases} \delta_{CP} = \frac{\|b^T b\|}{\|b^T A b\|} b \\ \quad and \\ \delta_N = A^{-1} b \end{cases} \tag{4.28}$$

where approximated Hessian $A = J^T J$ and the gradient $b = J^T(x\text{-}f(\kappa))$ with the Jacobian $J = [\frac{\partial q}{\partial \kappa} \ \frac{\partial t}{\partial \kappa}]^T$ and $f(\kappa)$ is the objective function defined in (4.17).

Depending on the method chosen, λ in (4.26) or (4.27) must achieve:

$$\|\delta_{DL}\| = \Delta \quad \text{or} \quad \|\delta_{DDL}\| = \Delta \tag{4.29}$$

Algorithm 4.2: *Double Dogleg minimisation process*

Input: $\{p_{cL}, p_{cR}\}$, $\{P_p\}$, κ_0
Output: κ_{optim}
Set: $\Delta_k = 1$, $\beta = 0.6667$, $\eta_1 = 0.15$, $\eta_2 = 0.75$, $\gamma_1 = 0.5$, $\gamma_2 = 2$, $\rho_0 = 1$, $\chi = 1e^{-4}$, k = 0, maxiter =50;
 # start of optimization task
while(not converged and not *maxiter*)
    if($\rho_k > 0$)
        Compute:
        $A = J^T J$, $b = J^T r$, $\|b\|$, $\|\kappa_k\|$, $\|\|x - f(\kappa_k, P_p)\|$ and $\|\delta_{CP}\|$ (4.28)
        G.N (Gauss-Newton) = false
            if($\|b\|_\infty < \chi$ or $\|r\| < \chi$)
                $\kappa_{optim} = \kappa_k$
                converged
            endif
    endif
    if($\|\delta_{CP}\| > \Delta_k$)
        $\delta_{DDL} = -(\Delta_k / \|\delta_{CP}\|) * \Delta_k$
    else
        if(not G.N)
            Compute δ_N (4.28)
        endif
        if($\|\delta_N\| \leq \Delta_k$)
            # take Gauss Newton direction
            $\delta_{DDL} = \delta_N$
        else
            Compute δ_{DDL} (4.27)
        endif
        if($\|\delta_{DDL}\| < \chi \|\kappa_k\|$)
            $\kappa_{optim} = \kappa_k$
            converged
        else
        Compute:
        $\kappa_{new} = \kappa_k + \delta_{DDL}$
        $A = J^T J$, $b = J^T(x - f(\kappa_{new}, P_p))$ *and* ρ_k (4.31)

        if($\rho_k > 0$)
            $\kappa_k = \kappa_{new}$ (4.30)
            k=k+1
        endif

        update trust region boundary Δ (4.32)
    endif

endwhile

Thus, in the case the chosen method is double dogleg for instance, the current point κ_k is updated according to the constrained trust region as:

$$\kappa_{k+1} = \begin{cases} \kappa_k - \frac{\Delta_k}{\|\delta_{CP}\|} & if\ \|\delta_{CP}\| \geq \Delta_k \\ \kappa_k + \delta_{DDL} & if\ \|\delta_{CP}\| < \Delta \text{ and } \|\delta_N\| > \Delta_k \\ \kappa_k + \delta_N & if\ \|\delta_{CP}\| < \Delta \text{ and } \|\delta_N\| \leq \Delta_k \end{cases} \tag{4.30}$$

After the calculation of the new point κ_{k+1}, the trust region Δ_{k+1} is then updated according to the reduction ratio ρ_k between the actual residual r_{act} and the predicted residual r_{pred} defined below:

$$\rho_k = \frac{r_{act}}{r_{pred}} = \frac{f(\kappa_k) - f(\kappa_{k+1})}{\varrho_k(0) - \varrho_k(\delta_{DDL})} \tag{4.31}$$

$$\Delta_{k+1} = \begin{cases} \gamma_1 \Delta_k & if \rho_k < \eta_1 \\ \Delta_k & if \eta_1 \leq \rho_k < \eta_2 \\ \gamma_2 \Delta_k & if \rho_k \geq \eta_2 \end{cases} \tag{4.32}$$

Where $0 < \gamma_1 < 1 < \gamma_2$ and $0 < \eta_1 \leq \eta_2 < 1$. The value of Δ_k is increased or decreased according to the quality of the model approximation of the objective function f (4.17). The algorithm converges when $\|b\| \leq \varepsilon$.

Algorithm 4.2 gives a description of the double Dogleg algorithm adapted to the presented two views local bundle adjustment scheme.

4.4.3 Motion Estimation Process

Let us consider $s = \{ p_{pL(i)}, p_{pR(i)}; p_{cL(i)}, p_{cR(i)} \}$ the set holding all the feature correspondences linking consecutives stereo image pairs (*i = 1, ..., n, n the number of correspondences*). The presented motion estimation algorithm is implemented on a RANSAC based scheme as a part of an efficient inliers selection strategy. At each step, it starts with the selection from the set s of 3 random pairs of initial features and their corresponding tracked features. These 3 random pairs form the set $s_{RNG} = \{ p_{pL(j)}, p_{pR(j)}; p_{cL(j)}, p_{cR(j)} \}(j = 1, 2, 3)$. The motion parameters of κ is initialised to $\kappa_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$ as this algorithm does not need a specific initial guess of the motion parameters to converge, especially when the quality of the feature pairs is good.

Algorithm 4.3: *RANSAC based motion estimation routine*

Input: s
Output: κ_{final}, $s_{inliers}$
Set: $\tau = 5$;
 # start of RANSAC routine for motion estimation
for k=0 to 50
    $N_k = 0$
    form the set s by getting 3 random pairs from s
    initialise $\kappa_k = \kappa_0$
    # call the chosen optimisation process
    κ_{optim} = DoubleDogleg(s_{RNG}, κ_k,) (Algorithm 4.2)
    or
    κ_{optim} = LevenbergMarquardt(s_{RNG}, κ_k,) (Algorithm 4.1)
    if(converged)
        # get the inliers correspondences
        for each pair from s do
            if($\| p_{cL} - p^*_{cL} \| + \| p_{cR} - p^*_{cR} \| < \tau$)
                $N_k = N_k + 1$
            endif
        endfor
        if($N_k > N_{best}$)
            #create an feature the set $s_{inliers}$ from the N_k inliers
            $N_{best} = N_k$
            $\kappa_{best} = \kappa_{optim}$
        endif
    endif
endfor

 # refinement stage using κ_{best} $s_{inliers}$
κ_{final} = DoubleDogleg($s_{inliers}$, κ_{best},) (Algorithm 4.2)
    or
κ_{final} = LevenbergMarquardt($s_{inliers}$, κ_{best},) (Algorithm 4.1)

Then, the minimisation method (Algorithm 4.1 or 4.2 depending on the chosen technique) is called taking as input the motion parameter κ_k and s_{RNG} from which $P_{p(j)}$ is the 3D position, obtained by triangulation, of the previous features. In the case where the optimisation methods led to a converging solution, the resulting estimation of κ_k for the current set s of random points is assessed for the whole set s. Each feature correspondence from which the re-projection error falls under a fixed threshold τ is considered as inlier.

At the end of the RANSAC routine, κ_{best} takes the value of the motion parameter estimation κ_k giving the largest amount of inliers. This enables us to form a new set of filtered feature correspondences $s_{inliers}$.

A refinement step runs the minimisation method (Algorithm 4.1 or 4.2 depending on the chosen technique) using only the set of inliers $s_{inliers}$ obtained from κ_{best}. This gives the final solution κ_{final} from which the inter-frame rotation $R_{(q)}$ and translation t are computed. Relative motion estimations are then accumulated along the platform's (vehicle's) route to reconstitute the whole travelled trajectory. Algorithm 4.3 describes the RANSAC based motion estimation routine.

This RANSAC based motion estimation routine optimises the motion parameter κ and at the same time discards all possible outliers that were contained in the set s. In addition to be very efficient by iteratively processing a small set of 3 random pairs, this routine showed an impressive robustness against outliers especially the ones belonging to dynamic objects such as vehicles, pedestrians, trams…etc. Knowing how much the outliers can be misleading, and especially how dramatic wrong inter-frame relative motion estimation can affect the whole generated trajectory, it is very valuable to have motion estimation routine that one can rely on. This is regardless of the behaviour of the scene content.

4.4.4 Results

In this sub-section, results of the visual odometry algorithm following the RANSAC based motion estimation routine (Algorithm 4.3) are presented. The algorithm is run on different outdoor urban environment datasets [107] including the one (dataset 2010_03_09_drive_0023) presented in the previous sub-section 4.4.3. In the first instance, trajectory comparison on dataset 2010_03_09_drive_0023 is given between 3D to 3D structure based quaternion motion algorithm and the 3D structure to 2D image plane approach based on Algorithm 4.3. Then, we will only focus on the lastly presented technique. However, this time we will evaluate the quality of the generated trajectories on different datasets depending on the minimisation method that is called in the RANSAC motion estimation routine. This includes Levenberg-Marquardt (LM) and double Dogleg (DDL).

4.4.4.1 Quaternion vs Local bundle Adjustment

In this Sub-section, we demonstrate the greater accuracy of using spatial to planar motion estimation methods than 3D structure only. Figure 4.16 and 4.17 show a comparison between trajectories respectively obtained with the quaternion motion estimation methods in Section 4.3 and local bundle adjustment (LBA) method using double Dogleg (DDL) minimisation. As it can be seen, the trajectory generated with LBA accumulates less error along the route and remains closer to the ground truth especially at the turning part.

However the most significant characteristic of LBA compared to the quaternion method, is in the estimation of the height. The three dimensional comparison plots in Figure 4.17 are showing this. Indeed, the benefit of using the re-projection error instead of the 3D position is obvious here.

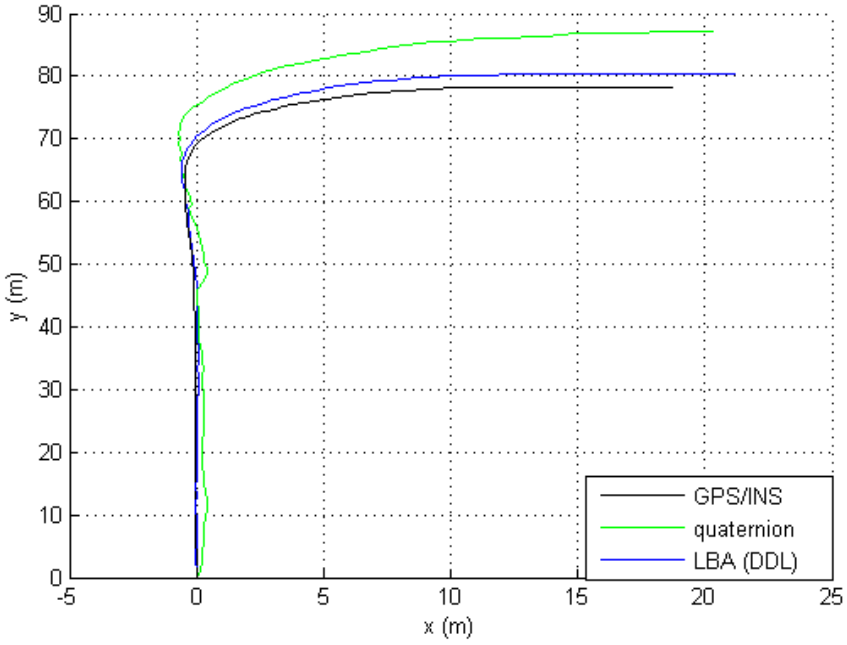


Figure 4.16. Two *dimensional plot of quaternion based and LBA based generated trajectories.*

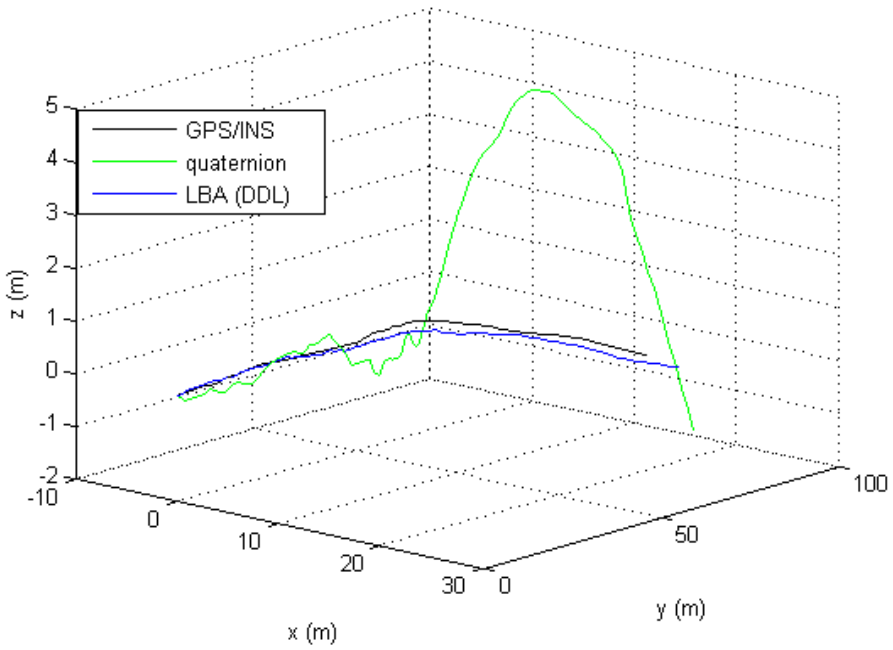


Figure 4.17. Three *dimensional plot of quaternion based and LBA based generated trajectories.*

The trajectory generated with LBA looks very similar to the GPS/INS. Figure 4.18 shows the 3D RMS error along the full route by the two methods. As it can be seen, the trajectory generated with LBA remains at low percentage (< 4%) of relative error especially at the beginning of the route. On the other hand, the trajectory generated using quaternion has evidently a higher percentage of relative 3D error. This shows the great advantage and accuracy of solving this problem from a spatial to a planar representation. The approximation resulting from the 3D positions are efficiently eliminated as shown in this case.

The 3D relative error (as shown in Figure 4.18) is calculated by taking the quotient of the root of the square difference of the 3D positions between the GPS/INS reference and the generated trajectory over the travelled distance until this point. This quotient is then multiplied by 100 to get the percentage.

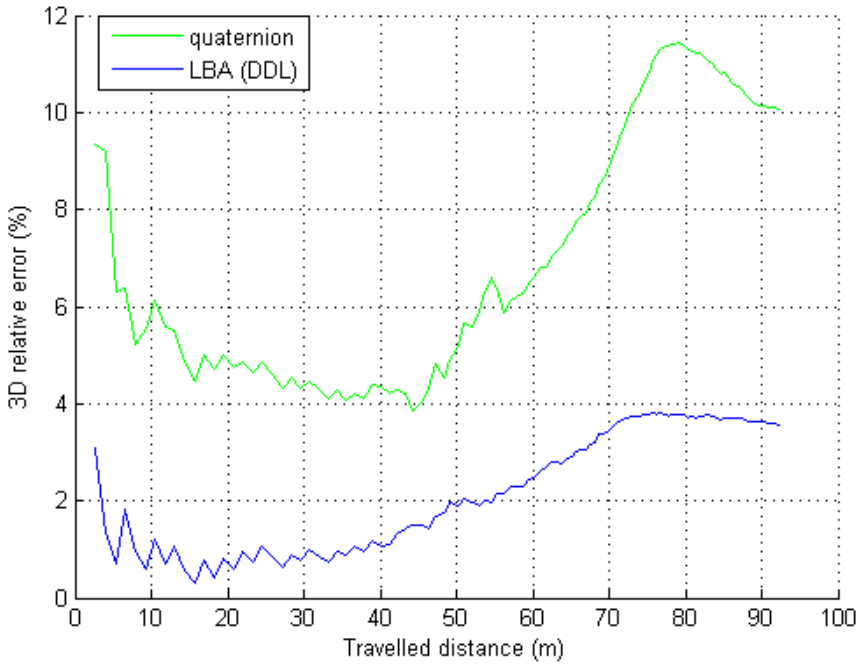


Figure 4.18. *Relative 3D squared error in percent over the travelled distance regarding GPS/INS reference trajectory.*

4.4.4.2 Local bundle Adjustment Minimisation Evaluation

In this sub-section, we focus on the evaluation of the minimisation techniques (Algorithm 4.1 and 4.2) used in the LBA motion estimation routine (Algorithm 4.3). The aim here is to demonstrate the benefit of the utilisation of the double dogleg method in a visual odometry context instead of the widely used Levenberg-Marquardt (LM) algorithm. The evaluation is led on several outdoors urban datasets [107] presenting different challenging situations for visual motion estimation process such as dynamic objects (vehicles, pedestrians, trams), long term stops with or without moving dynamic objects.

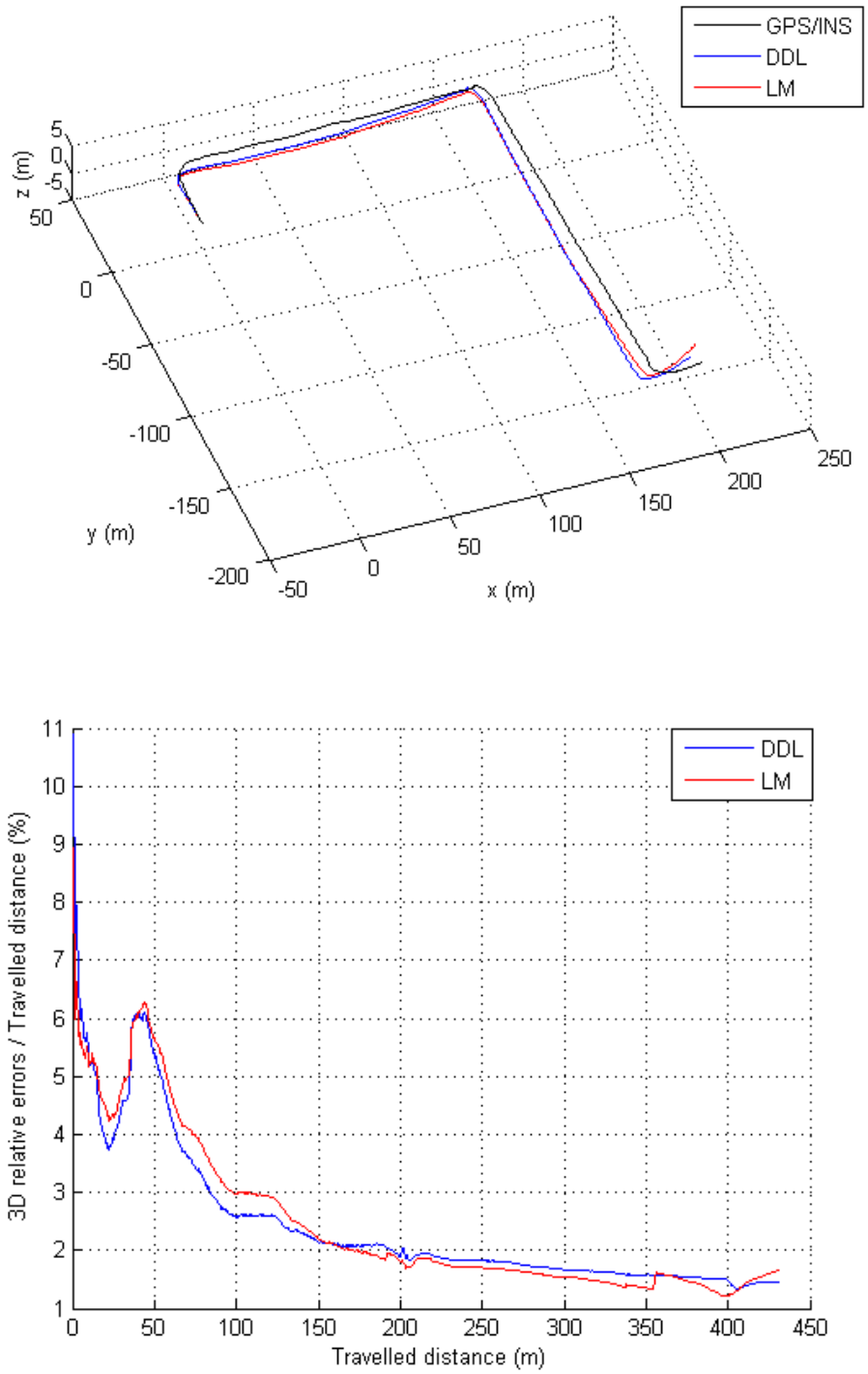


Figure 4.19. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2009_08_09_drive_0010.*

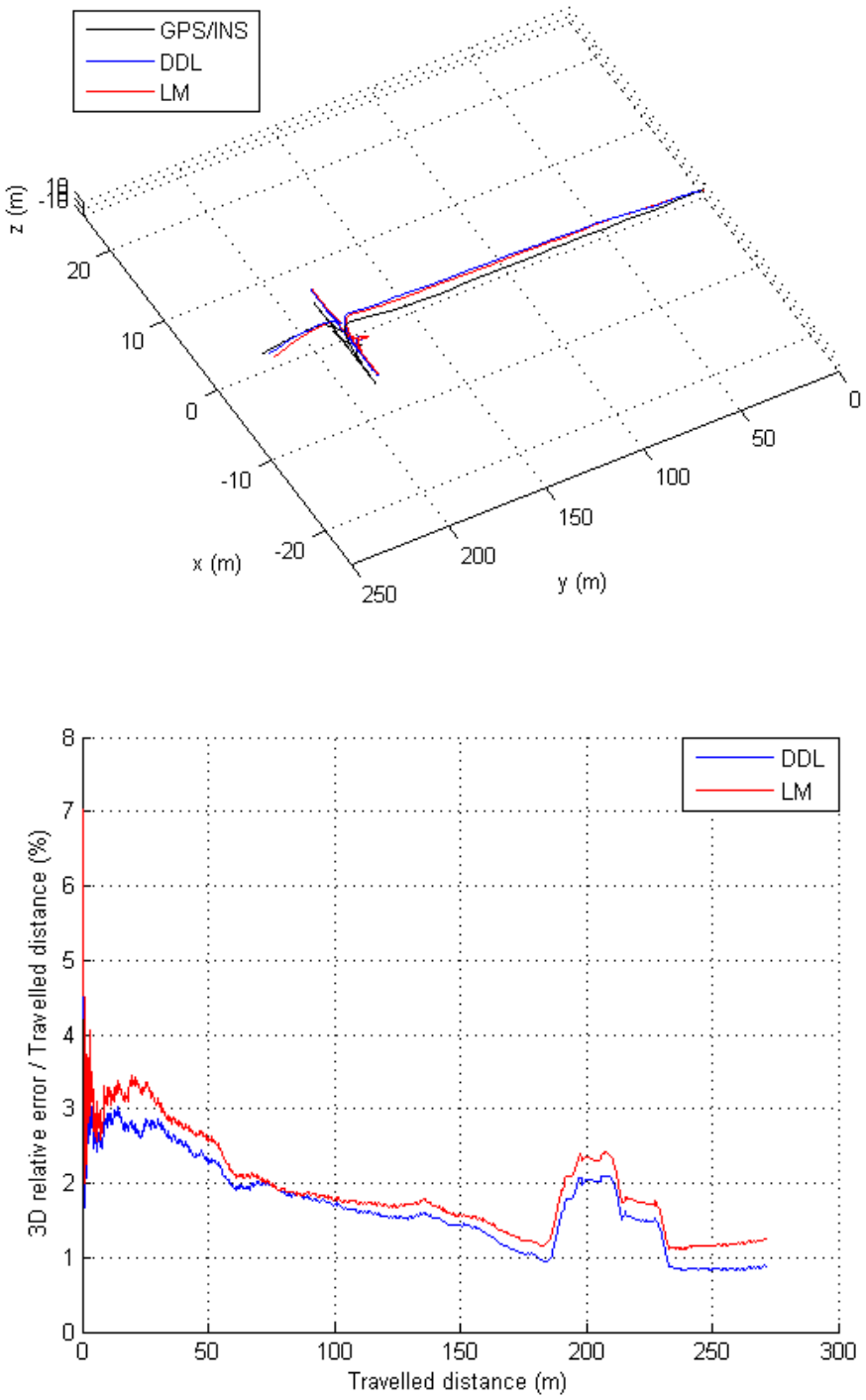


Figure 4.20. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2009_08_09_drive_0021.*

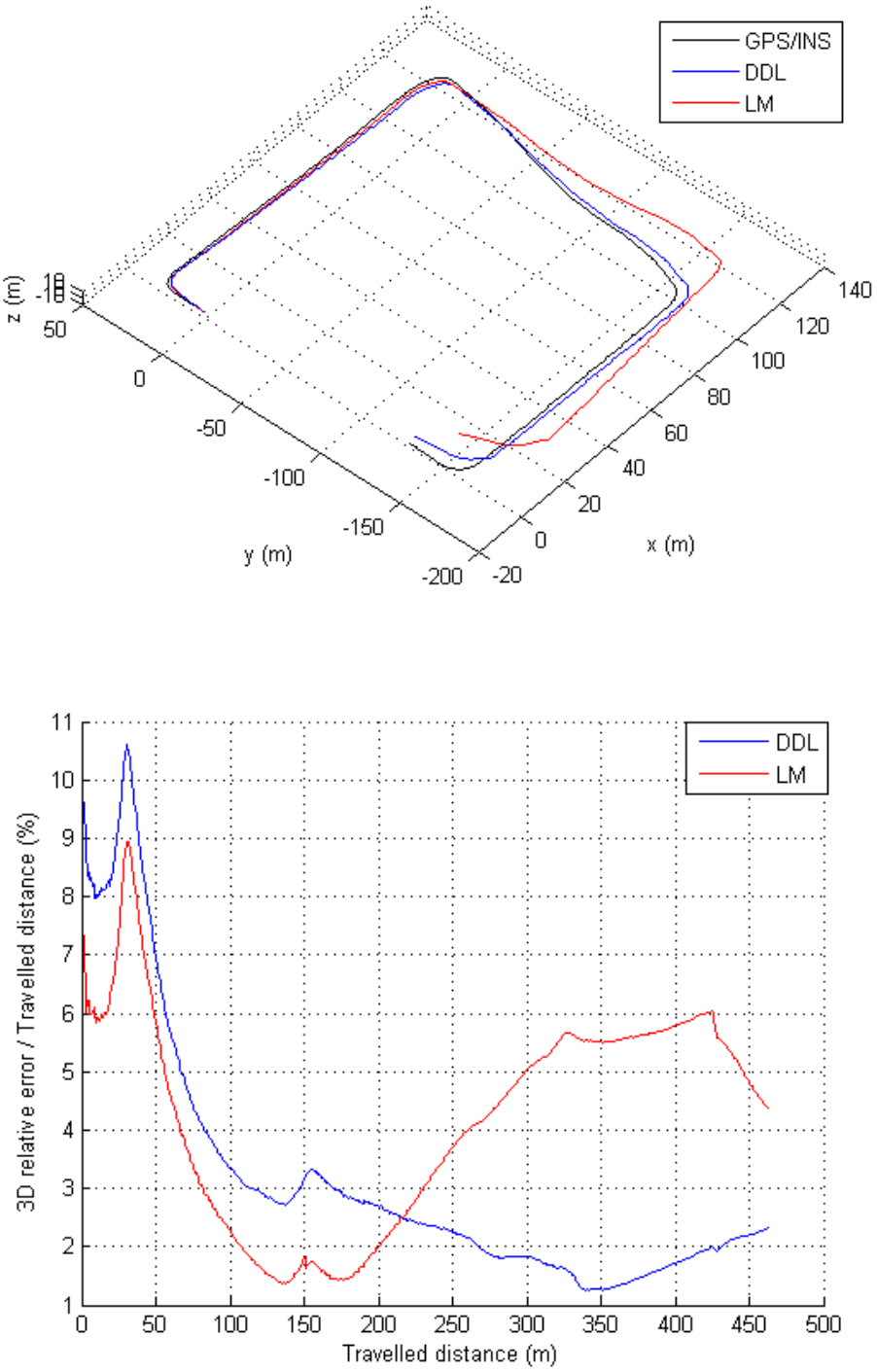


Figure 4.21. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2009_12_14_drive_0051.*

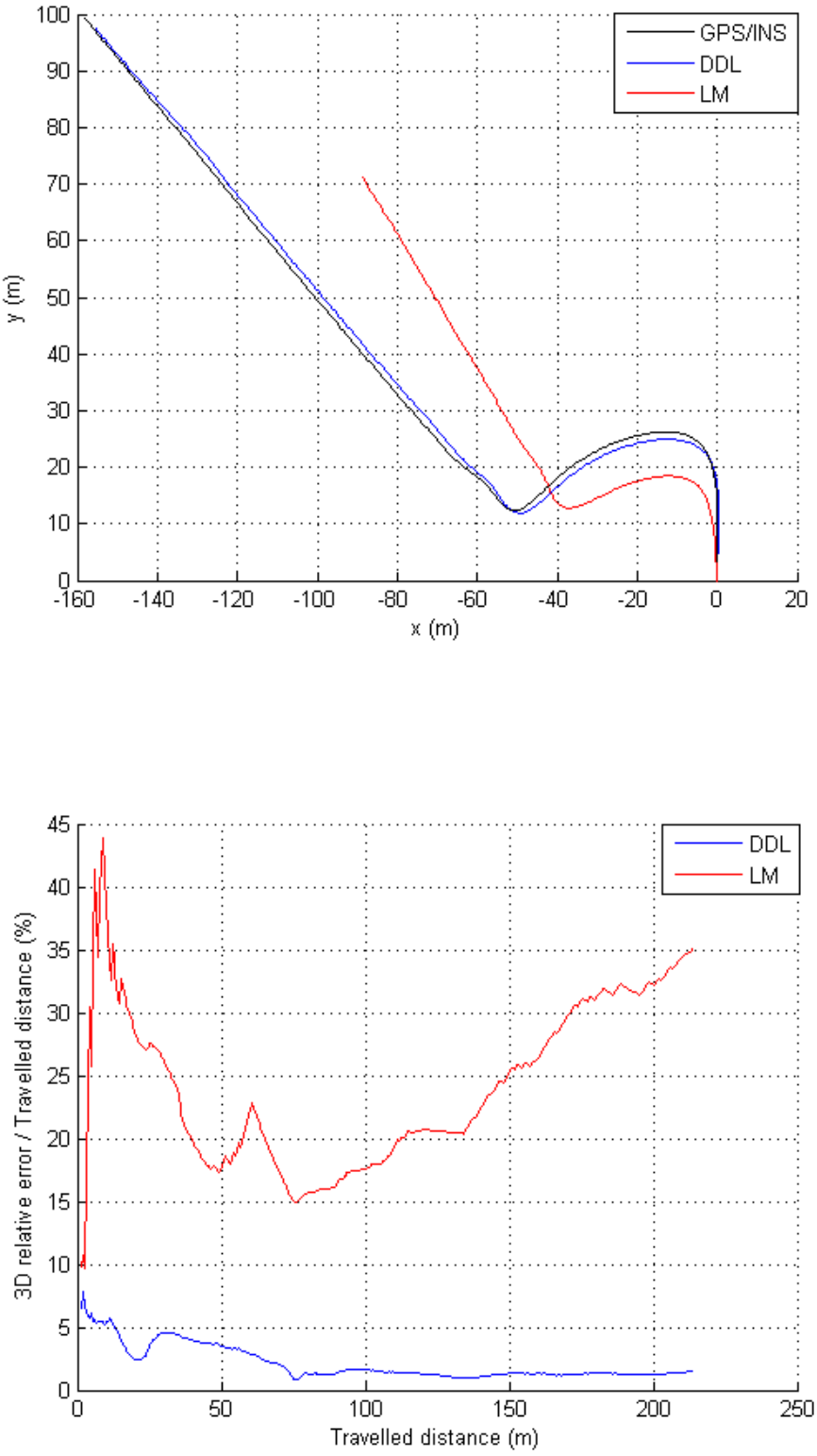


Figure 4.22. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_04_drive_0033.*

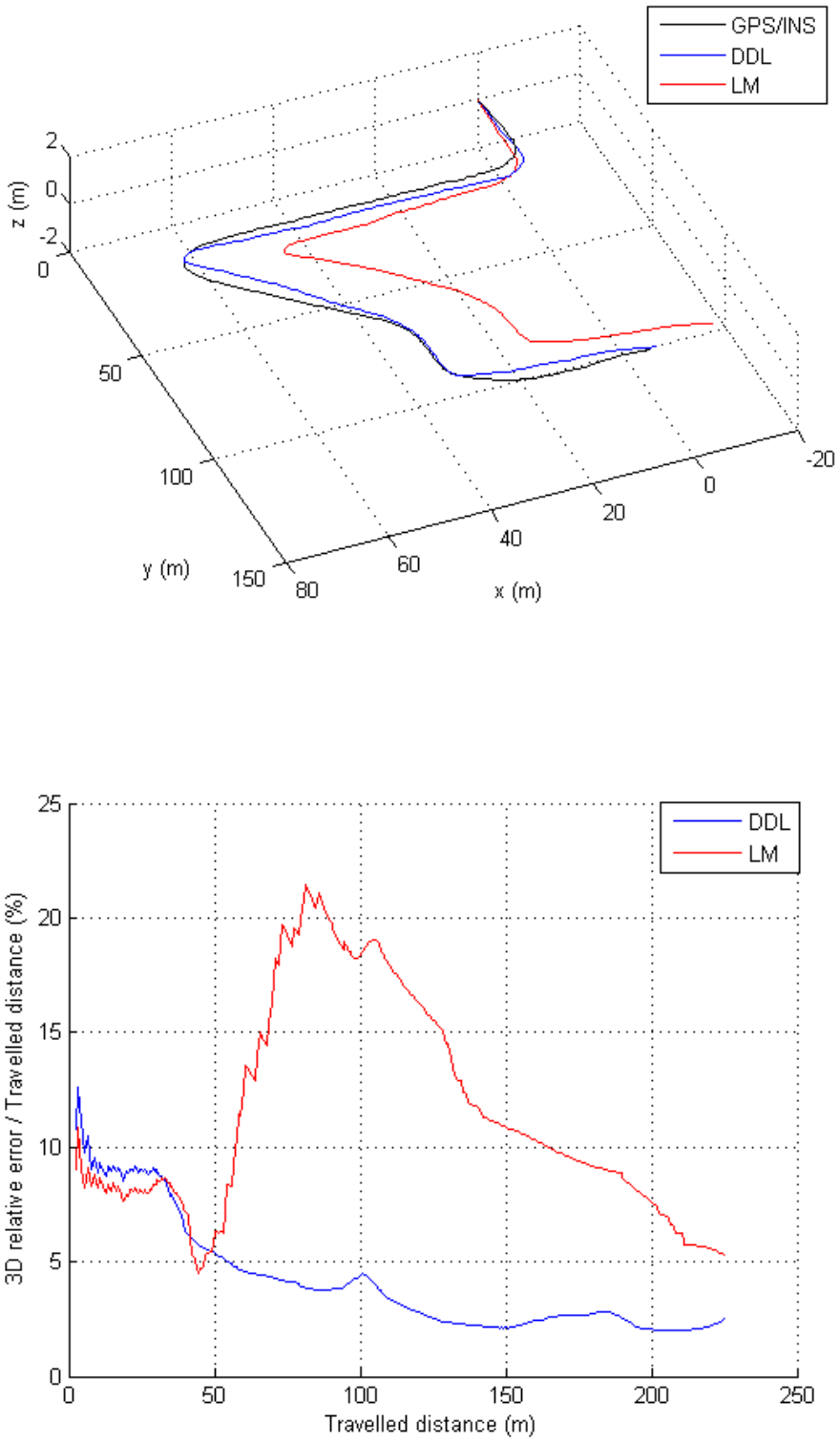


Figure 4.23. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0019.*

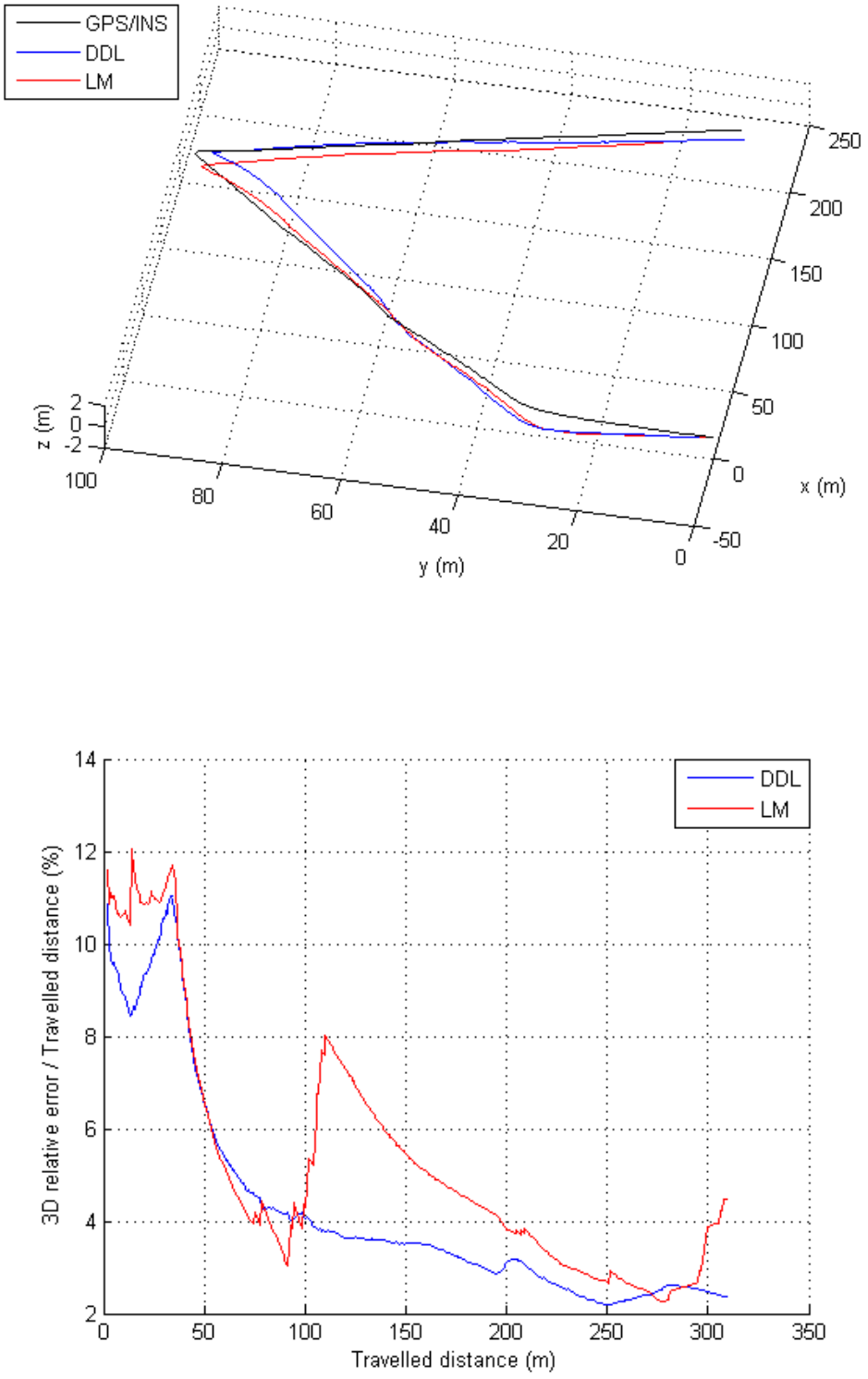


Figure 4.24. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0020.*

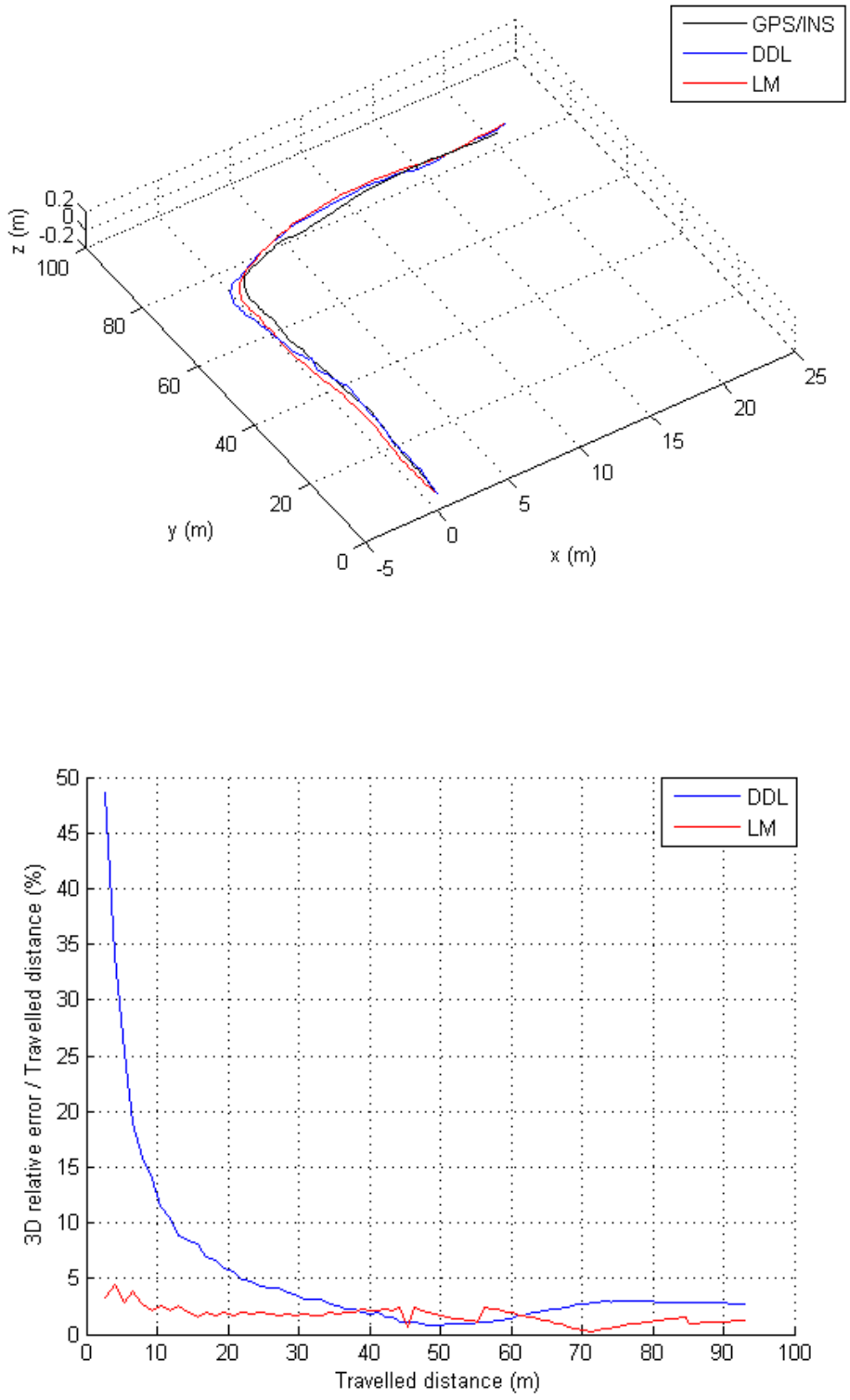


Figure 4.25. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0023.*

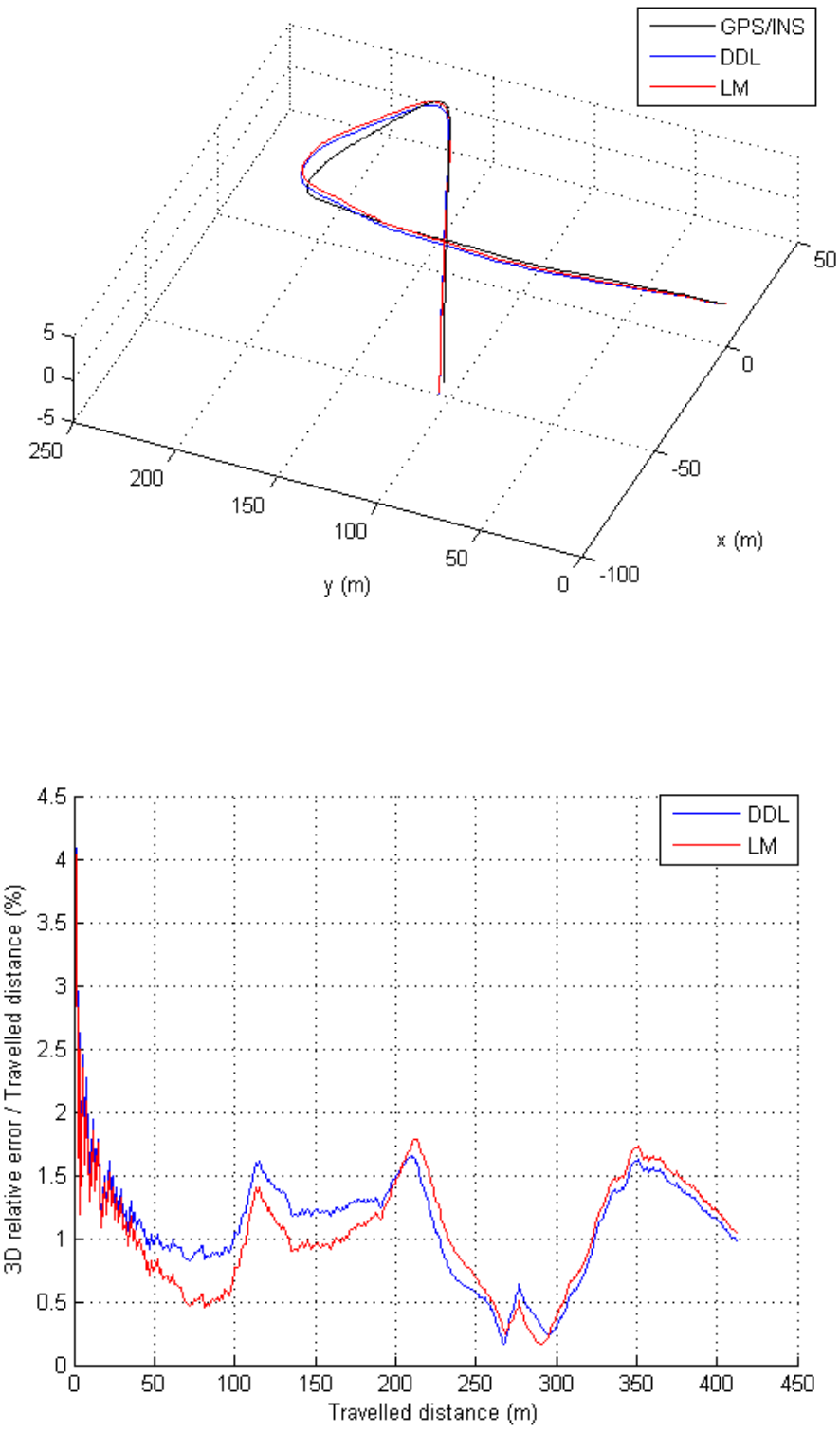


Figure 4.26. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0051.*

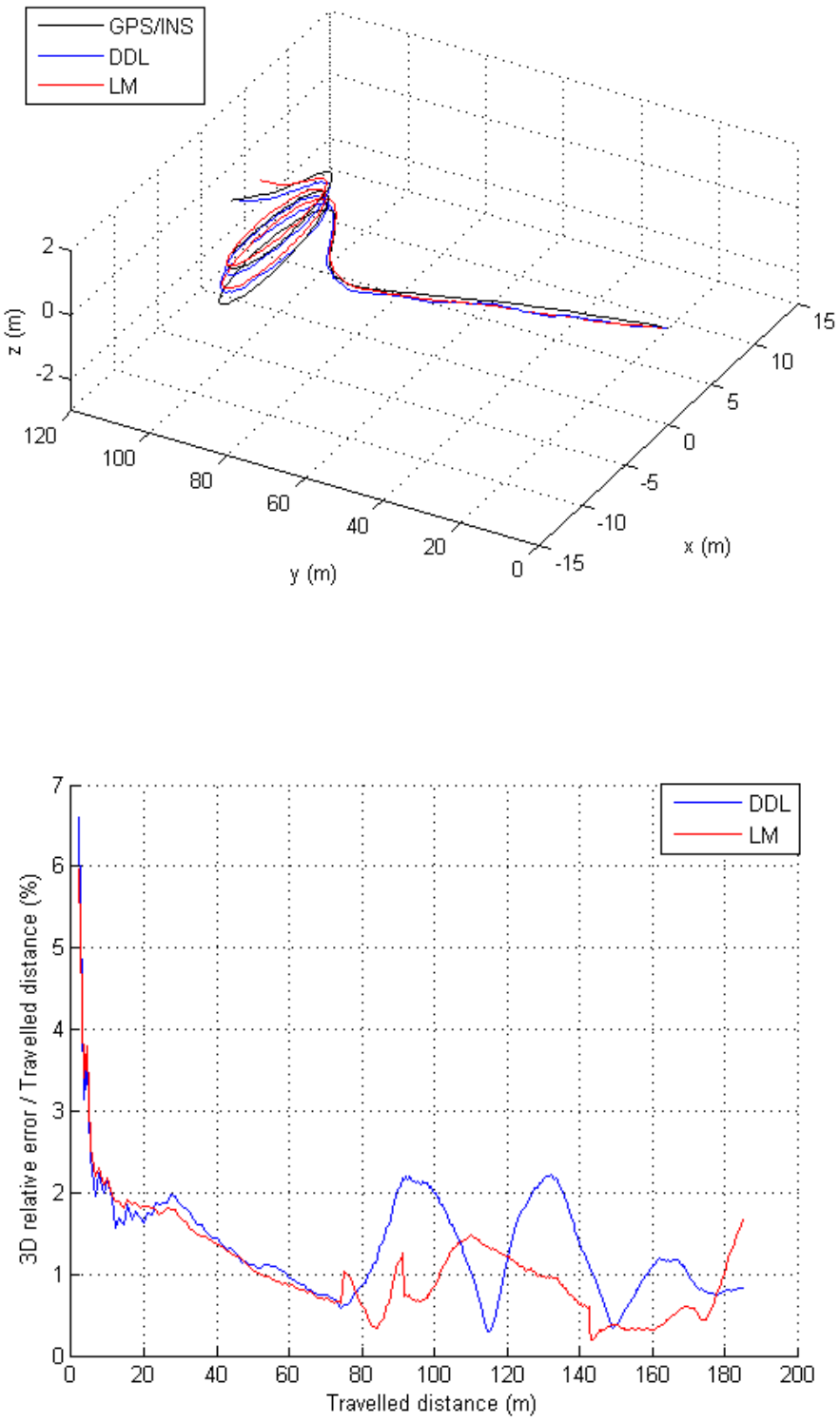


Figure 4.27 Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0081.*

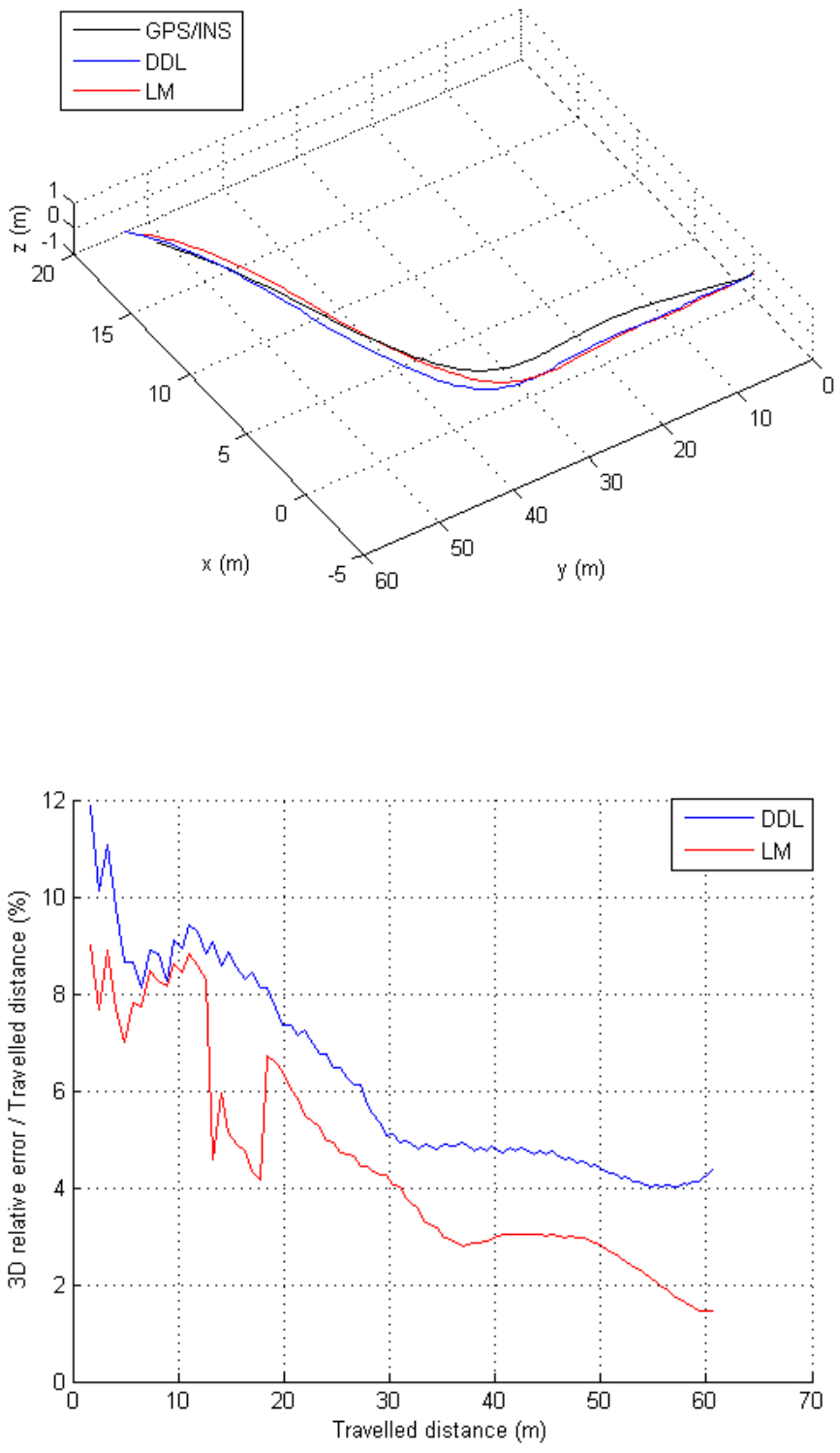
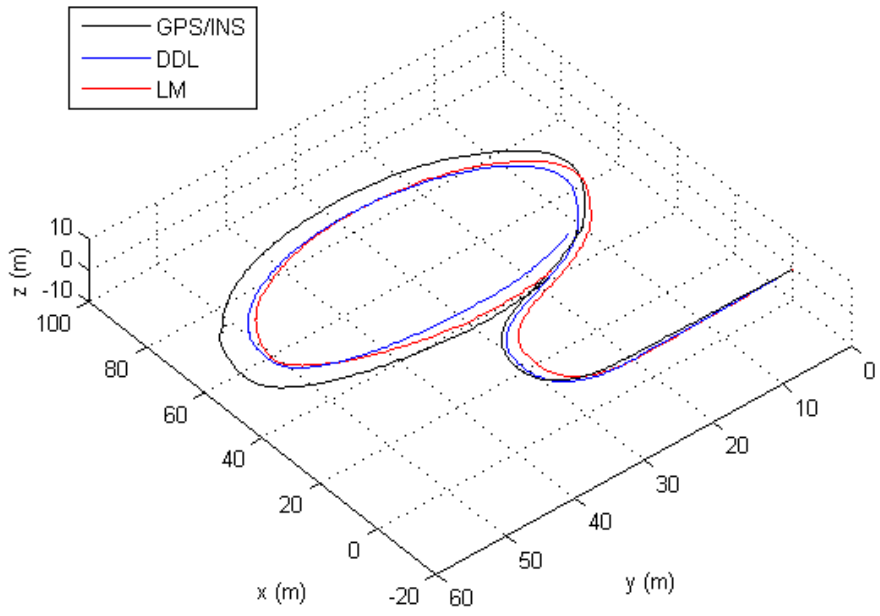


Figure 4.28. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_09_drive_0082.*



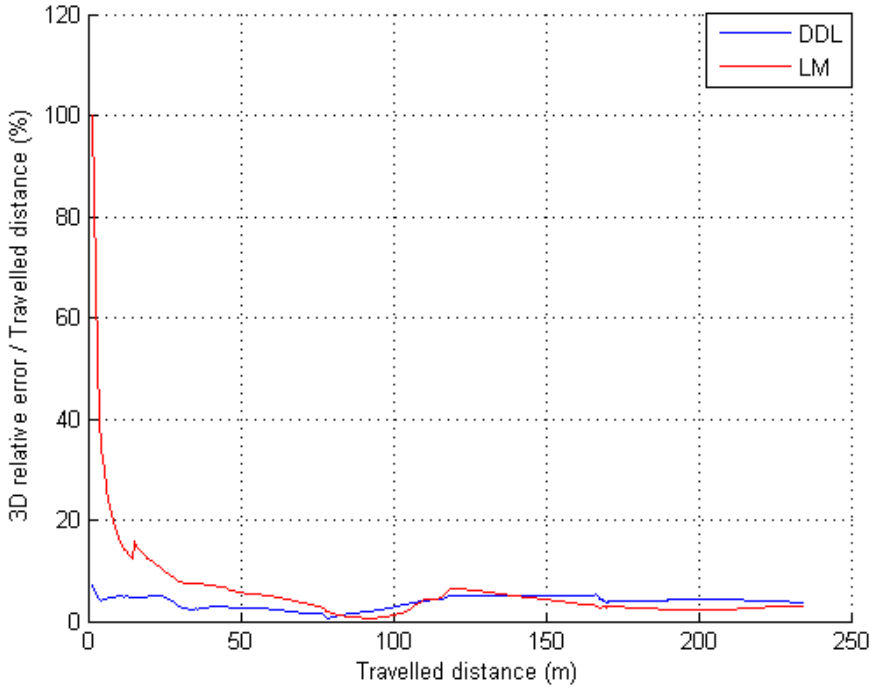


Figure 4.29. Three *dimensional plot of LBA based generated trajectories using double Dogleg and Levenberg-Marquardt (top); respective 3D relative error over the travelled distance (bottom) on dataset 2010_03_17_drive_0046.*

Table 4.1. *Summary of the Final Relative Position Error over Travelled Distance for the Different Datasets.*

		Double Dogleg (DDL)				Levenberg-Marquardt (LM)			
		2D relative error		3D relative error		2D relative error		3D relative error	
Datasets	Travelled distance	in m	in %	in m	in %	in m	in %	in m	in %
2009_08_09_drive_0010	431.06 m	6.23	1.44	6.23	**1.44**	7.05	1.63	7.19	1.67
2009_08_09_drive_0021	271.20 m	2.35	0.87	2.36	**0.87**	3.36	1.24	3.36	1.24
2009_12_14_drive_0051	462.28m	2.78	0.60	10.74	**2.32**	19.56	4.23	20.23	4.37
2010_03_04_drive_0033	213.65 m	3.24	1.51	3.25	**1.52**	75	35.11	75	35.11
2010_03_09_drive_0019	225.22 m	5.59	2.48	5.63	**2.5**	11.88	5.27	11.89	5.28
2010_03_09_drive_0020	309.74 m	7.12	2.29	7.33	**2.36**	13.76	4.44	13.85	4.47
2010_03_09_drive_0023	92.97 m	2.49	2.68	2.49	2.68	1.12	1.21	1.12	**1.21**
2010_03_09_drive_0051	412.30 m	3.52	0.85	4.03	**0.97**	3.89	0.94	4.29	1.04
2010_03_09_drive_0081	184.94 m	1.49	0.80	1.53	**0.83**	3.07	1.65	3.07	1.66
2010_03_09_drive_0082	62.69 m	2.59	4.14	2.75	4.40	0.50	0.80	1.12	**1.80**
2010_03_17_drive_0046	234.46 m	1.98	0.84	8.42	3.59	4.75	2.02	6.98	**2.97**

Figures 4.19 to 4.29 illustrate for the presented datasets the 3D relative error over the travelled distance and the related 3D trajectories generated with local bundle adjustment using either double Dogleg or Levenberg-Marquardt minimisation technique. Table 4.1 gives a summary error over the travelled distances. The general trend over these datasets shows a better quality of generated trajectory using Local bundle adjustment with double dogleg minimisation technique except for three datasets (2010_03_09_drive_0023, 2010_03_09_drive_0082, and 2010_03_17_drive_0046). In Figures 4.21, 4.23, and especially 4.22, the trajectories generated using Levenberg-Marquardt algorithm have much more difficulty to remain close to the GPS/INS reference. However, apart from these cases with both minimisation techniques used in the two views local bundle adjustment scheme, results achieved in term of error are between 1% and 4%, which is remarkable. This is even more true when using double Dogleg. Notably, state of the arts techniques in term of error achieved for long term trajectories with less challenges as proposed here are in the close range (lower than 1%) [12], [13]. The base of the presented

motion estimation algorithm following a two views local bundle adjustment scheme associated with double Dogleg methods is certainly more than promising.

4.5 Chapter Conclusion

In this chapter linear and nonlinear motion estimation techniques were presented as a completion of the preceding visual odometry stages shown in Chapter 2 and including feature detection, stereo matching and feature tracking. Firstly, a general quaternion based motion estimation method using 3D structure correspondences only was studied. Associated with the geometric property based outlier rejection scheme, this approach provided reasonably good trajectory estimation on long-range outdoors urban dataset. Nevertheless, this was not accurate enough especially in height estimation to fulfil all the requirements of this project.

A second general motion estimation method aiming to reduce the 3D feature position approximations by minimising the re-projected 3D points into the image plane was also investigated. This method was implemented using a two views local bundle adjustment (LBA) scheme with an innovative double Dogleg minimisation technique. Additionally, an inliers selection RANSAC based process is fully integrated to the motion estimation algorithm. Results on different long outdoor urban scenarios showed in a first instance that the implemented LBA algorithm considerably enhances the accuracy of the generated trajectory compared to a 3D structure only based motion (quaternion based) estimation algorithm. Second, the introduced double Dogleg nonlinear minimisation algorithm demonstrated higher stability and a lower relative error in trajectory estimation in the majority of the scenarios routes than the widely used Levenberg-Marquardt algorithm. The reached precision with LBA and its ability to cope with dynamic environment makes this approach very suitable to fit for a navigation system visual odometry algorithm.

5 IMU-Visual Assisted Feature Tracking for Motion Estimation

5.1 Overview

Motion estimation is the central part of visual odometry algorithms. This process computes ego-motion of a camera equipped platform relatively to a pair of subsequent frames. To do so, motion estimation algorithms use correspondences between consecutive images. The successive accumulation of these relative inter-frame motions provides the full trajectory that a camera equipped platform has followed. In Chapter 2, emphasis was put on feature detection and feature tracking. The reasons behind this are related to the critical role and influence that feature correspondences have on the motion estimation process.

Accurate estimation of the camera motion during navigation depends on the ability to successfully track features over successive images. KLT (Kanade-Lucas-Tomasi) is one of the most popular feature tracking techniques (see [21], [77], [78]). KLT considers local information derived from small search windows surrounding each of the interest points. This local process constrains template image analysis in time, space, and brightness. This is acceptable for image sequences with small appearance change i.e. high frame rate, coherent motion, stable illumination, etc... . As a consequence, it is relatively efficient on

image sequences with small changes in appearance between subsequent frames. However, it becomes very difficult or impossible for KLT tracker to deal with large inter-frame motions inducing a substantial optical flow. To deal with this problem, a pyramidal implementation of the KLT algorithm was developed [24]. The latter runs the KLT algorithm iteratively on a local template window through successive multi-resolution layers starting from the top of the pyramid (lowest resolution) until recovering the initial image size. This allows larger motions to be caught with less difficulty by breaking the distance through the multi-scale approach. Although it is reasonably efficient, it is also a time consuming technique not easily usable for real-time applications.

5.2 KLT Variation Methods

In general, all KLT-based variants modify the warping function using an affine model to adapt the template to the different conditions that might occur between two successive images, such as change in illumination, rotation and scale [79]–[81]. The work of [82] presents a gyro-aided KLT method. The instantaneous gyro angles are used to get inter-image rotation that serves to compute the homography matrix between two consecutive images. The obtained homography matrix is used to update the parameters of an affine photometric model for the warping function. The affine photometric model has 8 parameters allowing robust tracking to camera rotation and outdoor illumination. However, this model leads to a significant computational cost.

Another work described in [83] uses a gyro-aided feature tracking solution for video stabilization inspired by the human vestibulo-ocular reflex. In this contribution, gyroscope measurements from IMU sensor and intrinsic camera parameters are also used to obtain the homography matrix. In contrast to [82], a translational model is preferred to the affine one for the KLT warping function, for computation complexity reasons. As opposite to the majority of contributions using a mono camera, [84] uses a stereo camera on a UAV. In this work, GPS/INS information serves to find the rotation angle between stereo frames. It proposes two affine models of the warping function including orientation information. Assuming a constant distance between the UAV and the ground (i.e. no scaling changes), the angular information is either used to rotate the entire second image into the same orientation as the first image; or to rotate the tracking windows into the same orientation as the first image. Feature tracking is based on pyramidal KLT.

In these three contributions [82]–[84], the benefit of gyroscope information is significant allowing the KLT to cope with sharp rotation where it usually fails. However, this remains possible only at the condition of a quasi-pure or a pure camera rotation. Hence, it is assumed a negligible inter-frame translation regarding the scene depth. High frame rate enables to fulfil this condition, and can be easily achieved [83], [84] due to computational efficiency of the KLT translational model. On the other hand, [82] requires a parallel processing implementation to achieve high frame rate. Also, by using GPS/INS [84], the system is dependant of an external reference.

Our motivation is to propose an innovative computationally efficient IMU assisted KLT tracker, using the full IMU information that is robust against rotation changes but also important scaling between consecutive images. Consequently, KLT is partially released of its spatial constraint allowing low frame rate processing, which is not the case for gyro-only solutions. To enable a continuous and efficient use of accelerometer measurements, the IMU information has to be updated over time. This is why our IMU assisted KLT tracker technique is an integral part of a visual odometry algorithm, which is the second contribution of this work. Indeed, at each new image the inter-frame pose resulting from our visual odometry initialises the IMU. The overall solution is independent of any external source (e.g. GPS [84]).

5.3 IMU Integration in the Visual Odometry Framework

5.3.1 Visual Odometry and IMU Association

Limitations of classical visual odometry have been identified at early stages [16]. Hence, many fusion works with various sensors have been investigated in order to find the most complementary combination. IMU is particularly cheap and easy to implement with vision systems. Its high frequency provides precious motion information filling the interval gap of lower frequency associated to vision sensors (see [85]–[87]). It is well known that inertial measurements drift with time if there is no update from an external source [88]. Associating it with sensors such as GPS or a camera, allows the correction of IMU absolute position, and consequently prevents it from drifting over time. In [89], IMU information is used to provide the gravity vector, resulting in a reduction of motion parameters. Those are then combined with Hough Transform to estimate the ego-motion

without feature correspondence. The use of an Extended Kalman Filter (EKF) or one of it variants has been favoured and extensively chosen to fuse inertial and vision data, essentially to resolve pose estimation problems (e.g. [90]–[92]). Despite showing relatively good efficiency, Kalman Filter approaches based on the pinhole camera model involve costly update of the covariance matrix. This is true especially for the ones including features in the state vector.

The novel visual odometry approach proposed here is not a proper fusion, but more a complementary association with the IMU. Visual odometry benefits from a better quality of features with the IMU assisted-KLT whilst the IMU is initialised from visual motion estimation. Our approach aims to handle large motion and computational burden by exploiting high frequency measurements from the IMU.

5.3.2 Visual Odometry Framework Overview

Figure 5.1 illustrates the whole framework of our visual odometry algorithm, which includes IMU-assisted KLT feature tracking. The different stages composing the algorithm are divided into three categories: Vision only, IMU only, Combined Vision and IMU operations. The presented algorithm forms a closed loop designed to maintain the balance between visual and inertial information

5.3.2.1 IMU Initialisation

The IMU initialisation step consists in setting the initial value to the rotation matrix R_{imu} and the translation vector t_{imu}. R_{imu} is simply initialised with an identity matrix as we just need instantaneous gyro-angle. On the other hand, t_{imu} is initialised from the latest resulting visual inter-image pose translation vector t_V and its derivate $v_V = t_V/\Delta_v$ (Δ_v the time elapsed between two stereo frames). Assuming a uniform acceleration during a short interval of time, acquired accelerations a and angular velocities ω are then respectively double and simply integrated.

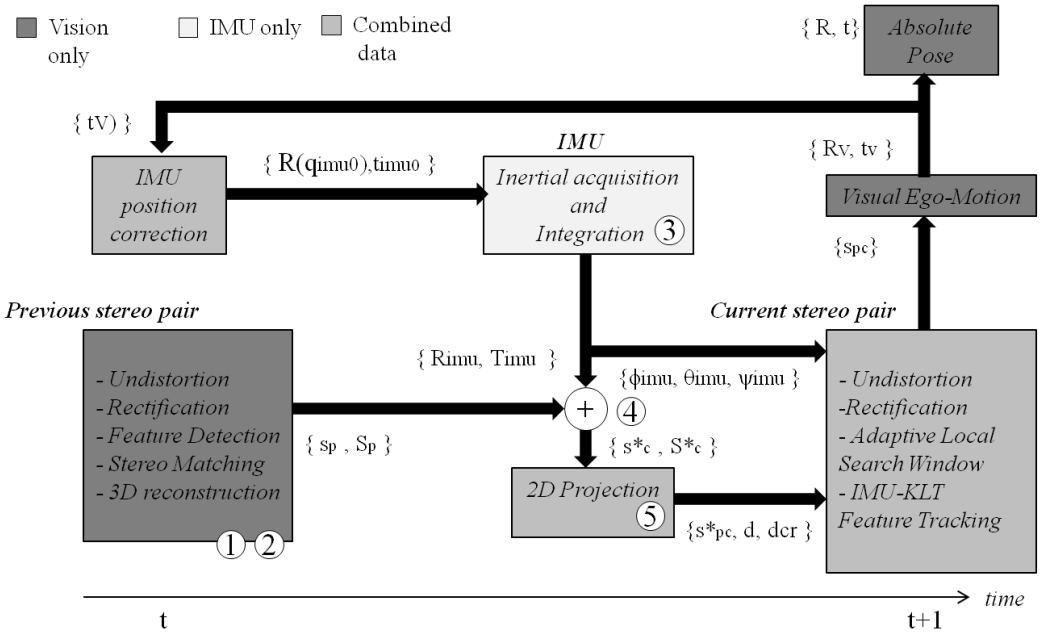


Figure 5.1. *IMU-assisted feature tracking principle illustration within the visual odometry framework.*

Resulting positions and angles are steeply accumulated at each new reading until a new image is acquired. At the end, we obtain the full inertial motion between previous and current stereo-pair represented by $R(q_{imu})$ and t_{imu}.

Note, that in our case we make use of an Xsens® Mti-g sensor, which gives the possibility to get calibrated accelerometer and gyroscope data. Gravity compensation is also applied to the Xsens® accelerometer data before being integrated to get the free acceleration.

5.3.2.2 Feature detection, stereo matching and 3D reconstruction

At each new stereo image pair, while the IMU data are initialised, feature detection stage is processed for each pair. Detected features are tracked from previous left image I_{pL} to previous right I_{pR}. To do so, we use a descriptor based stereo matching scheme. Matches that do not validate epipolar and disparity constraints are rejected. Correct stereo matches presenting a disparity that is too small are rejected as well. The remaining good correspondences form a set of 2D points that are projected into 3D using stereo calibration parameters by triangulation. This results into a set of 3D points representing detected and matched features in the stereo pair.

5.3.2.3 Visual-IMU Combination and Feature Tracking

The images where feature detection and stereo tracking was achieved from the previous stereo pair and are called I_{pL} and I_{pR} while the newly acquired stereo image are denoted I_{cL} and I_{cR}. At this point, the inertial inter-frame motion ($R(q_{imu})$ and t_{imu}) between previous and current stereo image pairs is combined with the formed set of 3D points. This results into a new set of post inertial motion 3D points that are projected into the current stereo pair of images I_{cL} and I_{cR}. This gives a set of 2D initial inertial guess. An adaptive local tracking window representing inertial guess neighbourhood is calculated for each point. Then, sub-images with the same size as the tracking windows are extracted at the location of each inertial guess and its related initial point. KLT is run between these sub-images in previous and current stereo images pairs resulting in a set of tracked features. Tracked features have to validate the same epipolar and disparity conditions that have been set earlier in the stereo matching stage.

5.3.2.4 Visual Motion Estimation

Visual motion estimation is computed from the correct set of tracked features by minimizing the sum of their re-projection errors into the camera coordinate system as it has been described in Chapter 4. We adopt a frame to frame approach where we are looking for motion parameters forming the inter-image rotation matrix R and the translation vector t that minimize the re-projection error. The solution is obtained using the previously presented double dogleg trust region method (Section 4.4.2.2). As a reminder, following a RANSAC scheme, we take as the initial solution the one that gathers the highest number of correspondences (inliers) whose re-projection errors go below a set threshold. Considering only the inliers resulting from the initial solution, optimization is run for the last time for the refinement of the motion parameters. Finally, we obtain the inter-image velocity from the motion parameter, which is input into the measurement vector of a Kalman filter. In an eventual case of failure in the motion estimation process, observed velocity is taken from the inertial motion parameters that served earlier in the IMU-KLT feature tracking stage. The absolute pose cumulates the inter-image rotation matrix R_v and the translation vector t_v with the previous ones. The loop is closed by using t_v for the initialisation step of the IMU.

5.4 Framework IMU Assisted Feature Tracking

Techniques using orientation information from an external sensor such as IMU's gyroscope [82], [83] or GPS/INS [84] have been developed in order to cope with fast camera rotations or severe shakes, which usually break KLT conditions. In this work, we develop a technique that also copes with sharp camera rotations but especially extends the use of KLT to severe scale changes.

5.4.1 IMU Feature Projection via Stereo 3D Reconstruction

The singularity of our technique resides in the use of stereoscopic properties in order to combine visual and inertial data. Indeed, in contrary to similar works that are based on 2D transform image operation [82], [83], we use 3D geometry to project initial features with the knowledge of inertial inter-image transform ($R(q_{imu})$ and t_{imu}) into current stereo images.

Figure 5.2 summarizes our idea and highlights five keys steps: Stereo matching, 3D reconstruction, IMU motion transformation, calculation of the post inertial motion 3D feature and finally, its projection into image plane of the current stereo pair.

In the first step, a set of *n* stereo correspondences $s_p=\{ p_{pL(j)}, p_{pR(j)} \}$ ($j = 1, \dots, m$, *m the number of points*) are obtained from the stereo matching stage. In the second step, s_p is projected into 3D using stereo calibration parameters by triangulation. A set of 3D points $S_p=\{ P_{(j)} \}$ is formed representing the position of the stereo correspondences in the space. Then, in step 4, the inertial inter-image transform composed of $R(q_{imu})$ and t_{imu} and resulting from integration in step 3 of IMU readings (accelerometer and gyroscope) is combined with S_p following the equation of motion (4.1) described below:

$$P^*_{cimu} = R(q_{imu})P_p + t_{imu} \tag{5.1}$$

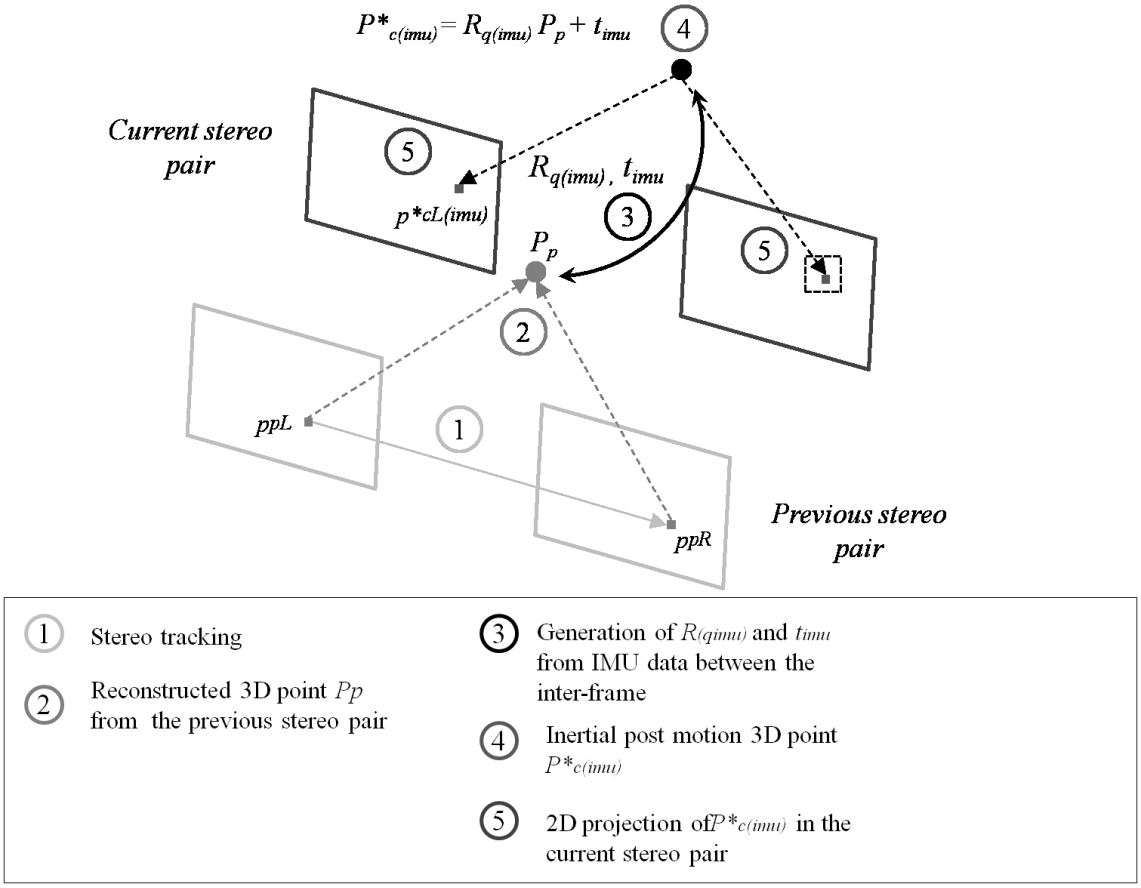


Figure 5.2. *Illustration of the main steps of IMU-assisted feature tracking principle for one point including: stereo matching, 3D reconstruction, IMU motion transformation, and projection into image plane.*

This gives in a set of post motion 3D locations $S^*_c = \{ P^*_{c(imu)} \}$ with $P_p = [X_p, Y_p, Z_p]^T$ and $P^*_{c(imu)} = [X_{c(imu)}, Y_{c(imu)}, Z_{ci(imu)}]^T$. Finally, in step 5, components of s^*_c are projected into the current stereo images pair camera coordinate using the stereo camera parameters following the same methodology of (4.19):

$$\begin{cases} p^*_{cL(imu)} = \begin{bmatrix} u^*_{cL(imu)} \\ v^*_{cL(imu)} \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \end{bmatrix} \begin{bmatrix} X_{c(imu)}/Z_{c(imu)} \\ Y_{c(imu)}/Z_{c(imu)} \\ 1 \end{bmatrix} \\ p^*_{cR(imu)} = \begin{bmatrix} u^*_{cR(imu)} \\ v^*_{cR(imu)} \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \end{bmatrix} \begin{bmatrix} X_{c(imu)} - B)/Z_{c(imu)} \\ Y_{c(imu)}/Z_{c(imu)} \\ 1 \end{bmatrix} \end{cases} \quad (5.2)$$

where f is the focal length, u_0 and v_0 are the coordinates of the central pixel, and B is the stereo baseline. This gives a set of inertial predictive pixel locations $s^*_c = \{p^*_{cL(imu)}, p^*_{cR(imu)}\}$ from which a high confidence tracking area can be built.

5.4.2 Adaptive Local Tracking Windows

Most of the time, projected inertial features give a relatively fair initial guess for the KLT. That said, and in order to benefit most from this, we aim to maximise the use of the KLT by restraining the tracking area to the neighbourhood of the inertial guesses. Indeed, restraining the tracking area has two major advantages. First, it reduces the probability of tracking wrong features, and the second is that it increases KLT computational efficiency. We call these areas adaptive local tracking windows (ALTW) as it is illustrated in Figure 5.3.

Let us call individual tracking set, the set regrouping the initial stereo points and their related inertial projections $s^*_{pc} = \{p_{pL(j)}, p_{pR(j)}; p^*_{cL(imu)}, p^*_{cR(imu)}\}$. Sizes of the adaptive local tracking windows are calculated according to global and local parameters forming the vector ζ:

$$\zeta = [\,|\varphi_{imu}|\ \ |\theta_{imu}|\ |\psi_{imu}|\ d\ \ dcr]^T \tag{5.3}$$

Global parameters $[|\varphi|\ |\theta|\ |\psi|]^T$ are the Euler angles deduced from the inertial inter-image information from $R(q_{imu})$. They are common to all set of features s_p/s^*_c. Local parameters $[d, dcr]^T$ represent respectively the disparity and the distance of the feature from the image centre. These are specific to each components of an individual tracking set s^*_{pc}. According to the parameters ζ, adaptive local tracking window size $A(\zeta)$ is calculated for each component of s^*_{pc} following the conditions below:

If the velocity *vehicle* > *3m/s* and ($|\varphi_{imu}|$ or $|\theta_{imu}|$ or $|\psi_{imu}|$) > 0.009 *rads* then:

$$A(\zeta) = ws + f(|\varphi|, |\theta|, |\phi|) + g(d, dcr) \tag{5.4}$$

otherwise

$$A(\zeta) = ws + g(d, dcr) \tag{5.5}$$

where *ws* is a constant 9x9 base size of the local tracking window which can be widen gradually according to the score obtained with the sub-functions $g(d, dcr)$ and $f(|\varphi|, |\theta|,$

$|\psi|$). For instance, typical values { 0.5°,1°, 2°, 3°, 4°} correspond to {6.9, 9.5, 16.5, 26.9, 41.5 pixels}. The sub-functions *f* and *g* are experimentally expressed as follow:

$$f(|\varphi|,|\theta|,|\phi|) = \frac{(1+10*\max(|\varphi|,|\theta|,|\phi|))^4}{0.2} \tag{5.6}$$

$$g(d,dcr) = \frac{4}{f*B} * d + 2\left(\frac{dcr-400}{100}\right) \tag{5.7}$$

With *f* the focal length, and *B* the stereo baseline. Notably, an upper limit of the adaptive local tracking window is fixed to 40 square pixels even if the score of $A(\zeta)$ is found to be higher.

This might lead to a maximum of four different ALTW sizes for each tracking set Q. However, we keep the adaptive local tracking windows with the larger size. Thus, each individual tracking set s^*_{pc} has a unique ALTW size $A(\zeta)$ surrounding each of its components. For each component of s^*_{pc}, we extract a region of interest, which has the size of the calculated ALTW resulting in a sub-set of 4 images $I_{ALTW} = \{I_{pLA(\zeta)}, I_{pRA(\zeta)}; I_{cLA(\zeta)}, I_{cRA(\zeta)}\}$. Before running KLT, a phase correlation stage is done between $I_{pLA(\zeta)}$ and $I_{pRA(\zeta)}$ and respectively between $I_{cLA(\zeta)}$ and $I_{cRA(\zeta)}$.

Phase correlation stage is a fast frequency based approach giving an estimate of an eventual translational offset between two similar images. If an offset is found, we use it to refine the position of inertial projected feature. In Figure 5.3, the concept of ALTW is illustrated in 4 main steps. Its application at different parts of a dataset is shown in Figure 5.4.

Discrete Fourier Transform (DFT) is calculated for the two related ALTW and serves then to compute the cross-power spectrum R as such:

$$R = \frac{G_a G_b^-}{|G_a G_b^-|} \tag{5.8}$$

with

$$\begin{cases} G_a = F\{I_{pLA(\zeta)}\} \text{ and } G_b = F\{I_{cLA(\zeta)}\} \text{ for the left pair} \\ \\ G_a = F\{I_{pRA(\zeta)}\} \text{ and } G_b = F\{I_{cRA(\zeta)}\} \text{ for the right pair} \end{cases} \tag{5.9}$$

where F is the forward DFT and the exponent " $^{-}$ " indicates the conjugate of the DFT . Then, the cross correlation (5.8) is converted back to the time domain and the translational shift σ is deduced from the peak location:

$$(\sigma_x, \sigma_y) = \underset{(x,y)}{\arg max} \quad (F^{-1}\{R\}) \tag{5.10}$$

where F^{-1} is the inverse DFT.

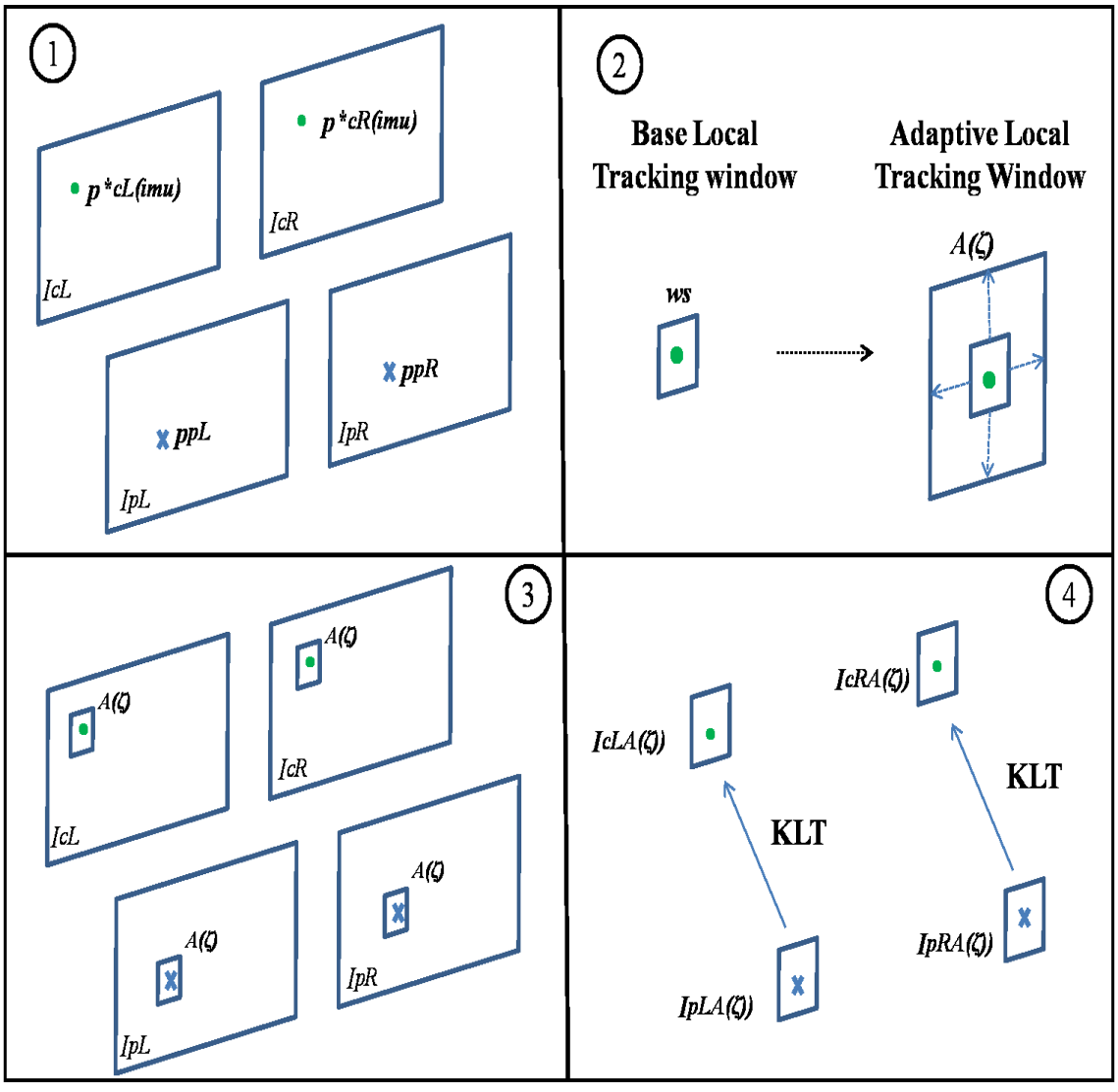


Figure 5.3. *Illustration of the ALTW concept in four main steps: inertial projection, size calculation of adaptive local tracking windows, sub-set image extraction and feature tracking with KLT.*

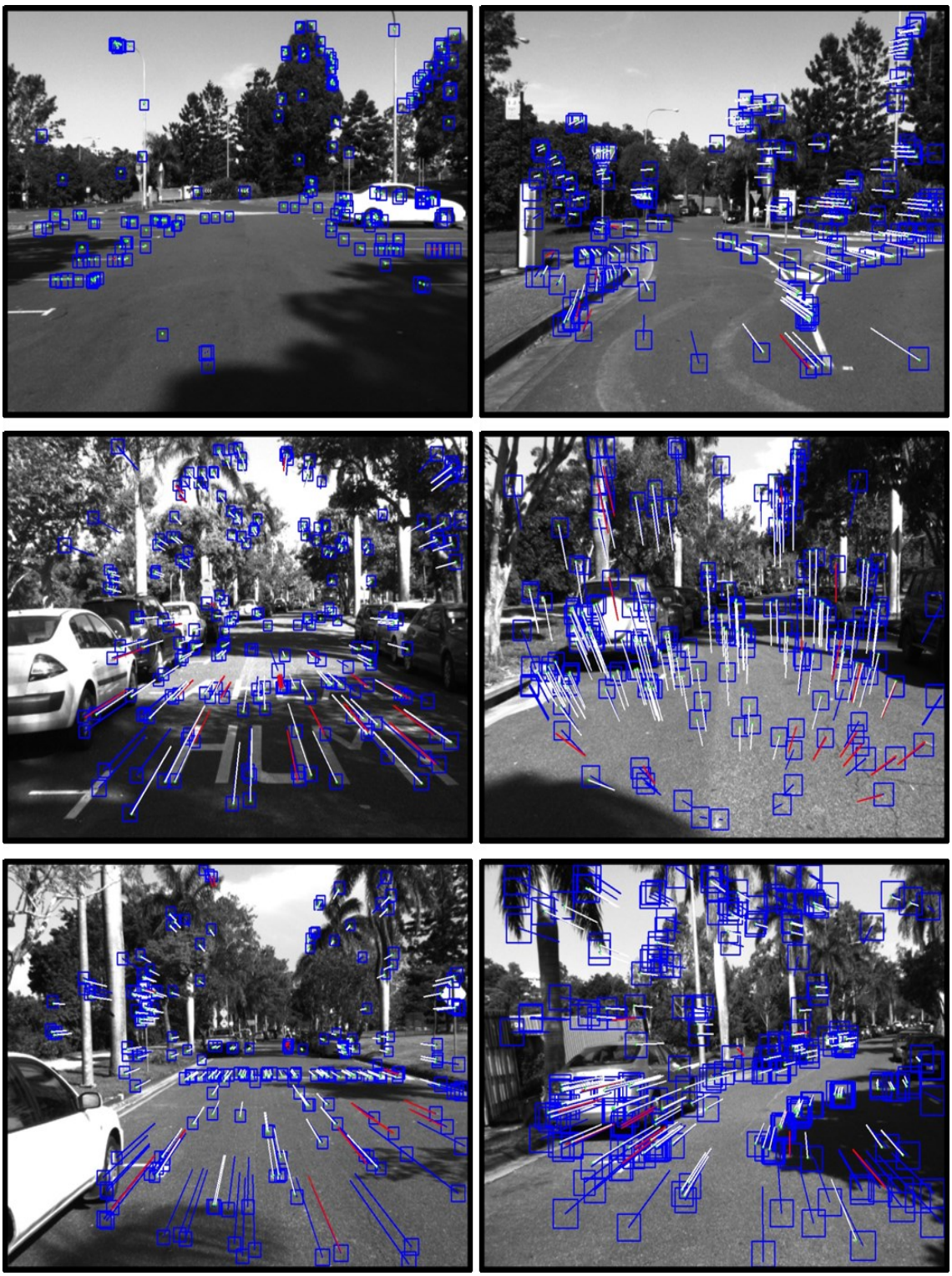


Figure 5.4. *Illustration of adaptive local tracking windows (blue squares) at different points of the dataset. blue lines - inertial optical flow, red lines - outliers, white lines - inliers.*

5.4.3 KLT Formulation

For this implementation, the chosen KLT model chosen is translational (2.15). This model takes as initial starting points of the warping parameter p, the inertial projected features as in (5.2):

$$\mathrm{p} = \mathrm{b} = \begin{cases} \begin{bmatrix} u^*_{cL(imu)} \\ v^*_{cL(imu)} \end{bmatrix} for\ the\ left\ pair \\ \begin{bmatrix} u^*_{cR(imu)} \\ v^*_{cR(imu)} \end{bmatrix} for\ the\ right\ pair \end{cases} \tag{5.11}$$

Thus, the inertial projection features, which have a great probability to get close enough to their related targets are provided as initial condition to minimize (2.14). The latter is re-written with the knowledge of the ALTW ((5.4), (5.5), and Figure 5.3) as:

$$e = \begin{cases} \sum_{-w_x}^{w_x} \sum_{-w_y}^{w_y} \left[I_{pLA(\zeta)}(\mathrm{x}) - I_{cLA(\zeta)}(\mathrm{w}(\mathrm{x};\mathrm{p})) \right]^2 \\ \sum_{-w_x}^{w_x} \sum_{-w_y}^{w_y} \left[I_{pRA(\zeta)}(\mathrm{x}) - I_{cRA(\zeta)}(\mathrm{w}(\mathrm{x};\mathrm{p})) \right]^2 \end{cases} \tag{5.12}$$

Indeed, the KLT is no more run over the entire images but on sub-images with the sizes and the locations of the calculated ALTW$A(\zeta)$. This presents many advantages. Firstly, the inertial projection features contain the full inter-image transformation (rotation plus translation), which enables the KLT to remain robust to scale changes. Secondly, the sub-images only extract the close neighbourhood of the points composing a set S^*_{pc}. Consequently it prevents the KLT from false tracking. Thirdly, the computational time of the KLT is considerably reduced because of the sub-images reduced sizes.

All these advantages are made possible by the combination of full inertial information with 3D geometry and stereoscopy allowing a precise prediction of the tracked features location. This is the main difference with the work of [82], [83] where inertial features are calculated from the rotation matrix R_{gyro} only using 2D image homography transformation as follows:

$$\begin{cases} \mathrm{b} = \mathrm{x}_{gyro} = H\mathrm{x} \\ with \\ H = KR_{gyro}K^{-1} \end{cases} \tag{5.13}$$

where H is the 3x3 homography matrix and K is the camera calibration matrix. Features are rotated according to the gyroscope information in the image plane and serve as initial conditions to minimize (2.14). The warping function of [83] follows the translational model and consequently it only requires b (2.15) as input for the warping parameter p. On the other hand, a more advanced affine photometric model is used with 8 parameters $p = (\eta_{xx}, \eta_{xy}, \eta_{yx}, \eta_{yy}, \eta_x, \eta_y, \alpha, \beta)$ where α and β deal with illumination change. For these two techniques [82], [83] based only on rotation information scale change remains an issue.

5.4.4. Inertial Dynamic Feature Rejection

The use of full IMU information combined with 3D geometry and stereoscopy brings to each feature a coherent motion behaviour that gives to all of these features the same projection trend. Indeed, the rigid vehicle-camera-IMU structure jointly moves towards the scene in a unique manner during each inter-frame. Hence, when one or more dynamic objects appear in the scene, their motion usually differ from the platform trajectory.

Thus, features detected on dynamic objects are projected with inertial knowledge as the rest of the static features of the scene. ALTW limits tracking to the neighbourhood of the inertial projection locations which has a good probability to not contain anymore the dynamic feature. Consequently, KLT fails to find a similar pattern which automatically eliminates a dynamic feature. Figure 5.5 illustrates a case where dynamic features belonging to a moving track are discarded.

In Figure 5.5 (bottom left), features characterizing the moving truck are tracked with Affine KLT (red optical flow). On the other hand, in Figure 5.5 (bottom middle) blue lines representing our inertial optical flow are short and inclined. This leads one to believe that the truck has only slightly moved during the inter-frame. Indeed, inertial projections appear far from the actual location of the dynamic objects. Consequently, in Figure 5.5 (bottom right) when we apply the ALTW (blue rectangles), which bounds the tracking to the strict neighbourhood of the inertial projected KLT features, it has no chance to track these dynamic features as it is done when using the Affine KLT Figure 5.5 (bottom left).

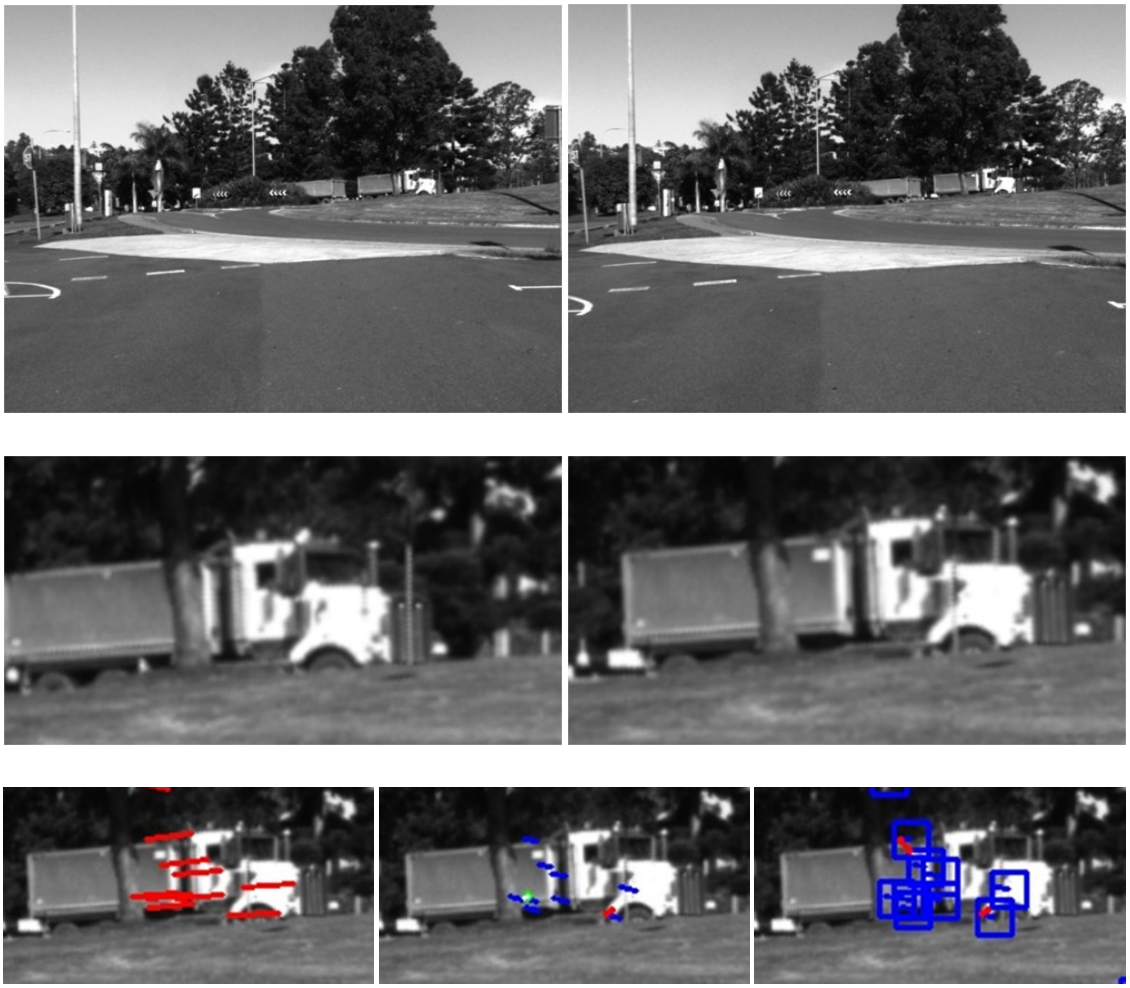


Figure 5.5. *Illustration of inertial dynamic feature rejection process: Top: Consecutive images; Middle: Zoom on the truck; Bottom left- tracking using Affine KLT; Bottom middle- tracking using our method; Bottom right- tracking using our method with ALTW displayed..*

5.5 Experiments and Results

In this section, we present the results of our visual odometry algorithm running on an urban environment dataset [108]. In this dataset, a car is equipped with stereo cameras (Point Grey® Flea2) and an IMU-GPS device (XSens® Mti-g) mounted on its roof. Visual data consist of 1024 x 768 stereo images acquired at 30Hz. IMU data provide calibrated accelerometer and gyroscope information at 100 Hz, and GPS data at 1 Hz. INS data are also provided in this dataset. It will serve as a reference in the visual odometry trajectory comparison tests. In the first instance, feature tracking performance only will be

evaluated. Then, it is the impact of feature tracking on the quality of the motion estimation that is assessed. Finally, motion estimation scheme performance only is evaluated.

5.5.1 Feature Tracking & Processing Time Performances

In this chapter, one of our aims is to assess the performances of our IMU-assisted KLT tracker in the presence of large optical flows. With this in mind, we decided to emphasize the scaling phenomenon on this image sequence by re-sampling it at 10Hz, 5Hz, and 3Hz sequences instead of its original 30Hz acquisition frame rate.

In order to provide a good understanding of the importance of using full IMU information and adaptive subset of images in our IMU-assisted KLT, three variations of our work are tested. The first variation uses accelerometer and gyroscope information and adaptive local tracking windows. The second variation is the same as the first one but without adaptive local tracking windows. The third variation is the same as the second one but using gyroscope information only. Results of our IMU-assisted KLT feature tracking in its three variation forms are compared to conventional KLT but also to the gyro-aided KLT method developed in [82]. In order to have a complete comparison, we adapted to our stereo motion scenario the code of [82], which is originally designed for monocular camera systems. The evaluation criteria are the rate of inliers over tracked features and KLT processing time. Table 5.1 describes abbreviations and characteristics of the compared KLT based techniques. Table 5.2 presents feature tracking and time related performances at different sequence frequencies for the techniques mentioned in Table 5.1. The general trend of Table 5.2 describes a drop of the percentage of correct tracked features while the processing time increases when the gap between consecutive frames gets bigger (i.e. lower frame rate).

T loses almost 50% of features on average at 10Hz and can only save 20% at 3Hz. These results are not surprising at all according to the nature of KLT, which hardly copes with scaling.

Table 5.1. *List of KLT Techniques with their Characteristics.*

Techniques	Abbreviation	Model	Pyramid Level	Patch size
KLT (visual only)	T	Translational	0	21 x 21
gyro-assisted KLT without Adaptive local tracking windows (gyro)	GT	Translational	0	21 x 21
IMU-assisted KLT without Adaptive local tracking windows (gyro + accelerometer)	IT	Translational	0	21 x 21
IMU-assisted KLT (gyro + accelerometer)	ITA	Translational	0	$A(\zeta)/2$ x $A(\zeta)/2$
gyro-aided KLT (gyro) [82]	GA	Affine	3	21 x 21

Table 5.2. *KLT Techniques: Tracking Performances, Time Processing and Ratio at Different Frequencies for the Full Sequence.*

		ITA	T	GT	IT	GA
Tracking Performances (%)	10Hz	**92.5**	54.5	56.5	88.3	67.5
	5Hz	**85.4**	32.2	34.2	75.3	59.2
	3Hz	**77.3**	18.7	19	62	47.8
Processing Time (ms)	10Hz	**17**	33	36	29	128
	5Hz	**17**	34	38	31	138
	3Hz	**18**	38	40	31	141
Time ratio	10Hz	1	1.8	1.9	1.6	6.9
	5Hz	1	1.9	2.2	1.7	7.9
	3Hz	1	2.2	2.3	1.8	8.1

GT that adds gyroscope information slightly improves the performance (as well as the processing time) of T. When using the full IMU information (IT), the tracking performance of T is greatly improved and computation time gets also reduced because of the higher precision of the inertial features. GA gives better results than GT. This is even more true at low frequency sequence. However, it remains below IT and its homography based affine model has a certain computation cost. ITA offers the best tracking performance with the lowest processing time. This demonstrates the remarkable advantage of the adaptive local tracking window concept, which enables ITA to be twice faster than T and to keep a relatively high rate of correct tracked features. For instance with 77.3% at 3Hz ITA saves 15.3 % more features than IT, which is the best method after it.

Table 5.3 presents results of the tracking performance for the different KLT based techniques in three challenging areas within the full sequence where the vehicle undergoes obstacles resulting in severe rotations (pitch and roll). In these three cases the results achieved reinforce the findings in Table 5.2. Performances of T get poorer even at 10Hz. On the other hand, we note that gyro based solutions (GT and GA) are better improving T than what they do on the average for the full sequence especially at higher frequencies (10 Hz and 5 Hz). GA gives relatively close results to IT, although it is less valid at 3Hz. ITA remains far better than the other techniques, coping impressively well with the huge scaling and rotational changes. Comparing performances between ITA and IT confirms the contribution of adaptive local tracking window in the improvement of the tracking performance.

Figures 5.6, 5.7, and 5.8 illustrate two consecutives images for each KLT based technique at 5Hz. Most of the tracked features from GA belong to far objects. These are also the one that are less affected with scaling. Hence, they are easily mapped by homography. In Figure 5.7 and 5.8, GA is severely affected by the large scaling and cannot track any features. Conversely, ITA and to a less extent IT are able to track features close to the camera presenting large optical flows that can reach more than 100 pixels in translation. These features are really important for motion estimation as they present a significant disparity, which means that they are less subject to errors.

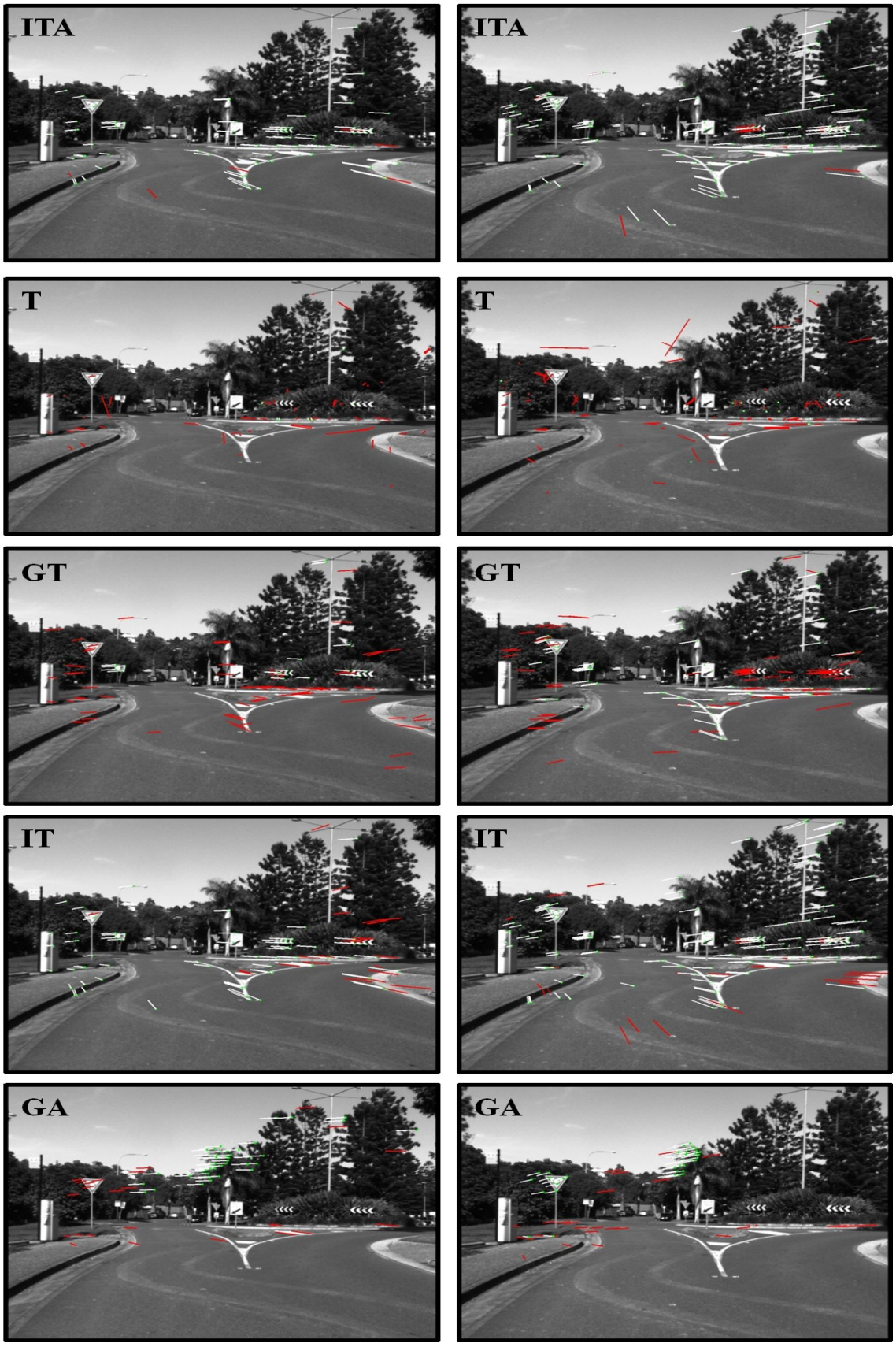


Figure 5.6. *Bump in the Roundabout case: white optical flow-inliers; red optical flow-outliers.*

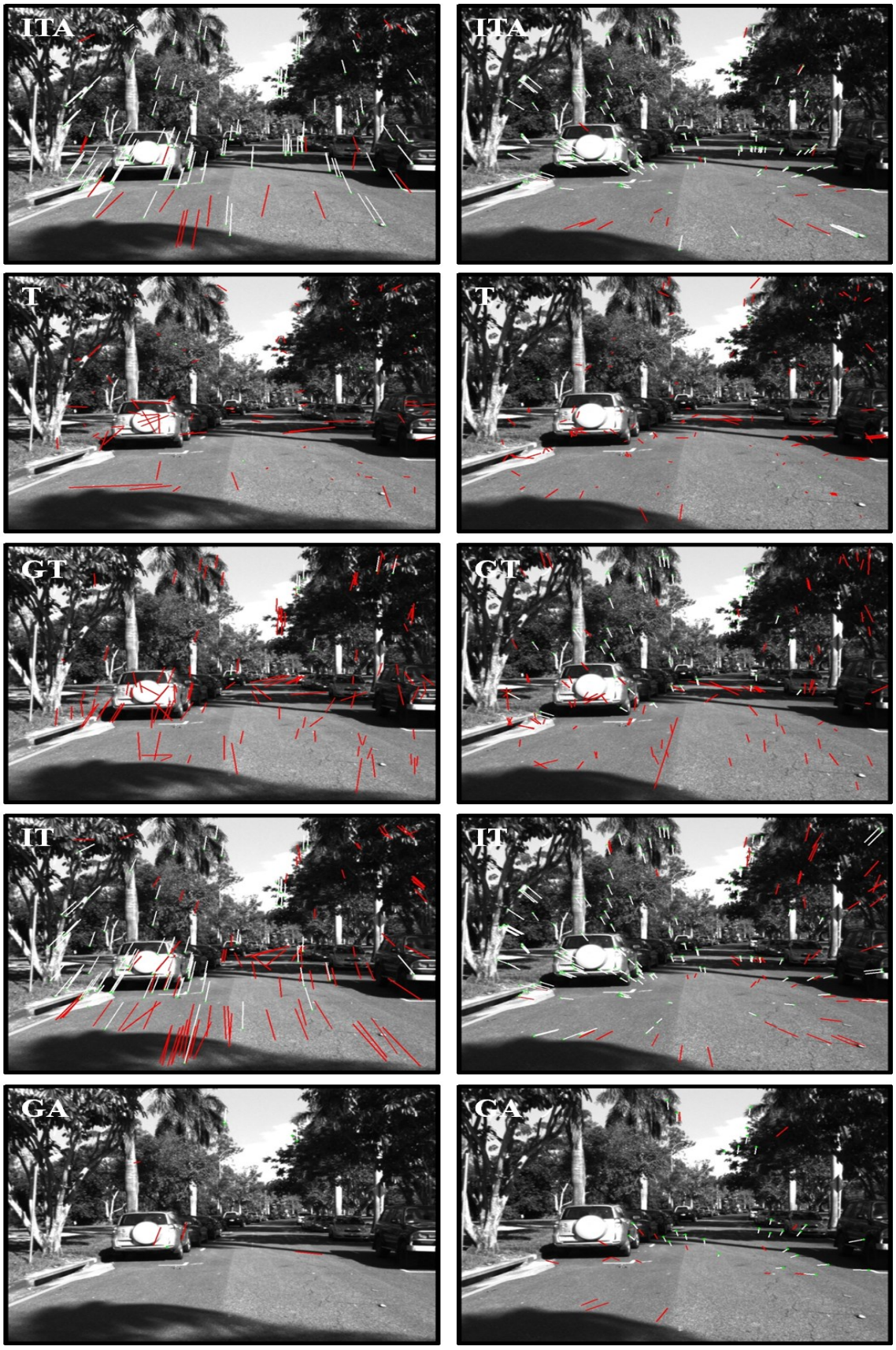


Figure 5.7. *1st Humped crossing case: white optical flow-inliers; red optical flow-outliers.*

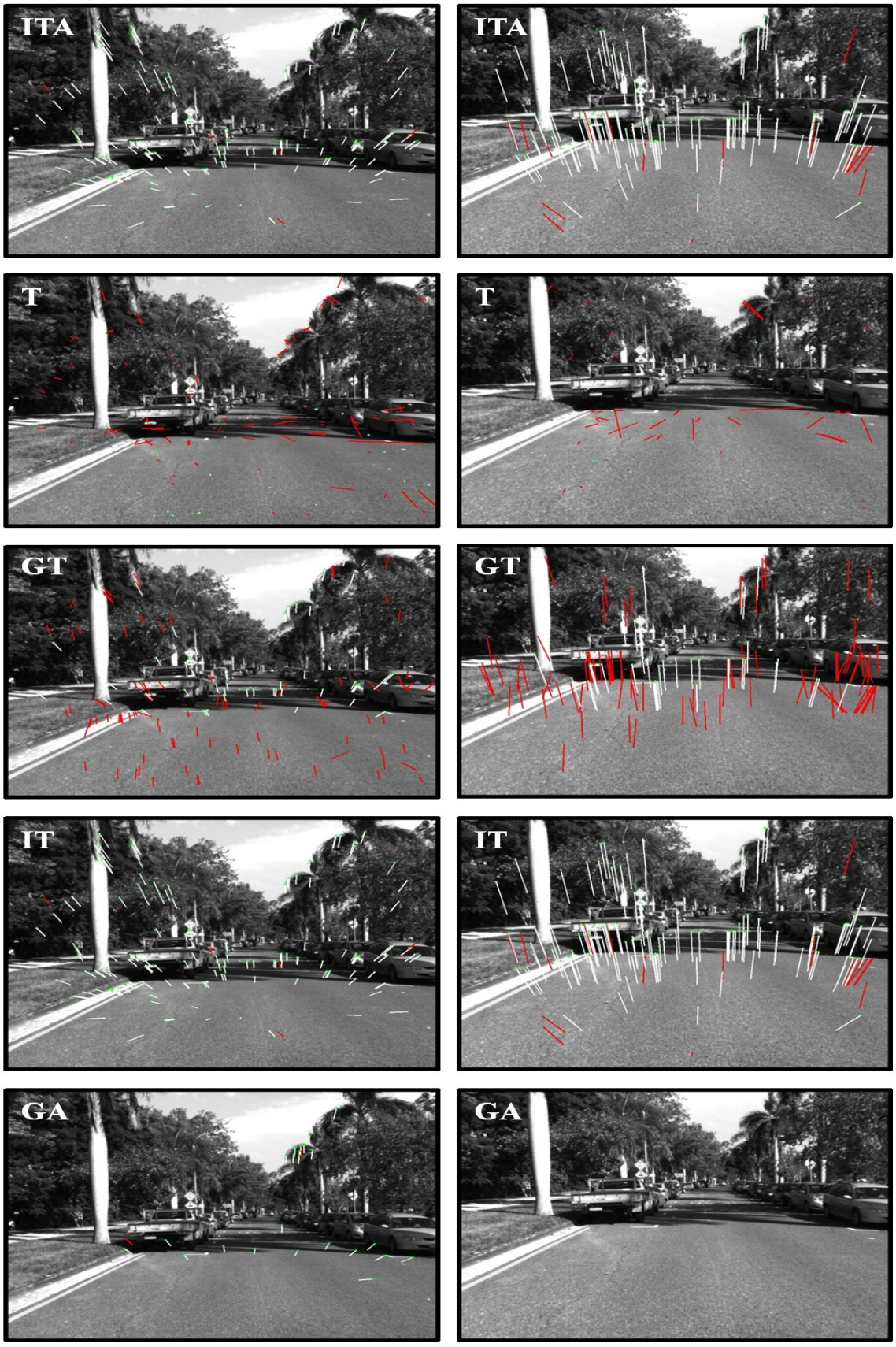


Figure 5.8. *2nd Humped crossing case: white optical flow-inliers; red optical flow-outliers.*

Table 5.3. *KLT Techniques: Tracking Performances at Different Frequencies in Three Challenging Areas.*

Tracking Performances (%)		ITA	T	GT	IT	GA
Bump in a Roundabout	10Hz	**97.2**	16.3	62.1	89.5	78.3
	5Hz	**89.9**	11.6	28	63.5	81
	3Hz	**79.2**	11.6	25.1	49	33.9
1st Humped crossing	10Hz	**95.9**	21.3	57.2	89.6	66.3
	5Hz	**86.9**	11	24.3	63.7	61.2
	3Hz	**88.1**	18.2	17.7	51.1	46.9
2nd Humped crossing	10Hz	**96.9**	13.6	60.4	87.7	87
	5Hz	**88**	14.8	26.5	76.4	63.5
	3Hz	**91.4**	13.2	18.1	69.1	34.9

5.5.2 Impact of Feature Tracking on Motion Estimation

We evaluate the impact of the KLT based techniques on the quality of motion estimation on the same dataset using as optimization method double Dogleg (DDL) for all of them. The vehicle is driven over 491m curved trajectory in an urban environment subject to strong contrasts. Figure 5.9 illustrates the trajectories generated with all KLT based techniques. T, GT and GA, final positions are far from the actual final position F. On the other hand, IT and ITA trajectories follow the road path and their respective final position is very close to F. This confirms the correlation between the number and the quality of the features and the motion estimation accuracy.

T, GT and GA have a lower rate of tracked features compared to ITA and IT. Furthermore, the majority of these features belong to far objects. Thus, motion estimation when it does not fail, gives an under estimated travelled distance. This makes errors accumulate leads to large drifts.

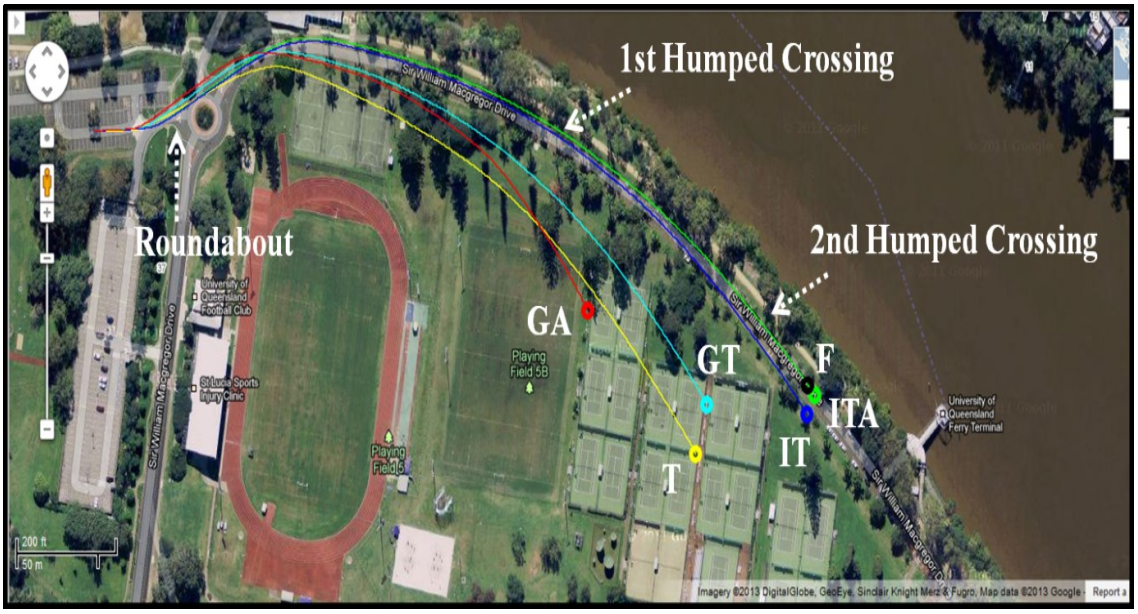


Figure 5.9. *Trajectories generated with KLT Techniques combined with double Dogleg optimization method at 10Hz: ITA (green) ITA; IT (blue); GT (cyan); T (yellow); GA (red); circle final position F (black).*

5.5.3 Motion Estimation

In this sub-section, we evaluate the performance of motion estimation regardless of the feature tracking method. The optimisation methods for motion estimation consist of double Dogleg (DDL), Dogleg (DL) and Levenberg-Marquardt (LM). ITA algorithm will be used for those three methods as the feature tracking technique. Additionally, a filtered INS/GPS trajectory is included in the comparison test. Notably, we found after having aligned INS/GPS trajectory on a satellite image map, it slightly deviates from the road path. Thus, we defined an as accurate as possible final ground truth position F according to the satellite map and the dataset images as illustrated in Figure 5.11. Figure 5.10 shows the trajectories generated with the three techniques at 10 Hz. DDL is the trajectory that remains the closest to the INS/ GPS. DL and LM start slightly drifting from INS/GPS trajectory approximately at half way. However, the drift is not penalising much their final positions.

Table 5.4 gives the 2D squared root error of DDL, DL and LM using ITA as feature tracking approach for all the image sequence frequencies. We additionally show the

results of DDL using IT as feature tracking approach as extra information. The latter is done in order to better appreciate the influence of the adaptive local tracking window concept. At 10 Hz, DDL combined with ITA is closer to F than the INS/GPS with a 2D error of about 1% of the travelled distance. DL, LM and DDL combined with IT get relatively close to F with errors around 2% of the travelled distance.

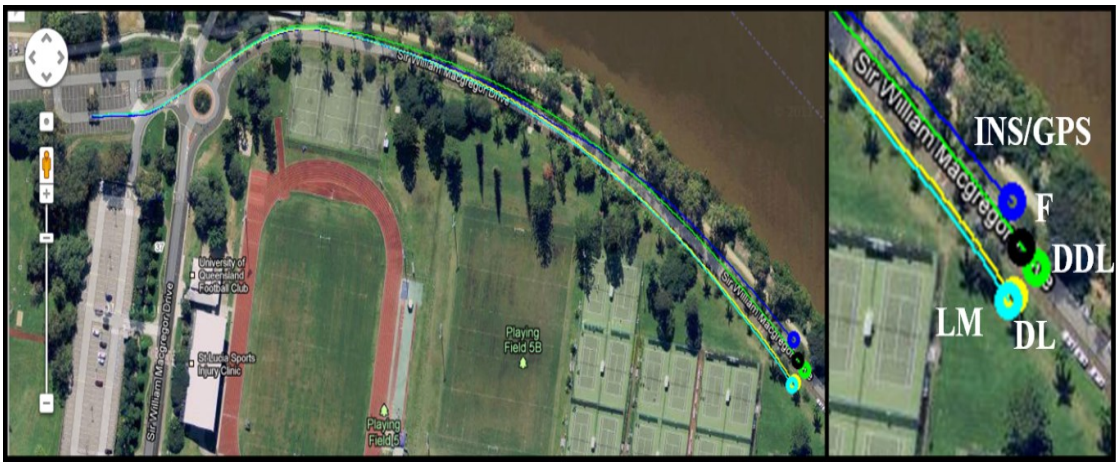


Figure 5.10. *Trajectories generated with ITA combined with different optimization methods at 10Hz: blue INS/ INS/GPS; green DDL ; cyan DL; yellow LM; black circle final position F. Left full map/right zoom on the final position.*

Figure 5.11. *Left-Zoom on the final position (white dot) in front of the "Head Hump" road marking highlighted with the red dashed line; Right-Related image from the dataset of the vehicle final position in front of the "Head Hump" road marking highlighted with the red dashed line.*

When the image sequence frequency gets lower, motion estimation accuracy decreases for all the techniques. However, DDL, and DL using ITA remains under 10% error of the travelled distance. This demonstrates the robustness of our technique to severe scaling in feature tracking but also in the motion estimation. Table 5.5 confirms that DDL is the technique that requires the least iterations to converge to a solution. In Figure 5.10, and Table 5.4, we demonstrated that INS/GPS can be challenged by our method in terms of accuracy in a certain context. In Figure 5.12, we show that our technique gives a much more trustable estimation of the height than that of the INS/ GPS. Indeed, the trajectory generated with ITA combined with DDL remains within an interval of 1 meter from the ground, which is very close to the reality (the real height).

Therefore, we demonstrated that with an efficient and clever fusion between inertial data and visual information we are able to enhance feature tracking of the conventional KLT. This solution is also independent from external sources of information (e.g. satellites) and can thus be employed as an interesting alternative to GPS or INS/GPS. Additionally, by offering the ability to process low frame rate sequences while keeping good performance and low computation meaning that the IMU-assisted KLT can be easily implemented for real time applications.

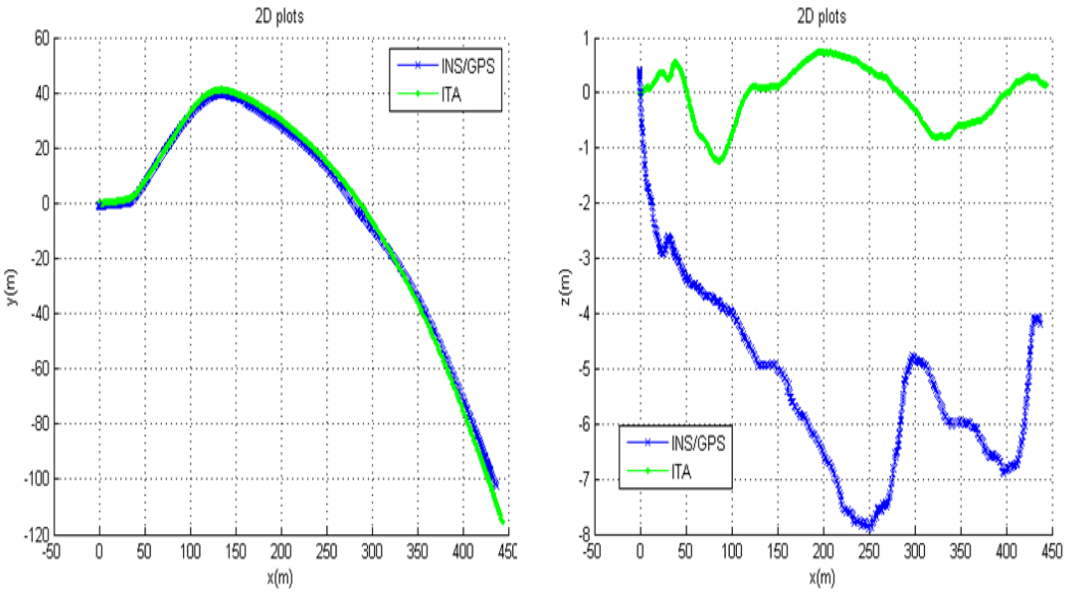


Figure 5.12. *Trajectories comparison between ITA combined with DDL at 10 Hz (green) and INS/ GPS (blue): Left-2D plot x-y; Right-2D plot x-z.*

Table 5.4. *Final Position 2D Relative Error for the Different Optimization Methods at Each Frequency.*

		INS/GPS	Double Dogleg (ITA)	Dogleg (ITA)	Levenberg-Marquardt (ITA)	Double Dogleg (IT)
10Hz	2D RMS (m)	9.78 m	5.03 m	10.33 m	11.74 m	12.48 m
	2D RMS (%)	1.99 %	1.02 %	2.10 %	2.39 %	2.54 %
5Hz	2D RMS (m)	9.78 m	24.93 m	32.01 m	55.35 m	37.66 m
	2D RMS (%)	1.99 %	5.07 %	6.52 %	11.27 %	7.67 %
3Hz	2D RMS (m)	9.78 m	36.96 m	44.01 m	146.5 m	52.22 m
	2D RMS (%)	1.99 %	7.53 %	8.98 %	29.85 %	10.63 %

Table 5.5. *Average Number of Iterations in the Full Sequence for All the Frequencies.*

	Double Dogleg (ITA)	Dogleg (ITA)	Levenberg-Marquardt (ITA)	Double Dogleg (IT)
Iterations	7.1	9.8	12.6	11.6

5.6 Chapter Conclusion

This new feature tracking technique called IMU assisted KLT improves the performance of the conventional KLT while reducing its processing time. IMU linear accelerations and angular velocities are integrated and fused with visual information in such a way that it significantly narrows the matching search region and consequently reduces ambiguity. An adaptive local tracking window enables to precisely locate areas of the inertial predictive features. This permits to the IMU assisted KLT to handle very large motions while keeping a low computational cost. Results obtained by the proposed feature tracker demonstrate higher performance than gyro-aided affine model, gyro-only translational model with a rate of successful tracks which does not go below 80% in the worst case. A motion estimation scheme based on Double Dogleg optimisation technique is introduced and IMU assisted KLT was validated with this scheme and keeps a certain level of performance despite severe scaling changes. The quality of the IMU assisted KLT tracked features enhances the estimated visual based motion and demonstrates higher accuracy than INS/GPS in certain conditions.

6 Smart Visual-IMU Standalone Navigation Sensor

6.1 Overview

In recent years, many research contributions looked at enhancing navigation systems compactness and accuracy. These progresses were made possible with the development of advanced software techniques and algorithms, but also thanks to a wider range of costly affordable off-the-shelve hardware and sensors. In spite of this, building a navigation system remains a challenging task because of the complex trade-off that has to be found between many criteria. Power, autonomy, weight and size represent the most predominant ones. Consequently, the reflexion leading to the right choice of components has to follow a well-defined strategy that needs to take into account compromises and complementarities. The software part has to be included in the early stage of this reflection as it depends heavily on limitations of hardware specifications. Indeed, computational burden can be critical and real-time processing adds a further restriction in the choice of techniques and algorithms. With these constraints in mind, it appears wiser, in term of autonomy and power consumption, to use passive sensors.

Developing reliable navigation sensors that operate in a GPS denied environment is still presenting a big motivating challenge. Although inertial navigation is considerably used

for localisation, it presents drifty characteristics. The latter, even if GPS is available, is not always made up for when long runs are considered. If GPS data is not available (temporarily or definitely), which is common in urban environments and most importantly in space environments (as we often consider in this thesis), precision of localising objects is not possible.

Passive visual sensors such as cameras present many advantages. Cheap, lightweight, smart, and benefiting from a large range of models, cameras have been extensively utilised in research within numerous domains of applications such as navigation, medical imaging, and surveillance. Visual odometry, has proven great achievements in the domain of navigation and localisation [2], [15], [16], [93]. Indeed, less than one per cent relative error achieved in the estimated trajectories has been reported in the literature. This places visual sensors as an almost unavoidable device to equip a navigation system with. Obviously, this concerns majority of applications where there is enough texture, illumination and overlapping static content of the scene between subsequent images.

Although it exists different variations of visual odometry approaches, the majority of them follow a feature based pipeline composed of distinct but interdependent stages summarized here as a reminder: 1) Image acquisition of the first stereo pair 2) detection of remarkable features in these images 3) Image acquisition a new stereo pair, 4) Feature matching and feature tracking between previous and current stereo image pairs, 5) Motion estimation, 6) Local optimisation of the motion parameters. As it can be noticed, it is not a straightforward process. However, a large range of available image processing and optimisation techniques offers relatively good flexibility and freedom to customise this pipeline. This is an advantage that helps building up strategies to fit the best specific conditions of visual odometry application areas. Having said that, real-time processing naturally limits this range of tools. Heavy computational techniques are hardly usable in this context even within a GPU implementation or other hardware acceleration processes. Another point to be considered in such a case is the quality of visual data. Indeed, high-resolution images give better definition of the scene and consequently improve image-processing performance. However, a low frame rate can be problematic for applications where camera motion is subject to fast rotations or if quick changes of the scene content

happen because of high-speed translational motion. Higher camera frame rate helps to reduce these effects.

If it is possible to adapt the resolution at a reasonable frame rate, problems of over or under illumination, and texture-less scenes might affect detection of salient features that feed visual odometry algorithm. Consequently, it will prevent to compute camera motion. To cope with these temporary gaps, it is possible to use assistance of an additional sensor. Inertial measurement unit (IMU), which also has the advantage of being a passive sensor, fits very well with this request. Its high data acquisition frequency enables filling an eventual gap resulting from lack of visual data. IMU data can be fused with visual data for pose estimation through filters such as Kalman and its variations [94]–[97]. Besides, it can also be used to assist the visual tracking process. Thus, a combination of visual odometry and IMU will be advantageous for a navigation solution presenting a good level of accuracy.

Choosing the most suitable sensors to acquire meaningful data is one aspect of the problem when developing a standalone visual odometry based navigation sensor. However, and in order not to retain advantages for which sensors have been selected, associated processing devices need to respond to tight and high expectations. Indeed, cameras and IMU generate an important amount of data. In online applications as we are dealing with in this chapter, a delicate balance between processing capabilities, size and power consumption has to be found. The selected ITX single board computer present interesting characteristics. This low-power motherboard benefits from all the advantages of high performance processing of the standard motherboards while being much smaller (can reach 72x100 mm size). It also includes all usual connectivity found in any desktops or laptops (USB, Ethernet, video interface...etc).

6.2 Related Work

Multitude of works has contributed into improving visual odometry. Surveys and more recent tutorials give a detailed picture of the different approaches involved [14]–[16]. In this Section, we mainly focus on real time stereo feature based visual odometry approaches in the first instance and then more specifically on embedded/online contributions.

In order to avoid use of computationally expensive statistical methods to reject outlier, [66] achieved real time using a strong Euclidean constraint combined with dense stereo to select inliers from a set of initial 3D correspondences. Another major contribution based, this time, on feature tracking has influenced largely the following works in this domain is given in [2]. This contribution introduced many improvements among which including the use of RANSAC in the motion estimation stage for outlier rejection. Nister has also changed the processing of relative motion to 3D points projection into a two dimensional camera pose problem while it is used to be seen as three dimensional point registration problem only [2]. Indeed, as seen in Chapter 4, minimising 2D image re-projection errors is more accurate than minimizing 3D feature correspondences errors. Following the same methodology of Nister, a successful implementation of visual odometry was made possible for Mars Exploration Rovers [98]. Even if all visual odometry process latency was around 3 minutes (from image acquisition until the camera pose generation), it is effectively real time considering the relatively slow motion of the rover in addition to the limited specifications of the hardware. In [99], as part of DARPA LAGR program, a stereo based visual odometry approach was fused with IMU and GPS information through a Kalman filter to avoid long term drifts for outdoor long trajectories. In this work, real-time is achieved in part by using a closed form of SVD computation matrix, which is the most time consuming task of their process. Howard [12], used the same methodology as [66] with an improved inliers selection scheme based on groups of consistent matches enabling faster point to point comparison. Also, this implementation gives impressive position errors lower than one percent on long term trajectories. More recently, an interesting real-time 3D reconstruction of a trajectory from a stereo video was enabled using an efficient dense stereo matching combined with multi-view images from visual odometry algorithm in [100].

In the works cited above, real time performance is achieved running on CPU processor based on desktop, laptop or directly integrated to the robot [2], [66], [98], [99]. A visual odometry implementation for small robots uses an OMAP3530 board, which has the particularity of being composed of a DSP (C64) and an ARM (Cortex A8) [101]. This work takes advantage of the two board components in splitting different tasks between those two components. Dense stereo is done by the DSP while feature detection, matching and ego-motion are computed by the ARM. Thus, stereo vision and visual odometry are

parallelized which enables faster execution of the whole process. The motion estimation algorithm used for this contribution is the one developed in [12]. The whole visual odometry algorithm processes 512x384 (0.2 MPixel) at 6 Hz.

More recently, [102] presented an independent stereo vision and IMU perception unit equipped with the same OMAP3530 board. It is also equipped with an FPGA board and an Intel Core2Duo 1.86 GHz CPU board. In the same philosophy as [101], each board has been attributed a task. ARM is collecting and integrating IMU data, FPGA board is computing disparity image using Semi Global Matching (SGM). CPU tasks consist in stereo images acquisition, feature detection and matching, then ego-motion. The visual odometry algorithm used in their work follows the same methodology as in [66]. However it improves the process by fusing IMU data with visual odometry through an EKF in order to compensate for the delay of the vision pipeline and to strengthen the state estimation. Processing 1024x508 (0.5 MPixel) stereo images, the total visual odometry runs at 5 Hz.

In our work, visual odometry stages are split between CPU and GPU devices. Feature detection and features tracking suit very well to be parallelized operations according to the sparse and independent nature of features. Additionally, use of IMU data to predict feature location in subsequent stereo image pair increases feature tracking efficiency while decreasing processing time. In contrary to [101], [102], we do not use dense stereo to generate disparity maps as it remains an expensive operation in terms of computation even when running in a dedicated device. In fact, we preferred a sparse approach enabling us to track up to 750 initial features on high resolution images (1.2 MPixel) in real time. Regarding the visual odometry algorithm, we demonstrate in similarly to [12], [101] that a two frames approach is enough to achieve accurate ego-motion.

In this chapter, we present an innovative smart and robust navigation system solution equipped with high-resolution stereo cameras (1.2 MPixel) and assisted with an inertial measurement unit (Figure 6.1). The solution is controlled with a single board computer (SBC) with a 1.9 GHz Dual Core CPU and an NVIDIA chipset integrated GPU. The development of this solution is to provide a real time and accurate localisation in a GPS denied environment. As mentioned in the Chapter 1, European Space Agency (ESA) has requested the developed solution within a collaborative Space research program we are involved in. Following our previous work in the navigation domain with ESA, a

requirement to develop a robust localisation sensor that can be mounted on different platforms (Mobile robots, manipulators, hand-held…etc) was critical. High sampling rate of the IMU enables us to get an inertial estimate of the motion between two stereo image pairs through the integration of gyroscope and accelerometer data. Based on these data, we proposed following the methodology developed in Chapter 5, to combine them with reconstructed 3D features from 2D stereo matches from the previous pair. This provides us a relative precise guess of the previous detected features 3D position at the current time. The latter are re-projected into the current stereo pairs resulting in a 2D feature guesses to the KLT feature tracker. Thus, we contributed at providing a faster tracking for KLT as well as low frame rate processing. Furthermore, in addition to what was presented in Chapter 5, the feature detection, stereo tracking and feature tracking (Section 2.7) process is accelerated under a GPU scheme. Consequently, we are able to run our standalone localisation sensor and process images at their highest resolution in real time. This device, relying only on passive sensors, does not depend on any external source of information such as GPS and provides a solution for indoor and outdoor exploration tasks.

6.3 Navigation System Hardware

In this section, hardware components selected to develop our standalone navigation system are presented. This will be followed by a discussion regarding the reasons behind our choice. As it has been mentioned in the introduction, several constraints need to be considered in order to design an independent and flexible visual navigation system. It requires being smart, with minimum footprint but also powerful enough to manage inertial and vision sensors as well as handling the whole visual odometry pipeline in real time.

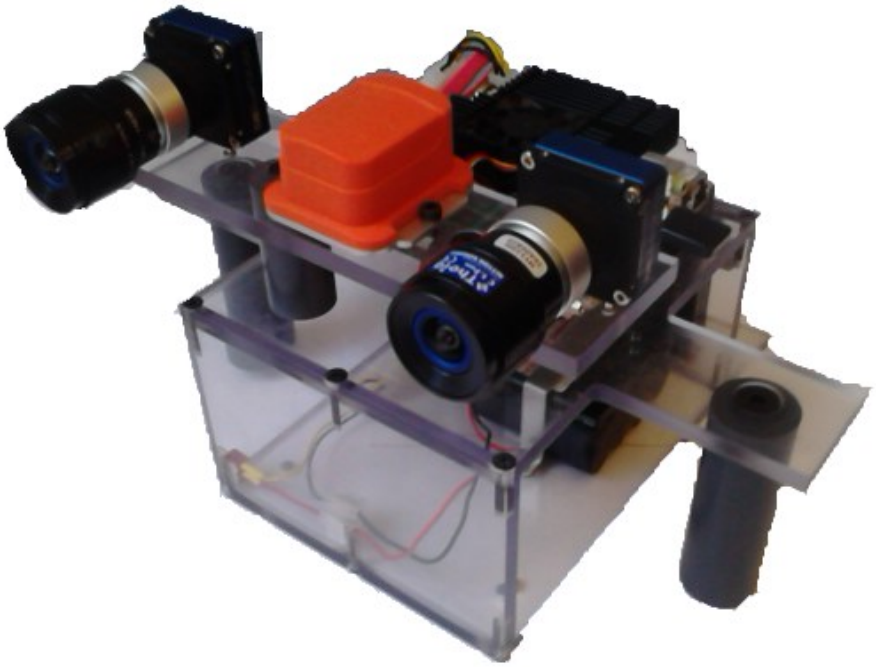


Figure 6.1. *Illustration of the proposed smart visual/IMU standalone navigation sensor.*

6.3.1 Board Selection

Choosing components that suit best our application is essential. This choice is not straightforward due to the large range of available hardware, which can be combined with each other. In our case, our preference went on the nano ITX single board computer (SBC) for several reasons. ITX boards offer a large flexibility in size, processors, and peripheral connectors. For example, ITX SBC standard models start at 170x170 mm in for mini ITX. Then, comes smaller models at 120x120 mm for nano ITX. Pico ITX is 72x100 mm size and finally the mobile ITX size is 45x75 mm. Regarding processors choice, it depends on what one's budget is given in addition to computation constraints.

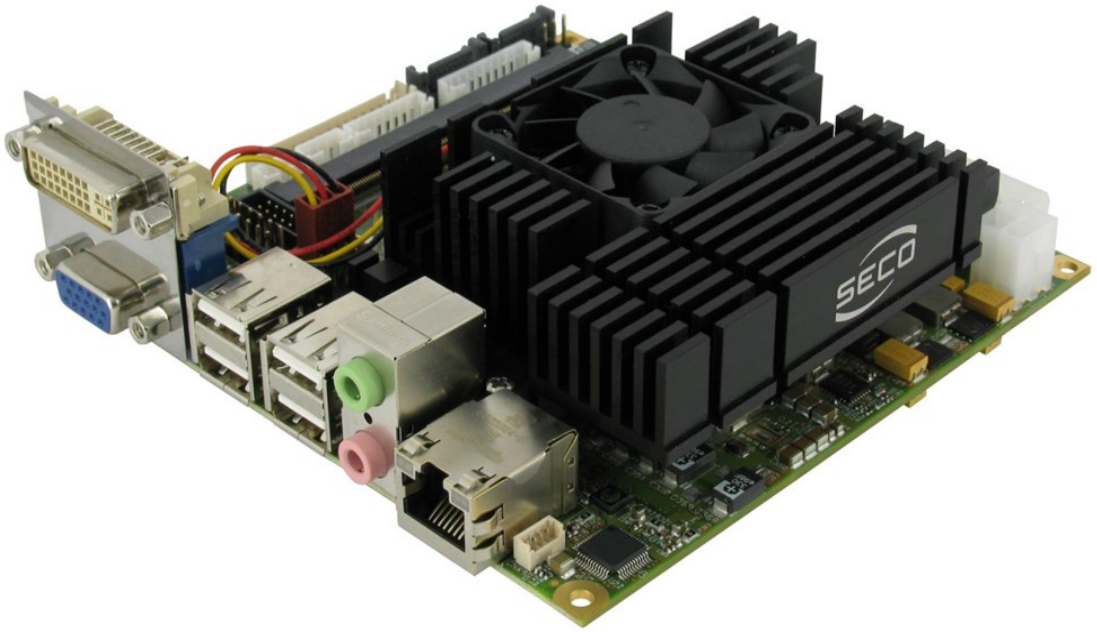


Figure 6.2. *Illustration of the selected SECOITX-ION Single Board Computer.*

Indeed, the latest ITX can support 4th generation of Intel i5 or i7 processors. Also other Intel processors such as Atom or Celeron along with other types of processors such as AMD, ARM, Cortex, Freescale IMX are available.

In our case, we choose a SECO brand nano ITX SBC (SECOnITX-ION) which has an Intel Celeron Dual Core T3100@1.9 GHz CPU processor but also an NVIDIA MMP9 Embedded ION chipset and a 16 cores GPU integrated controller NVIDIA® GeForce 9400M (Figure 6.2).

It gives the advantage of having quite a powerful CPU processors compared to available commercial SBCs. It also provides a small GPU processor compared to latest graphic cards. However, it revealed to be very beneficial and sufficient for our application. To take the maximum advantage from this SBC's capability, we added a 4 GB DDR3 memory which is the maximum that can be handled by the dedicated SO-DIMM socket. We used one of the two SATA sockets to which we connected a DELL PDA1000B 1TB portable external hard drive.

Figure 6.3. *Illustration of the selected battery pack (left), and power supply unit (right).*

The four USB connectors were connected respectively to the two stereo cameras, IMU sensors, and Wi-Fi dongle. The Wi-Fi dongle was used for SSH communication between SBC and an external laptop to launch and stop the visual odometry program. The SBC is powered via a +12 V_{DC} AT/ATX connector. We used a DC-DC picopsu-80-WI-32 (pico power supply unit (picopsu), 80W, wide input 12-32V), which has the great advantage to be small, silent, fan-less and with a very small footprint (Figure 6.3).

Two modes of operations were designed:

- Development mode: when we needed the board to be powered for long time during development and testing phases, picopsu is connected to an 80W AC adapter via a P4 to DC jack connector.
- Experiment mode: when we use our sensor navigation system in real time for dataset acquisition and visual odometry trajectory generation, picopsu is connected to a Tenergy Li-ION 14.8 V 5200 mAH battery pack (Figure 6.3) via a P4 connector linked with power switch enabling or not powering of the SBC. This battery pack enables us to run the visual navigation system a little bit longer than an hour, which is very useful for long runs.

Figure 6.4. *Illustration of the selected cameras (left), and lenses (right).*

6.3.2 Camera Selection

In visual applications where cameras have a central role, it is essential to select the ones with characteristics that fulfill the requirements. In the case of visual odometry, the visual system is prone to navigate in different indoor and outdoor environments where illumination conditions often change. These repeated changes or transitions between environments might be challenging for visual sensors and can lead to biased or corrupted images acquisition. Another constraint when designing a navigation system is the dimension of the visual sensors. We opted for MvBlueFOX-IGC USB 2.0 cameras based on the CMOS Aptina MT9M034 image sensor with a resolution of 1280x960 pixels (Figure 6.4). They present an advantageous compact design. We also equipped these cameras with Theia MY110M ultra wide angle 1.7 mm lenses with a field of view of 110°x94° (Figure 6.4). These ultra wide lenses provide also very low distortion, which is very beneficial during stereo rectification process. In a visual odometry context, having a wider field of view is really appreciable. It gives more information of the scene content. Consequently, this gives a higher probability of having detecting salient features remaining for longer on the camera's field of view than with standard lenses. Furthermore, with a larger field of view overlap between stereo cameras considerably facilitate stereo correspondences.

Figure 6.5. *Illustration of the selected inertial measurement unit.*

Also an entire C++ API that enables tight control of the different camera functionalities such as capture, frame rate, exposure, gain, time stamping and time delay is provided. The fact that it is coded in C++ facilitates integration of camera functions within our program that is also based on C++.

6.3.3 IMU Selection

The inertial measurement unit plays an important role in our solution, by assisting the feature tracking operations. The chosen inertial device is an Xsens Mti-G. It is an integrated GPS and IMU with Navigation, Attitude and Heading Reference System (AHRS) processor (Figure 6.5). It is based on MEMS inertial sensors including also 3D magnetometer and a static pressure sensor as well as a miniature GPS receiver. In terms of dimension its compact size (60x50 mm) suits very well with our navigation system design requirements. Xsens Mti-G is linked to our navigation system via an RS232 to USB converter. Note that in our solution, we only use calibrated accelerometer and gyroscope data (no GPS is enabled). Indeed, software integration of Xsens Mti-G to our algorithm is made using the provided SDK which are coded in C. It gives us a relative flexibility on data selection as well as handling remarkably well time stamping and data transfer time.

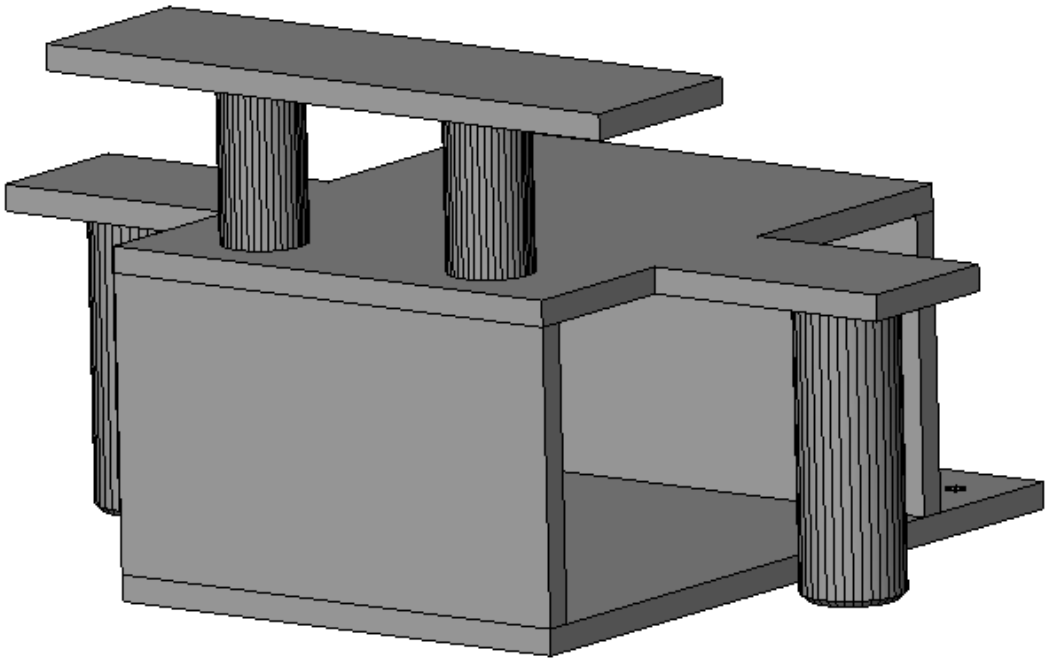


Figure 6.6. *Illustration of the navigation system structure designed with SolidWorks software.*

6.3.4 Whole Hardware Structure Design

The structure carrying all the components has been designed in a cubic form for convenience with a rectangular stereo plate on its front top and two handles on its sides allowing handheld navigation as illustrated Figure 6.6 and 6.7.

The two cameras are placed on each side of the rectangular stereo plate in a way that the stereo baseline is 16 cm. This chosen distance combined with the ultra wide angle provided with the Theia lenses give a good compromise between design compactness and stereo vision properties. The Xsens device reference frame was carefully aligned as accurate as possible in the middle of the baseline structure to facilitate reference frames transformations with the left stereo camera.

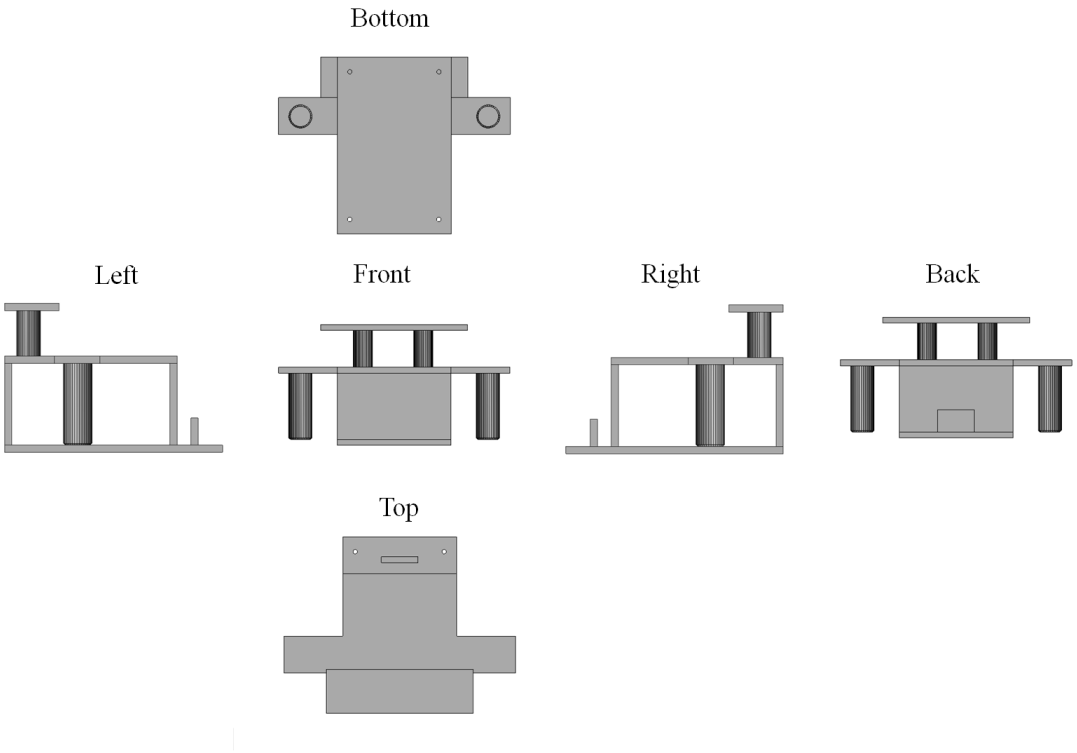


Figure 6.7. *Detailed views of the navigation system structure designed with SolidWorks software.*

The latter is taken as the main reference frame. The SECOnITX-ION SBC is mounted on the top of the structure in order for the SBC's fan not being obtruded, allowing thus the SBC avoid heating thanks to a good air circulation. However, this is not the only reason. The SBC occupies a central role and needs to be placed in such way that it is accessible to all the other components. Indeed, its four USB connectors host the stereo camera, inertial sensor, and Wi-Fi dongle. Its SATA port hosts the portable external hard drive, which is placed at the back of the structure. The battery pack is fixed inside the cubic structure (in between the two handles). The pico-psu unit enables empowering the SBC through a +12V p4 connector but also the portable external hard drive via its HDD power connector. Thus, according to its paramount role, it is natural for the SBC to be positioned in the centre of the whole structure. Four holes have been added to the whole structure on the bottom plate to eventually screw this navigation system on moving robotic platforms. Dimensions of the whole structure is 16x20 cm (approx.) excluding the handle "wings". The total weight is about 2.5kg (approx.).

The chosen sensors as mentioned above have been selected for their suitable dimensions and specific features. However, it was really important for us to build up a framework including all the hardware that uses the same programming language. Indeed, using the SDK for the inertial sensor and C++ API for the stereo camera enabled us to identify transmission time to the host. Besides, time-stamps are generated using Boost library [105]. Combining all this information gives us the opportunity to synchronize sensors information.

6.4 Navigation System Software Development

The software part of our navigation system divides the visual odometry pipeline operations between CPU and GPU memories (Figure 6.8). Two threads are created in CPU memory. The main thread is in charge of image acquisition from the stereo camera and manages the visual odometry algorithm. The second thread handles IMU data acquisition. Our visual odometry algorithm starts with image acquisition of the most recent stereo pair I_{cL} and I_{cR}. Each time new images are captured, the main thread gets from the second thread inertial measurements that have been accumulated between two stereo image pairs ($\{I_{pL}, I_{pR}\}$ and $\{I_{cL}, I_{cR}\}$).

Calibrated accelerometer and gyroscope measurements are acquired at a frequency of 100Hz and are given a timestamp before gravity compensation stage. Individual timestamps and known transfer time delays enable us to synchronise inertial and visual data. By integrating inertial data we obtain an inertial motion estimation matrix composed $R(q_{imu})$ and t_{imu}. This provides an estimate of the motion that happens between the previous and current stereo pair. This matrix is used later in feature tracking stage. Newly acquired images are undistorted and rectified using the cameras calibration matrix [106]. Once current stereo image and the inertial motion ($R(q_{imu})$ and t_{imu}) are obtained in the CPU main thread, these are transferred (in addition to the previous stereo image) to GPU memory. Feature detection is run on I_{pL} using GPU implementation of good features to track [21]. Then, matches are tracked on image I_{pR} using a GPU implementation of pyramidal KLT [24]. A small filtering function discards wrong matches which do not validate epipolar and spatial constraints. This results in a consistent set previous feature pairs s_p.

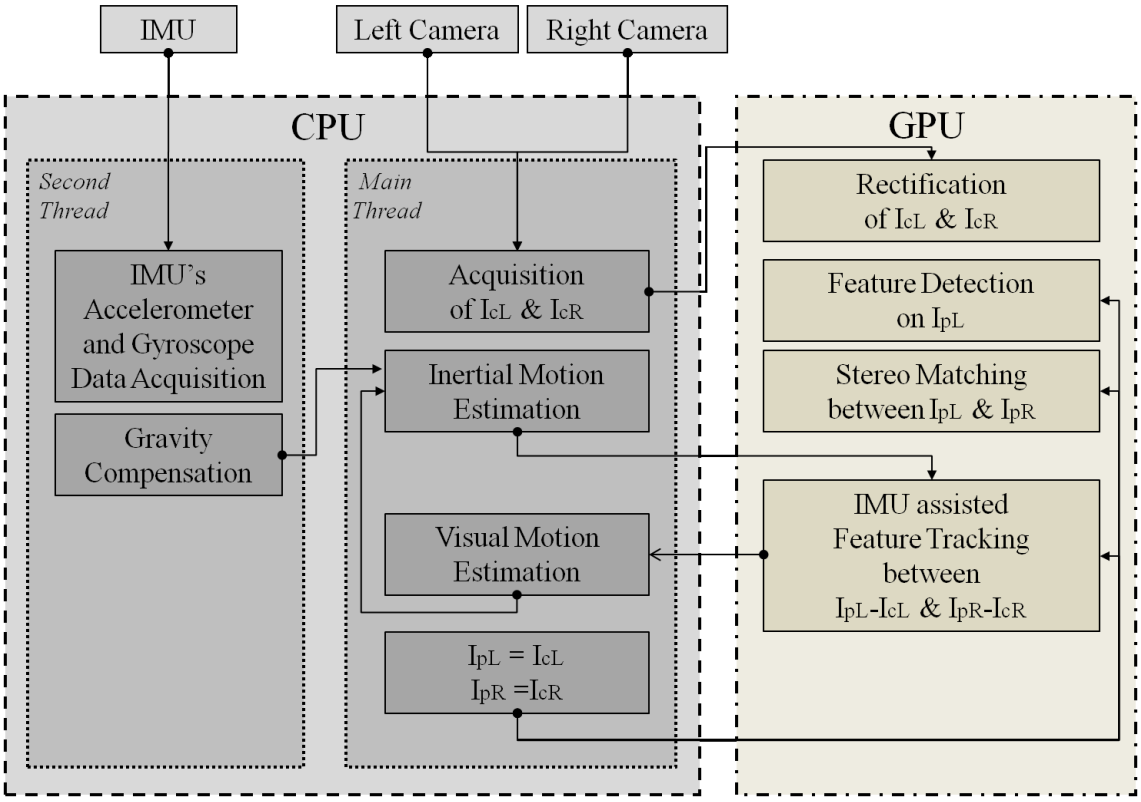


Figure 6.8. *Software implementation structure.*

This set is then combined with the recently obtained inertial motion ($R(q_{imu})$ and t_{imu}) in order to obtain the set of inertial estimated features s^*_c. The resulting set of guesses s^*_c feeds the GPU implementation of pyramidal KLT algorithm where feature tracking operation is led between respective previous and current images ($\{I_{pL}, I_{cL}\}$ and $\{I_{pR}, I_{cR}\}$). A post filtering function checking for the epipolar and spatial constraints is conducted again. However, this time it concerns the current stereo image. As a result, a final set S^*_{pc} which consists of pre-filtered previous and inertial current features is constituted. S^*_{pc} is transferred to CPU memory as input to the motion estimation function. Visual motion estimation is computed by minimising re-projection errors between consecutive stereo pairs. Velocities are then calculated from the resulting visual motion estimation R_v and t_v, and serve as initialisation for the integration step in the next inertial motion estimation stage. R_v and t_v are accumulated in a global motion matrix Rt_{global} memorising the full trajectory done by the intelligent navigation sensor.

6.4.1 System Framework

Software and hardware communication is not an easy and straightforward task. Nevertheless, hardware manufacturers provide linkers such as drivers, SDK, and API to facilitate interconnection. However, each sensor has its own protocols and classes. Generally, these linkers enable a deep control of the sensors helping users to exploit all their functionalities. In addition to sensors communication links, the programming language of the main program and the different libraries to be used are essentials. One of the objectives when designing a software framework is to standardise intercommunication between all the involved components. Having said that, there are robotic frameworks which aim to unify all robotic components and to implement links between them. Among them we can find R.O.S (Robot Operating System) used at Willow Garage, Player, Urbi or Orca for the most famous ones. These are also called robotics middleware and presented as a kind of "glue" to connect hardware and software parts of a complex robotic system. They include a large range of robots and hardware (cameras, lasers, audio...etc) that is continuously increasing. Despite this remarkable standardisation effort for the majority of sensors, only basic functionalities are reachable through these middleware. For our application we wanted to have a full access the cameras and IMU functionalities in order to better control and optimise the best utilisation of data streams. This is why we have implemented our own linkers using the camera's API and the IMU's SDK.

Our main program is coded in C++. We use computer vision functions from OpenCV C++ library. POSIX Thread is used for thread management and C++ Boost POSIX time library for timestamp generation and time related operations. These were carefully chosen to use the same programming language but also to be portable on Linux or Windows operating systems. Our program was tested offline on Windows 7. However, it is an Ubuntu 12.04 LTS version that is installed in our navigation system to run our visual odometry algorithm online.

6.4.2 IMU Assisted KLT Feature Tracking

Here, we give a reminder of feature detection-stereo matching-feature tracking scheme used in our visual odometry algorithm and presented in Chapter 5. Usually, feature

detection, stereo matching and feature tracking are the most time consuming operations of a visual odometry pipeline. The strategy that has been deployed aims to execute these three operations with a minimum time while keeping high performance. Also, in the choice of used techniques we deliberately avoid robust but computationally expensive techniques such a SURF or SIFT. Indeed, even if there are GPU implementations of these techniques they remain by their nature slower than other GPU implemented feature detection techniques. This is why we opted for a GPU implementation of Good Feature to Track [21]. Indeed, we found the latter provides a good compromise between speed and robustness allowing us to compute up to 750 features in our visual odometry pipeline. A great number of features increases the probability to have homogeneous distribution of features in the acquired scene, which is essential in visual motion estimation stage.

We detect features on previous left image Il_{k-1} and track them on the right image Ir_{k-1} using GPU implementation of pyramidal KLT. This allows us to save one operation by skipping feature detection on previous right image. Even if it might lead to very few odd stereo matches, a large majority of correspondences are correct. High-resolution images play an important role on the quality of stereo matches found with the KLT approach. The proposed two steps operation is more time-efficient than computing descriptors for two images or than computing dense stereo. Also except a fast post filter discarding incoherent matches on epipolar and spatial constraints, we do not need extra operations. These latter are usually required in stereo matching that have to classify and compare different candidates on right image for only one feature on left image. This enables us in part to process a great number of features in real-time. Once stereo correspondences have been filtered, we get a set of 2D features between previous stereo image. Then 3D positions of the features composing this set are recovered by triangulation. The next stage introduces inertial motion estimation.

Inertial motion estimation resulting from integration of the MU readings (accelerometer and gyroscope), is combined with the set of 3D features from previous stereo image following the equation of motion. A 3D inertial post motion points are obtained. Finally, this new set of inertial estimated 3D positions are re-projected into the image plane of current stereo image pair. At the end we get a set of 2D inertial feature guesses. Figure 5.2 is summarizing this process.

When re-projecting features into current stereo image, we remove from the set the ones that fall outside the image plane. Remaining features are used as guess locations in pyramidal KLT feature tracking stage. Consequently, this reduces slightly the computation cost of the function. Indeed relatively accurate guess locations bound search area is used and dedicated search time to locate the right candidate is decreased.

6.4.3 Visual Motion Estimation

Given the group of feature correspondences between consecutive stereo image pairs, the camera motion is computed by minimising errors resulting from their re-porjection. In this two frames approach bundle adjustment, only motion parameters (composed of 4 quaternion and 3 translations) are optimised. RANSAC is used to determine a set of inliers. At each iteration, three feature correspondences are randomly chosen. We apply rotation and translation transformation obtained from motion parameters to the set of 3D features in previous stereo image pair using the equation of motion. The resulting 3D positions are then re-projected on the current stereo image pair. Then, we iteratively minimise the sum of errors of features re-projection using our introduced double dogleg trust region method [73], [75]. If it converges, we obtain the camera motion estimation and update the motion parameters. Then an inliers selection process is carried out using last the motion parameters. If the norm of the error re-projection sum of a feature correspondence lies under a certain pixel threshold, it is considered as an inlier. Motion parameters giving the highest rate of inliers are kept and then used in a motion refinement stage using only the inliers. From the local camera motion obtained, we derive relative translations to get local velocities that serve in initialising subsequent inertial motion estimation.

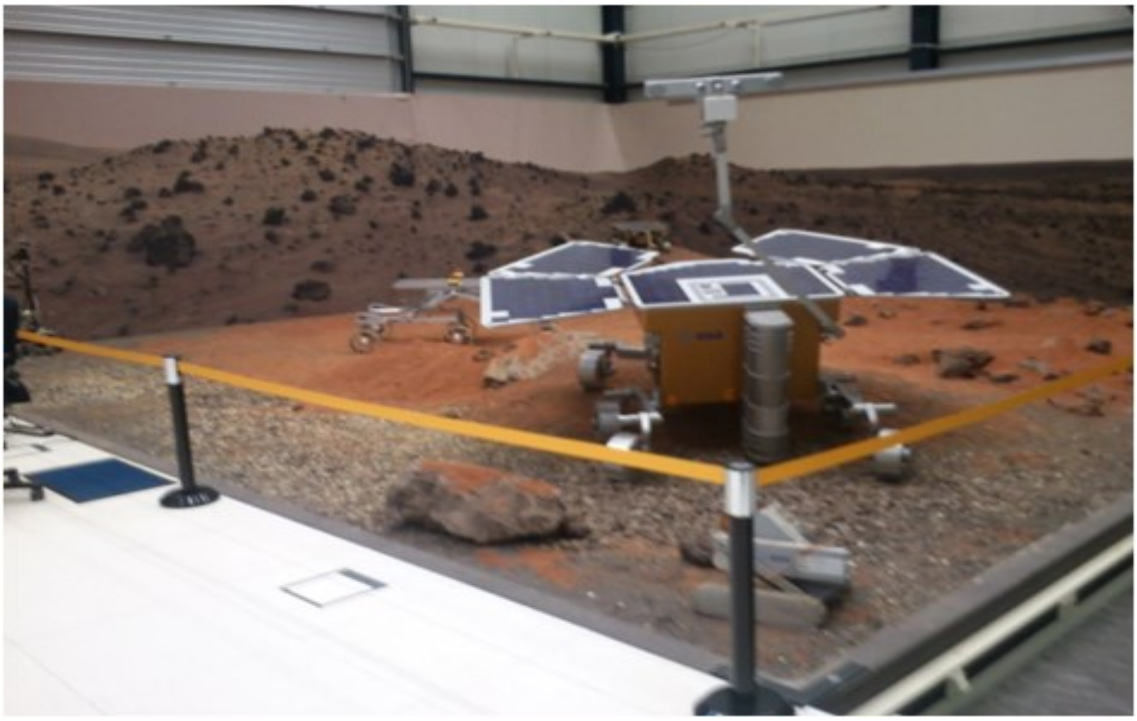


Figure 6.9. *Mars test-bed at ESA (European Space Agency).*

6.5 Experiments and Results

The assessment of the designed visual navigation sensor performance is divided into two parts. The first part focuses on runtime analysis of the full visual odomerty pipeline. The second part evaluates visual odometry accuracy in trajectory generation.

The visual navigation system was run in experiment mode (Section 6.3.1). We connected to the navigation system via SSH from another machine to launch the visual odometry program. This connection is made possible with the Wi-Fi dongle plugged on one of the SBC's USB port. Experiments took place in ESA's laboratory equipped with a 9m square pool reproducing a Mars ground like environment (clay, rocks...etc, see Figure 6.9). The visual navigation system was handheld and runs were made walking around the pool following specific paths. Working in conditions that try to reproduce Mars visual environment led us to face some of those challenging aspects such as limited feature environment or texture less zones. Also, non-homogenous pool's ground creates instability. It made the navigation system subject to recurrent and sometimes sudden variation in height due to clay bumps, holes or slipping surfaces when walking on it.

Figure 6.10. *Illustration of a 50 mm spherical retroflective marker.*

In this section, more specific evaluation of the proposed visual motion estimation technique is given with the comparison between Double Dogleg and Levenberg-Marquardt algorithms.

6.5.1 Experiments Set Up and Vicon System

In order to validate our solution in term of accuracy, we need to have a strong and reliable navigation reference. This is provided by the Vicon motion capture system (Bonita) that equips the ESA laboratory and which consist of 10 networked Infrared (IR) cameras. During my stay at ESA, I had to participate to the full setting up of Vicon system, including cameras installation, cabling, system calibration, and data acquisition. In this section, I briefly relate the main steps I went through in order to establish these excellent experimental conditions.

As mentioned above, ESA laboratory has 10 cameras available, 7 of them are 1.3 Megapixel resolution (MX13 +) and the three remaining are 2 Megapixel resolution (MX20 +). They also provide a high frame rate capture capability up to 100 fps, which provides an efficient capture system. Theses IR cameras track 50 mm spherical retroflective markers (Figure 6.10) that appear isolated from the scene background because of their high reflectivity. Thus, the 3D positions (X, Y, Z) of one or more markers can be precisely recovered by triangulation through the Vicon Nexus software.

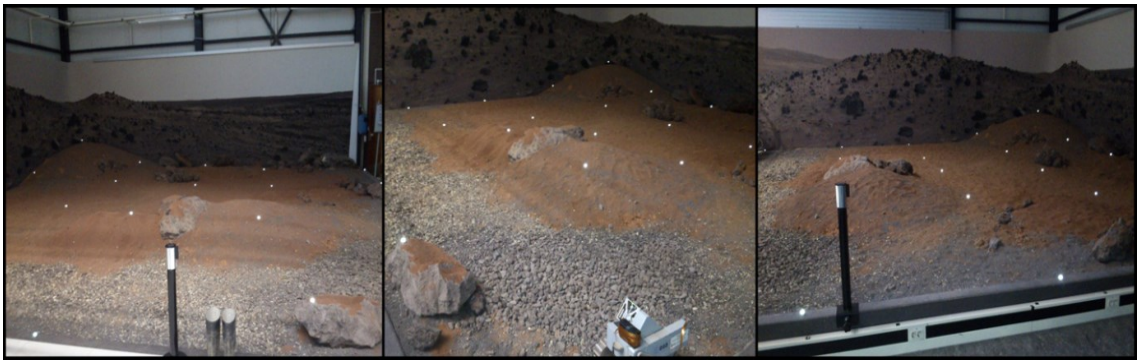


Figure 6.11. *Three different views of the pool after markers placement at the cameras position adjustment stage.*

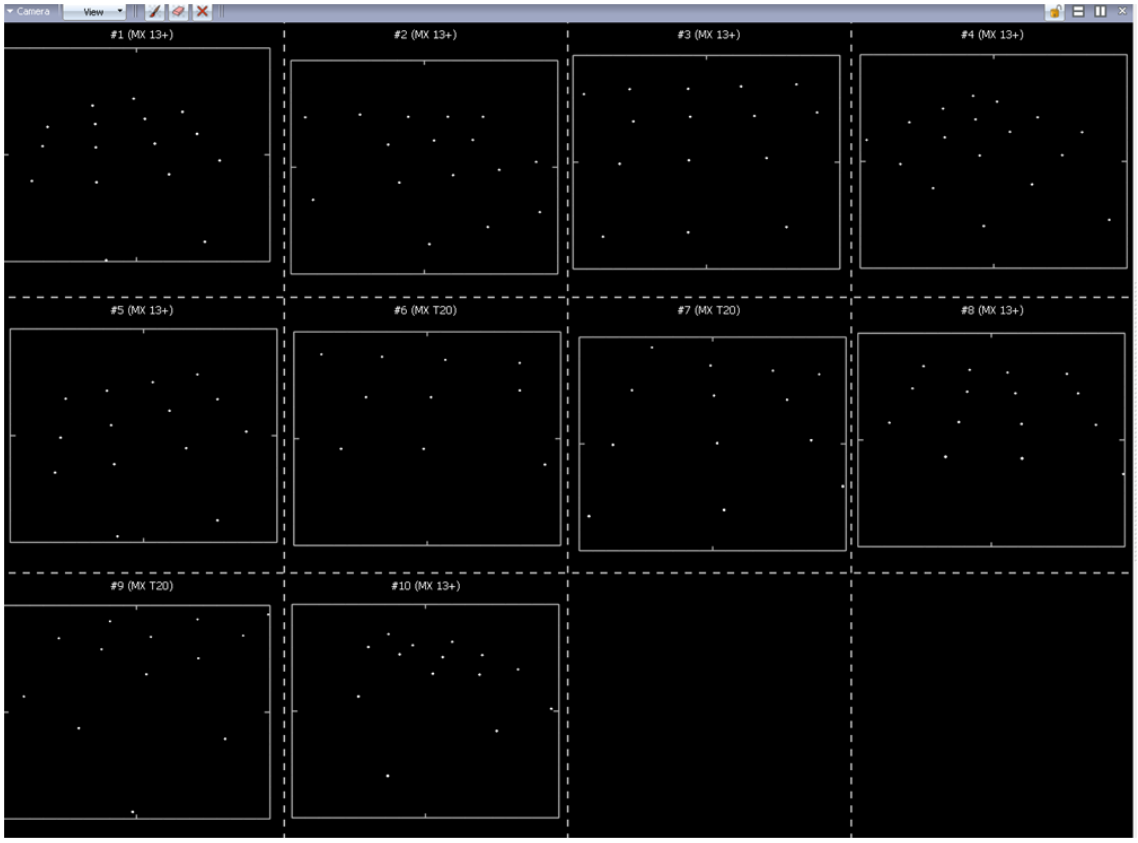


Figure 6.12. *View of the Vicon Nexus software display of each camera field of view at the cameras position adjustment stage. Markers are represented by white dots.*

However, and in order to achieve the triangulation, markers need to been seen from at least 2 cameras. According to this, the cameras were placed around the pool in such way

that the full volume is covered by the camera's field of views. In order to check if the cameras field of views were effectively covering the full pool's surface, it has been decided to place markers on the floor over the whole full surface (Figure 6.11).

Then, with the help of Vicon Nexus software each camera was checked, allowing us to see how many and which markers appear for each of them (Figure 6.12). This gave us a relatively good idea of the camera field of view orientation but more importantly to adjust the cameras positions and orientations to get better volume coverage. The cameras were considered well positioned after checking as each marker in the pool is seen from at least 3 cameras (one more than the minimum required).

Once the final positioning and orientation of the cameras were set, cameras were linked to the networking devices. The latter consist of an Ultranet and a Giganet that supply communication and synchronisation for maximum of 8 cameras via Gigabit Ethernet. The host desktop where the Vicon Nexus software is run is linked to the Ultranet, which is considered as a primary networking device. The 7 MX 13+ cameras are connected to it. The Ultranet is also linked with the Giganet, which is considered as secondary networking device. The remaining MX 20+ cameras are thus linked to the Giganet. The final architecture of the Vicon set up is illustrated in Figure 6.13 and 6.14.

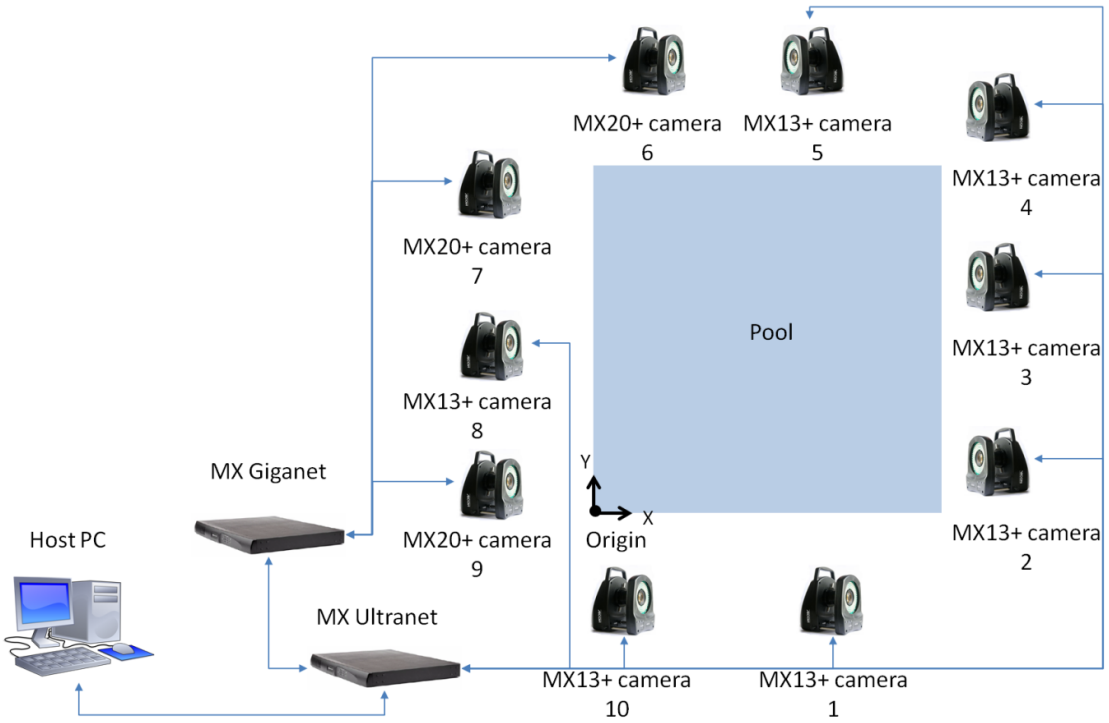


Figure 6.13. *Final Vicon architecture.*

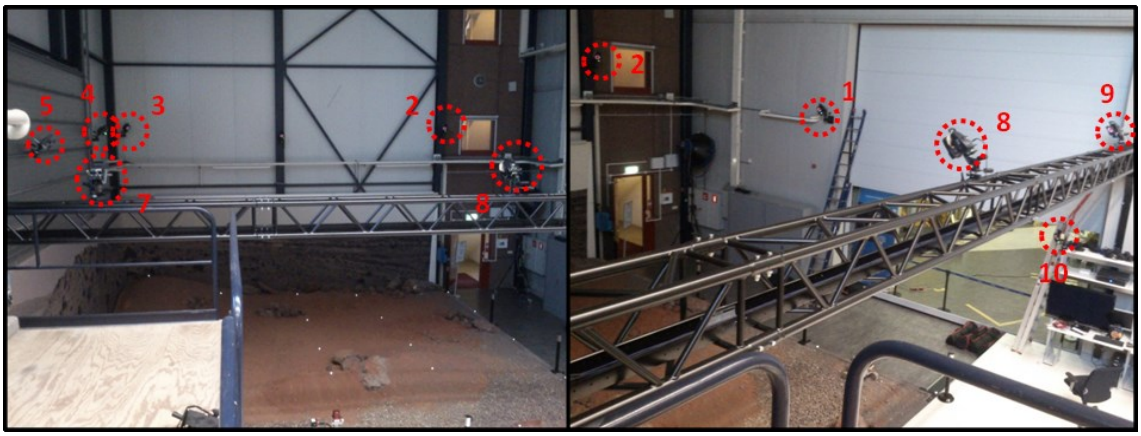


Figure 6.14. *Two views of the final cameras' position.*

Figure 6.15. *Origin (0,0,0) set up through the 4-markers l-frame*

The calibration stage enables us to obtain the internal and the external camera parameters of each camera. External parameters (cameras position and orientation) are used in the Vicon Nexus software to determine the position of each camera to each other regarding a set origin (Figure 6.15). The internal parameters (focal length and image distortion coefficients) are used in the 3D reconstruction process.

In order to do the calibration process, a 5-marker wand (Figure 6.16) is moved all over the pool while Vicon Nexus software calculates the image error projection of each camera. In order to have a good volume representation, the 5-marker wand needs to be moved up and down and also cover the maximum of the available surface (Figure 6.17).

The calibration process was stopped when the image error projection for each camera dropped below 0.01 pixel (Figure 6.18).

Figure 6.16. *5-markers t-frame wand.*

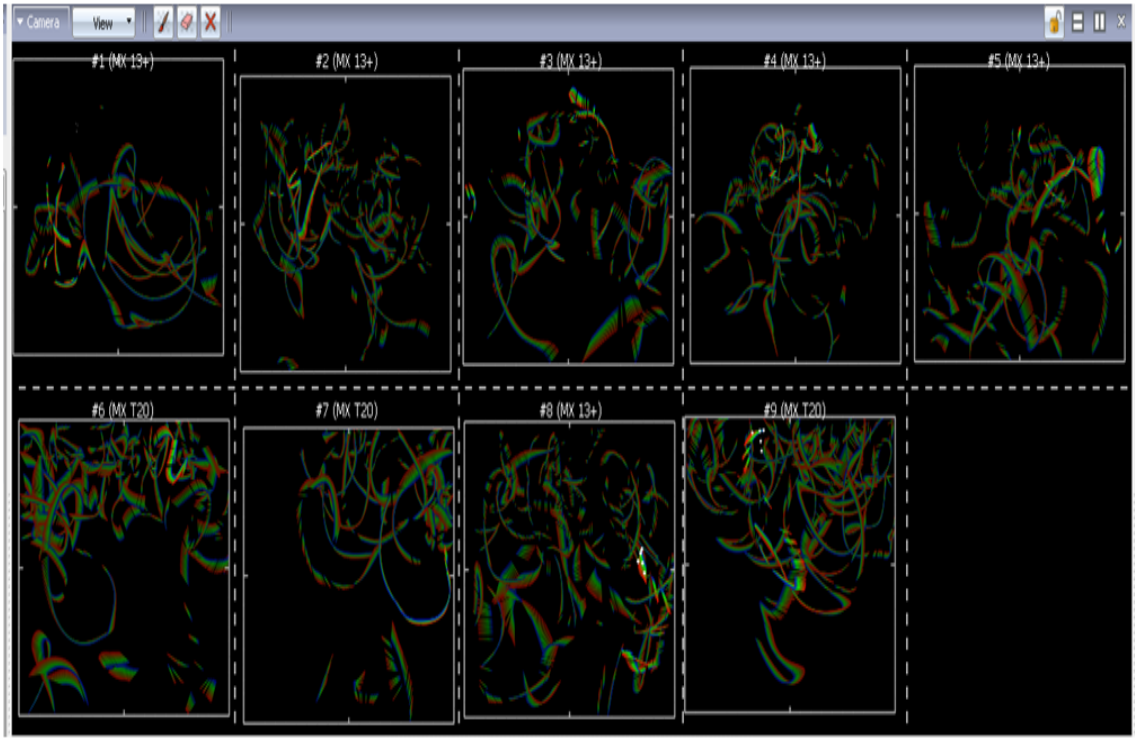


Figure 6.17. *Vicon Nexus software display of the 5-markers t-frame wand mark history for each camera.*

Camera	Wand Count	Image Error
#1 (MX 13+)	1371	0.0614136
#2 (MX 13+)	1727	0.07951
#3 (MX 13+)	1281	0.0699427
#4 (MX 13+)	1314	0.0904978
#5 (MX 13+)	1391	0.0701684
#6 (MX T20)	2874	0.0840696

Figure 6.18. *Vicon Nexus software display of the image projection errors for each camera.*

At the end, Vicon Nexus software gives 3D representation of the full volume as well as the camera positioning according to the pool (Figure 6.19). In order to represent the navigation sensor within the Vicon Nexus software, four markers were placed on the visual navigation system. Two on the left camera on its opposite sides, one on the top of the IMU and the last one on the external side of the right camera (Figure 6.20). Markers have been placed in such way that only the stereo plate is represented. We are mostly interested by the left camera position. Having the markers on the top of the structure is suitable for the Vicon capture. These markers are more likely to be fully visible by the different cameras.

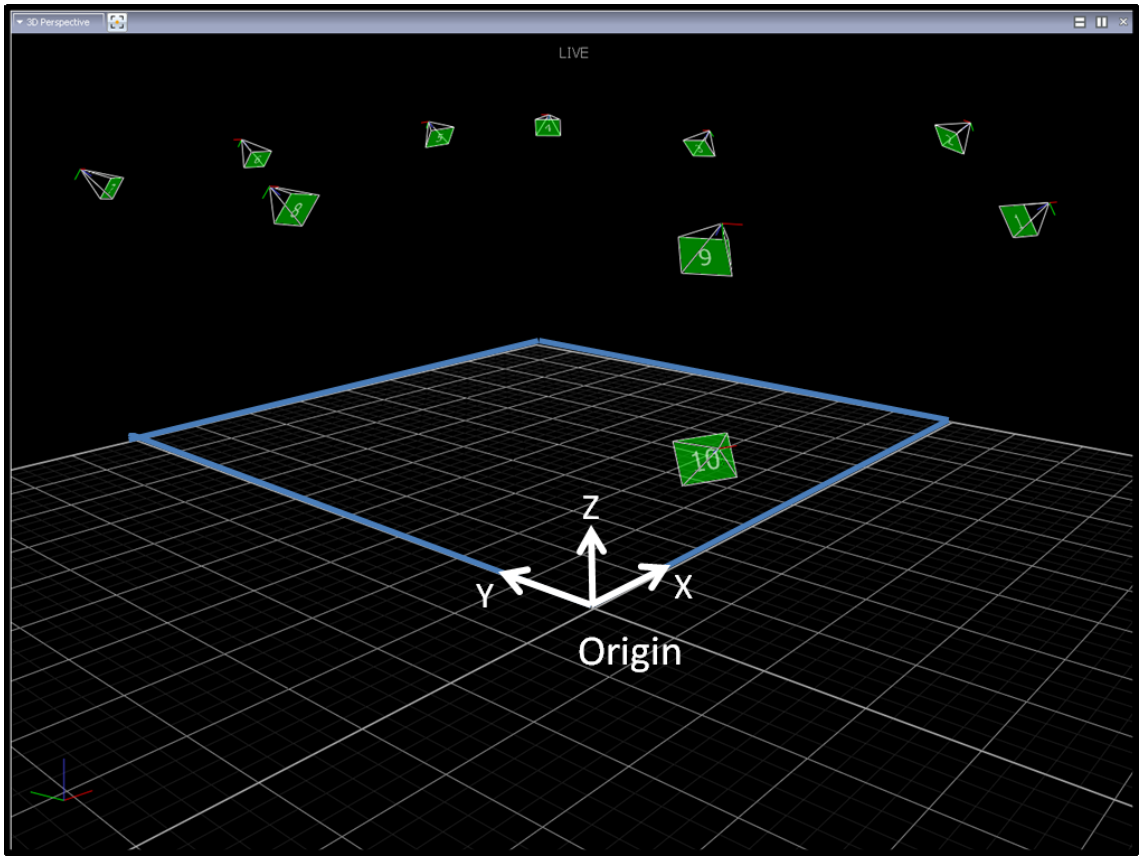


Figure 6.19. *Vicon Nexus software display of the Vicon MX camera setup according to the pool (blue squared perimeter).*

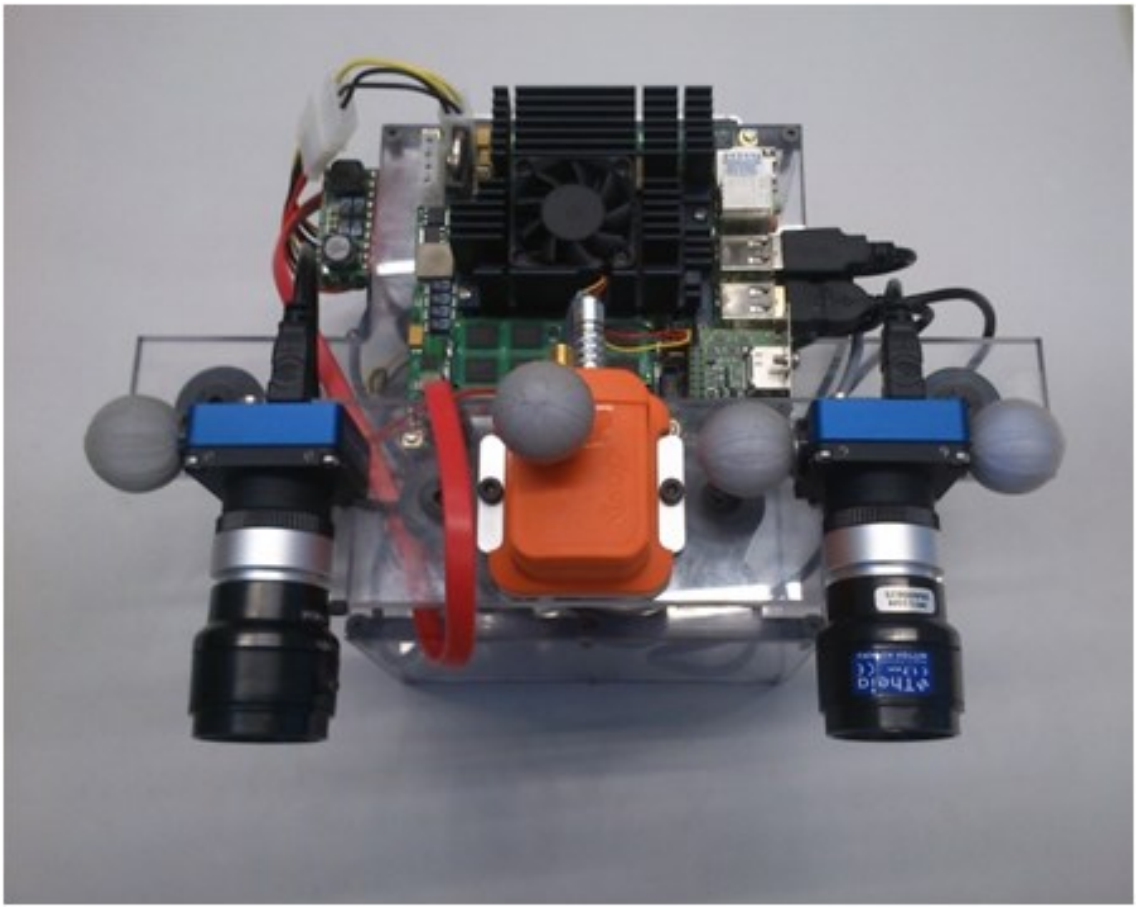


Figure 6.20. *Marker positions for our navigation sensor.*

6.5.2 Visual Odometry Runtime Performances

In this section, we give a detailed runtime analysis of each step of the visual odometry pipeline for two representative runs (A and B). According to the specifications of the chosen components that form our visual navigation system, we wanted to use each of them at their full capabilities in order to get the best benefit. For instance, the maximum frame rate at full resolution (1280x960) of MvBlueFOX cameras is 24 fps. In reality, this frame rate is particularly very challenging and not required to achieve accurate trajectory generation. However, our objective is to process full resolution stereo images while achieving visual odometry at reasonable frame rate allowing us to produce precise ego-motion.

The runtime of the complete visual odometry pipeline is given in Table 6.1 and illustrated in Figure 6.21 for run A.

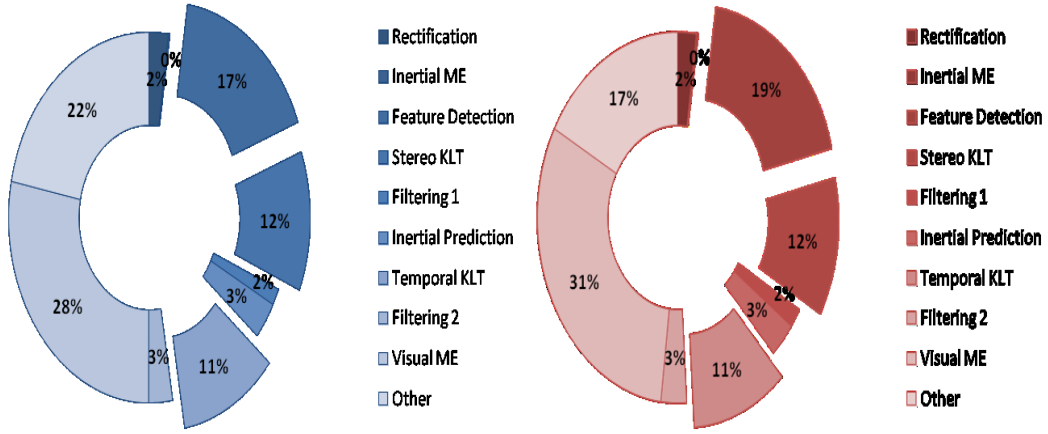


Figure 6.21. *Run A: runtime breakdown in percentage of visual odometry pipeline using double Dogleg algorithm (left) and Levenberg-Marquardt algorithm (right) for visual motion estimation stage.*

In Figure 6.21, feature detection, stereo KLT, and temporal KLT are run on the GPU device and represent almost half of the full visual odometry runtime. This shows the importance of using parallel programming to achieve real time. Visual odometry runtime depends highly on the number of features processed. Hence, the main visual odometry stages are features dependent. Thus, features sparse approach suits very well parallelisation, which proves to be very efficient in this context.

In Table 6.1, we can obverse that except for the visual motion estimation stage, runtimes of the visual odometry pipeline using the two different techniques are almost equivalent. Indeed Double Dogleg algorithm is faster than Levenberg-Marquardt. Table 6.1 shows that a very satisfying 5fps is reached at a resolution of 1.2 Megapixel starting with 750 detected features. Comparable recent works such as [101], or [102] reported equivalent frame rate with lower resolution images and less initial features. In [101], visual odometry pipeline runs 512x384 (0.1 MPixels) stereo images with 400 initials features at approximately 6 fps. In [102], with a more equivalent hardware configuration to ours, it reaches approximately 4.5 fps with 1024x508 (0.5 MPixels) stereo images and 500 initials features. It can be noticed, that choosing to skip feature detection for the right image showed to be a good proposition. Indeed, even if it is run on GPU memory, feature detection stage is the second most time-consuming task after visual motion estimation

(17%). Carrying feature detection on both stereo images would probably not double this runtime as it is run in parallel but will significantly increase it.

Table 6.2 shows that the number of inliers is significant starting from correctly stereo tracked features (88.5% using double Dogleg and 89.4% using Levenberg-Marquardt). Also, in [101], their algorithm which uses a more robust dense stereo algorithm finishes with 140 inliers for 400 initial features.

Table 6.1. *Run A: Runtime of Visual Odometry Pipeline at 1280x960 Resolution and Starting with 750 Initial Features.*

	Using Double Dogleg		Using Levenberg-Marquardt	
	in ms	in %	in ms	in %
Rectification	4.4	2	4.4	2
Inertial ME	0.3	0	0.4	0
Feature Detection	33.5	17	38.8	19
Stereo KLT	24.1	12	24.6	12
Filtering 1	2.8	2	3.4	2
Inertial Prediction	6.4	3	6.3	3
Temporal KLT	22.5	11	22.6	11
Filtering 2	5.6	3	6	3
Visual ME	55.9	28	63.7	31
Other	43.5	22	35.1	17
Total	199	100	205.3	100
Frame Rate	5.02 fps		4.87 fps	

Table 6.2. *Run A: Average Feature Related Operations Results Through Visual Odometry Pipeline on 247 frames.*

	Detected Features	Correctly Stereo Tracked	Inliers
Using DDL	569	235	208
Using LM	569	235	210

Table 6.3. *Run B: Runtime of Visual Odometry Pipeline at 1280x960 Resolution and Starting with 750 Initial Feature.*

	Using Double Dogleg		Using Levenberg-Marquardt	
	in ms	in %	in ms	in %
Rectification	4.4	2	4.4	2
Inertial ME	0.4	0	0.6	0
Feature Detection	36.4	19	35.3	17
Stereo KLT	17	9	18.6	9
Filtering 1	3.2	2	2.7	1
Inertial Prediction	5.7	3	6.3	3
Temporal KLT	21	11	21.6	11
Filtering 2	6.1	3	5.5	3
Visual ME	63.2	32	69.9	34
Other	36.4	19	40	20
Total	193.8	100	204.9	100
Frame Rate	5.16 fps		4.88 fps	

A similar conclusion can be drawn in run B. Runtime of the complete visual odometry pipeline is given in Table 6.3 and illustrated in Figure 6.22 for run B. In Figure 6.22 it can be observed that runtime operations are proportionally similar to run A.

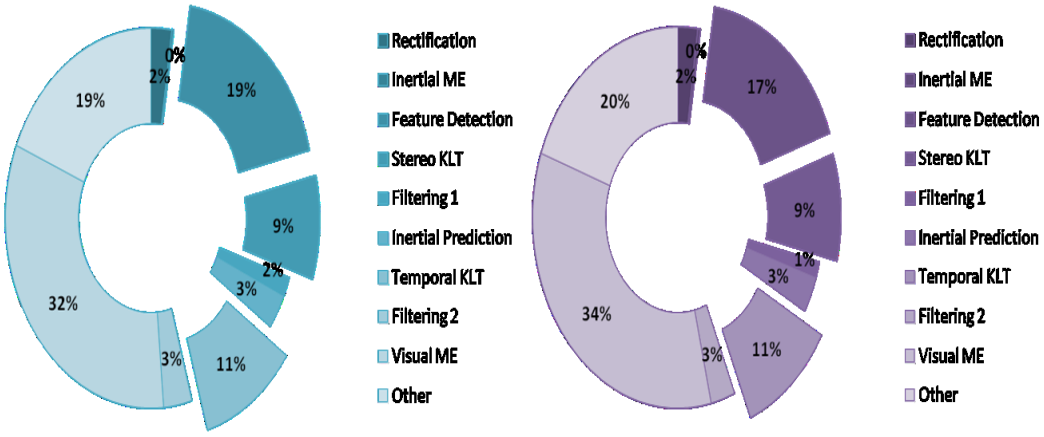


Figure 6.22. *Run B: runtime breakdown in percentage of visual odometry pipeline using double Dogleg algorithm (left) and Levenberg-Marquardt algorithm (right) for visual motion estimation stage.*

Table 6.4. *Run B: Average Feature Related Operations Results Through Visual Odometry Pipeline on 242 frames.*

	Detected Features	Correctly Stereo Tracked	Inliers
Using DDL	457	221	193
Using LM	457	221	195

Table 6.5. *Run A: Final Position Relative Error Comparison in Trajectory Generation.*

	2D relative error		3D relative error	
	in m	in %	in m	in %
DDL	**0.147**	**0.87**	**0.207**	**1.21**
LM	0.250	1.48	0.371	2.18
Geiger [100]	0.169	0.99	0.251	1.48

Indeed visual motion estimation is still the most computational expensive operation. It is also the operation, which impacts the most the overall runtime between double Dogleg and Levenberg- Marquardt algorithm. Table 6.3 still shows that the overall runtime allows us, here as well, to reach 5 fps at 1.2 Megapixel

The rate of inliers from stereo matches is still high with 87.3% using double Dogleg and 88.2% using Levenberg-Marquardt. This is quite a remarkable result regarding the 5 fps acquisition constraint resulting in large inter frame optical flow.

6.5.3 Visual Odometry Trajectory Generation Performances

For this assessment, we walked around the space pool for the two runs (A and B) closing the loop. Figure 6.23, shows the results of our embedded visual odometry based navigation system running with double Dogleg algorithm (in blue) and with Levenberg-Marquardt (in magenta), as well as with Geiger visual odometry solution [100] based on Gauss-Newton (in red) compared to the reference motion capture trajectory (in black).

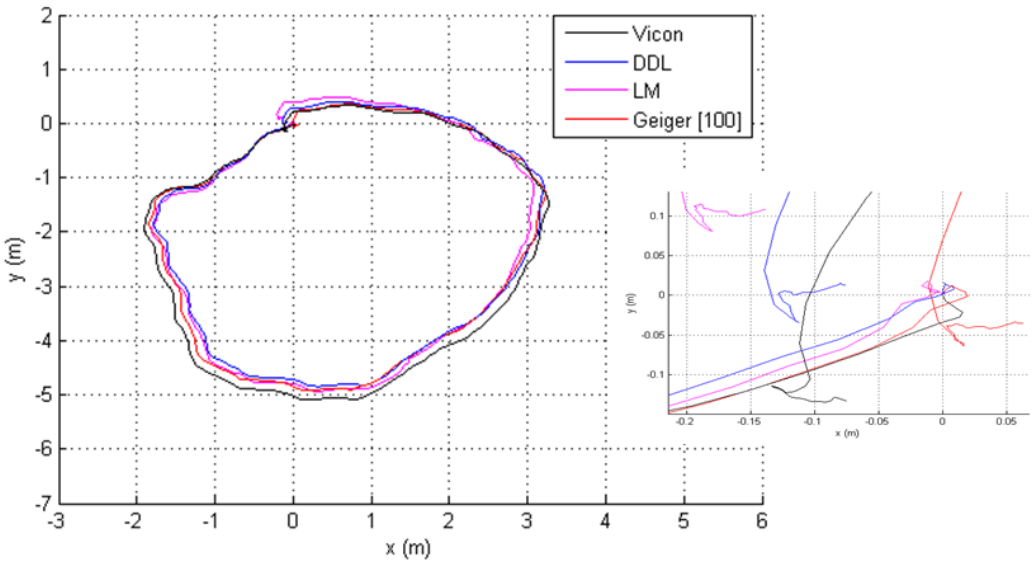


Figure 6.23. *Run A: 2D plot and zoom on final position of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) and Levenberg-Marquardt algorithm (magenta), compared to Vicon reference (black) and Geiger[100] visual odometry algorithm (red).*

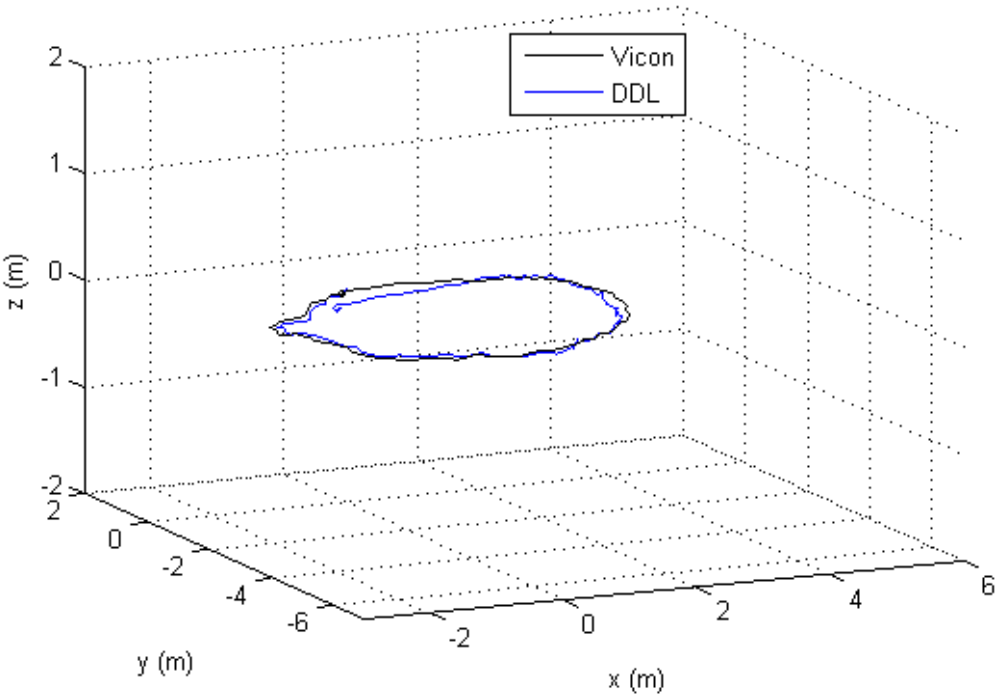


Figure 6.24. *Run A: 3D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black).*

All the compared trajectories give almost the same shape as the Vicon reference trajectory. However our trajectory remains closest to the ground truth for the major part of the route and lies at the final position only at 14.7 cm from the Vicon trajectory closing point for a total travelled distance of 16.95 m. In Figure 6.23, a zoom of the starting and final position in 2D is also given.

In Figure 6.24, where 3D trajectories are plotted, we can observe that our program using double Dogleg performs remarkably well in height estimation.

Figure 6.25 shows that the root-squared error remains constant over the time with of course some fluctuation. It also can be observed in Figure 6.26, a monotonic decrease of the error along the travelled distance. This allows us to reach at the end of the route a remarkable error below 1% of the travelled distance.

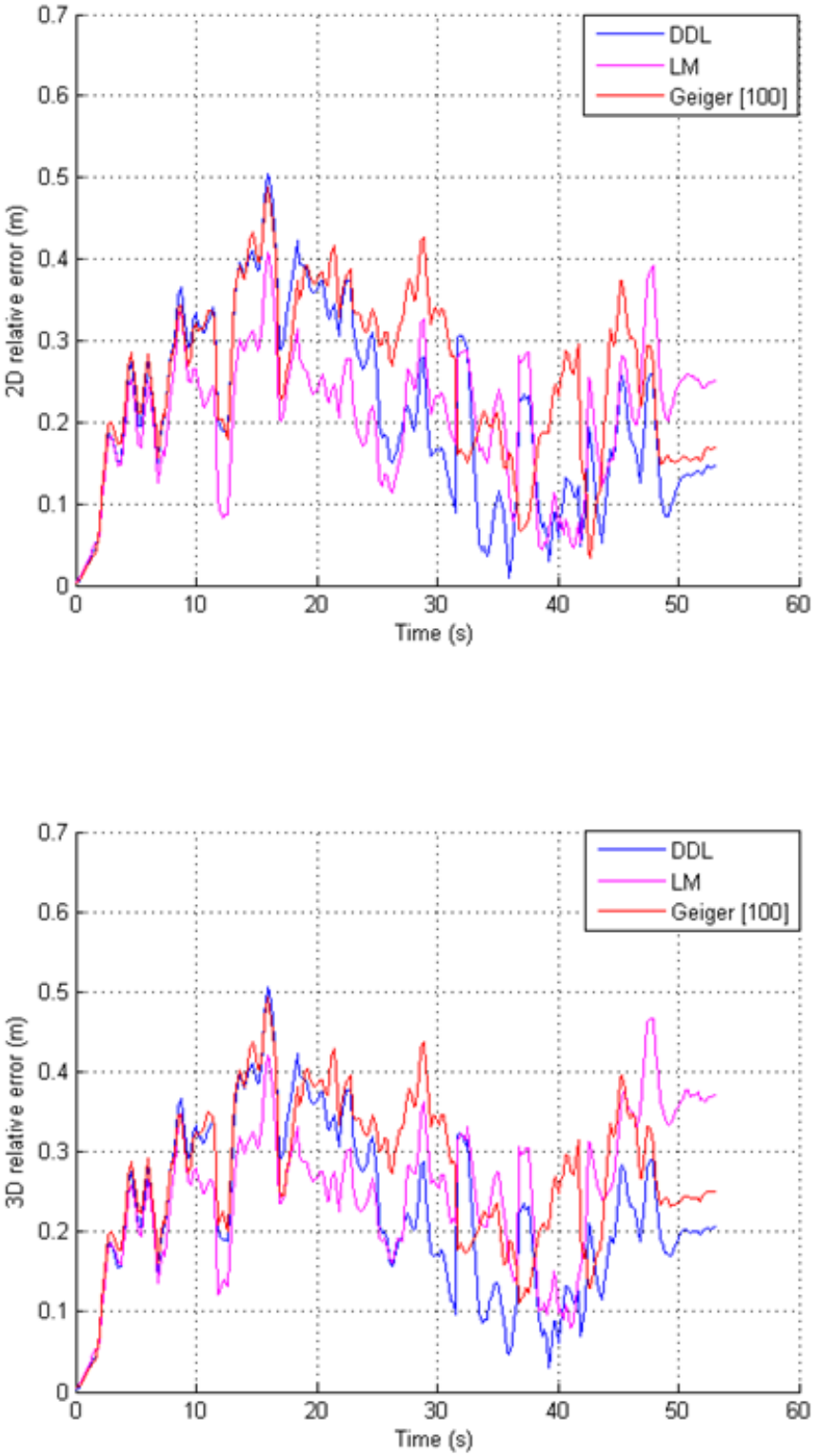


Figure 6.25. *Run A: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory.*

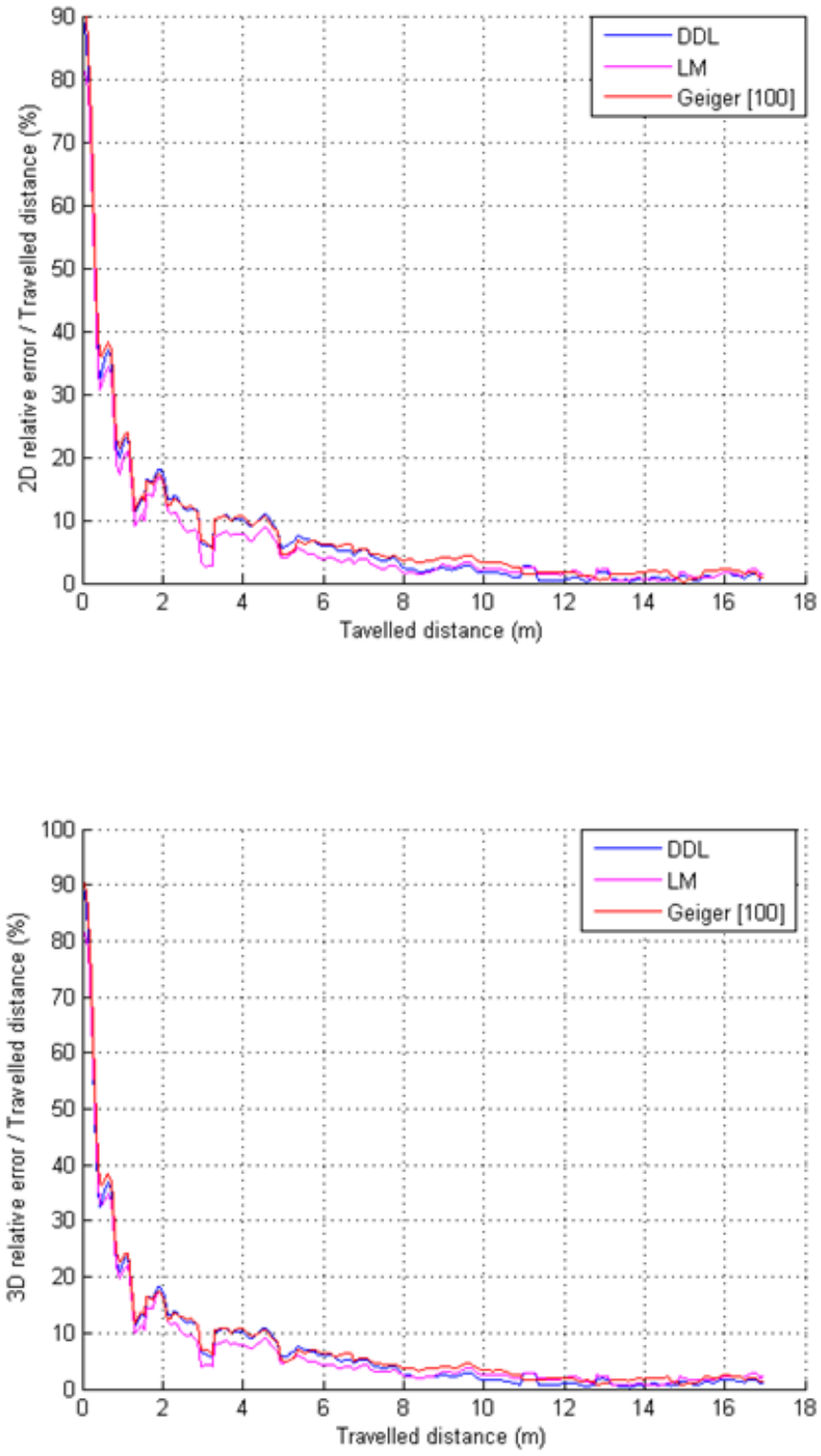


Figure 6.26. *Run A: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory.*

Figure 6.27, shows the results of our visual odometry algorithm running in our visual navigation system in run B. In this run, all the compared trajectories also give the same shape as the Vicon reference trajectory until a certain point. After this point only the trajectory generated with our navigation system using double Dogleg is remaining close enough from the ground truth, lying at the final position only at 7.6 cm from the Vicon trajectory closing point for a total travelled distance of 11.17 m (Table 6.6).

Our visual odometry based navigation system using Levenberg-Marquardt is not closing the loop but remains not too far from the 2D final position (23.2 cm, see Table 6.6). On the other hand Geiger visual odometry algorithm is finishing the farthest from the Vicon final point. This case (Run B) allows us to enlighten the utility of IMU information, which enables our navigation sensor algorithm to cope with this kind of uneven problem minimising the final trajectory to be affected too much.

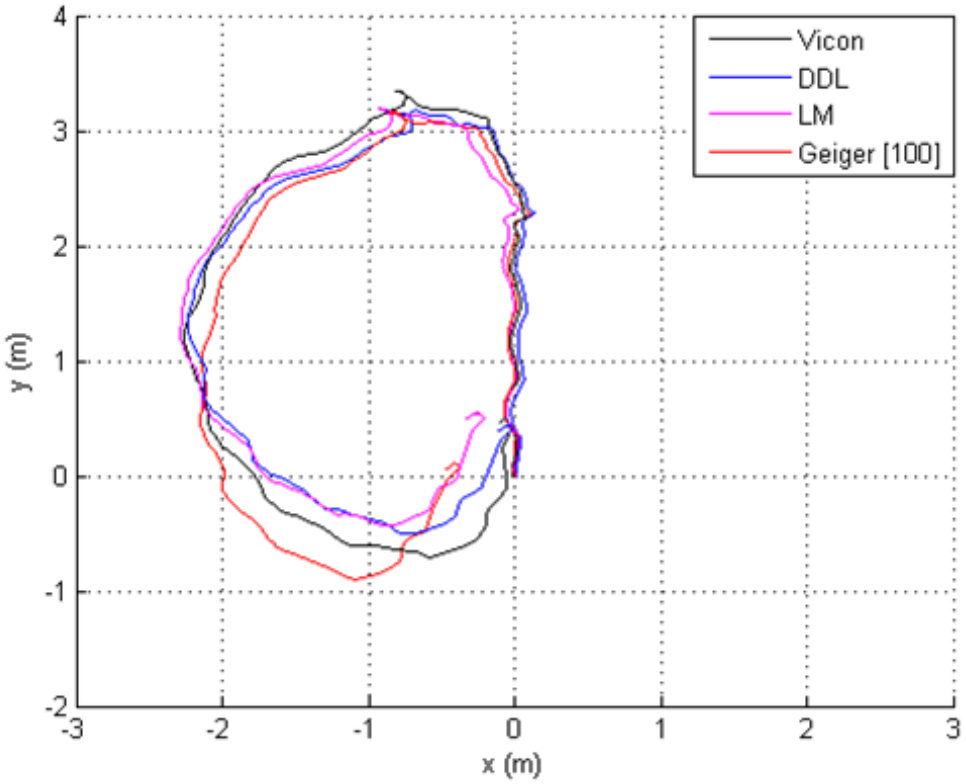


Figure 6.27. *Run B: 2D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) and Levenberg-Marquardt algorithm (magenta), compared to Vicon reference (black) and Geiger[100] visual odometry algorithm (red).*

This problem was caused by WiFi connection temporal loss which gave 1s delay between two frames. In Figure 6.28, where 3D trajectories are plotted, we can observe that our algorithm using double Dogleg performs remarkably well in height estimation expect at the end where the delay has a certain consequence on the Z axis.

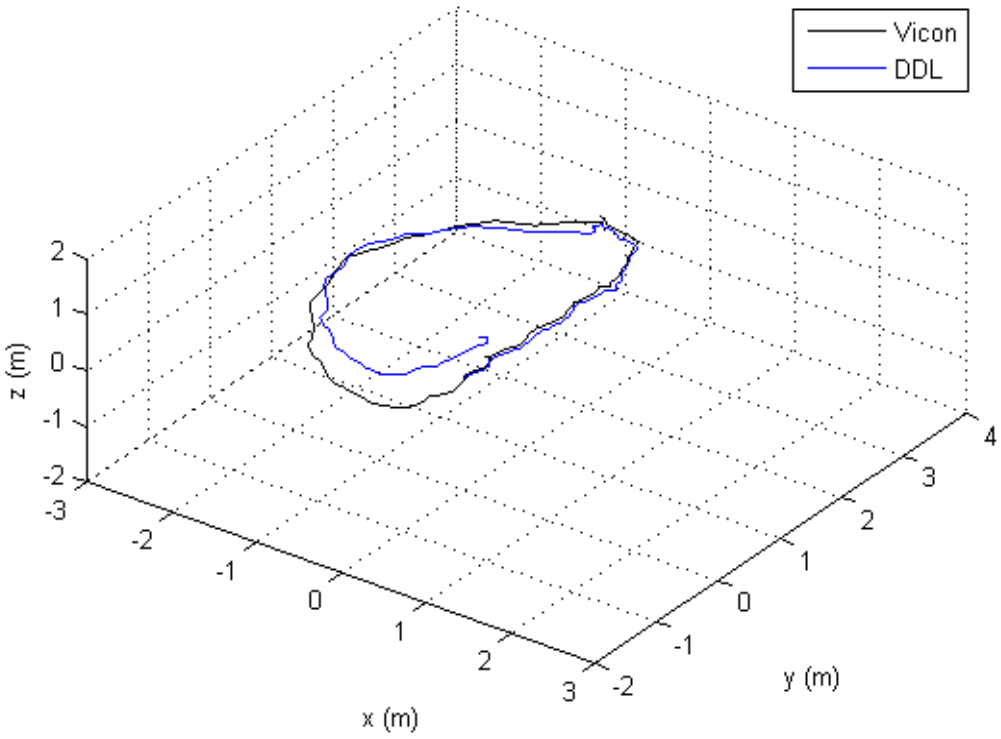


Figure 6.28. *Run B: 3D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black).*

Table 6.6. *Run B: Final Position Relative Error Comparison in Trajectory Generation.*

	2D relative error		3D relative error	
	in m	in %	in m	in %
DDL	**0.076**	**0.68**	**0.413**	**3.7**
LM	0.232	2.07	0.518	4.64
Geiger [100]	0.548	4.91	0.626	5.61

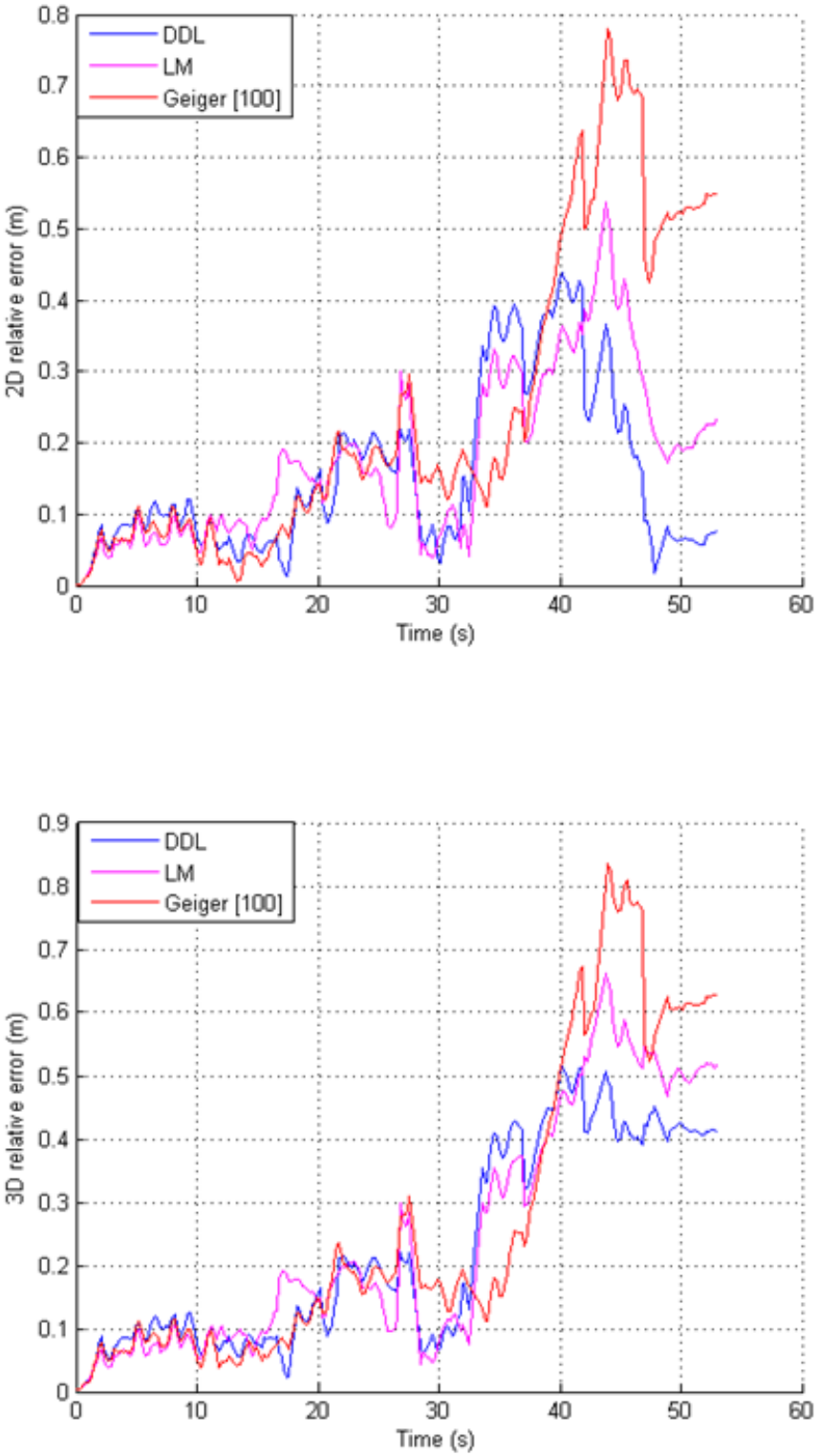


Figure 6.29. *Run B: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory.*

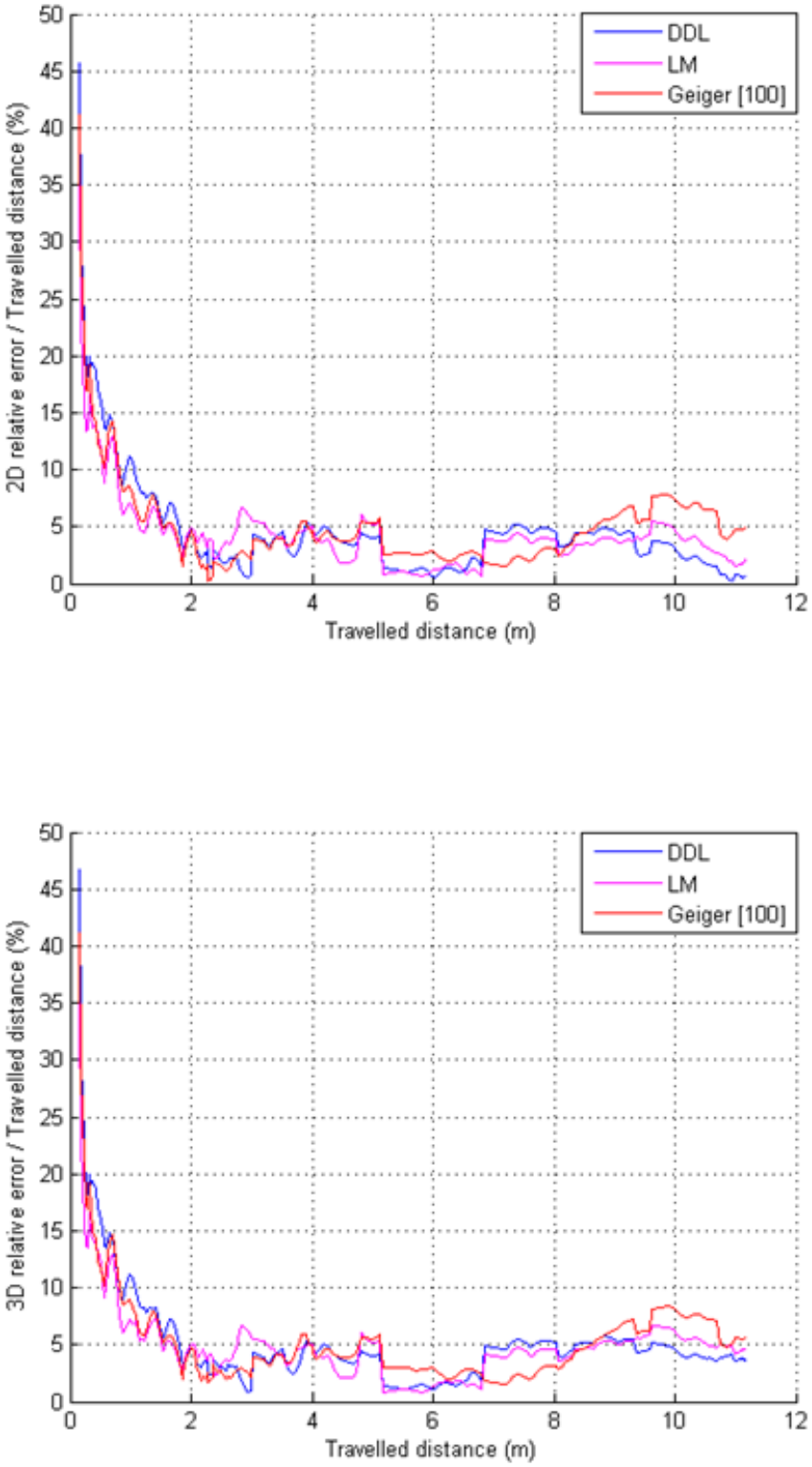


Figure 6.30. *Run B: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory.*

Figure 6.29 and 6.30 show that the root-squared error remains constant over the time with of course some fluctuation except at the end of the sequence. However, the navigation sensor solution using double Dogleg for 2D root square relative error is self-contained.

6.5.4 Visual Odometry Trajectory Generation with HDR Imaging

For this assessment, we evaluated the impact of the use of HDR imaging on as input for the full visual odometry process. For these datasets we walked in the pool in L-shape route were we compared trajectories generated with our visual odometry algorithm running with double Dogleg with and without activation of the HDR mode on the MvBlueFox stereo camera. This was done for two illuminations conditions: indoor lighting and dark conditions which gives for runs.

Figure 6.31 illustrates the rendering images of the pool with and without HDR mode activated for indoor lighting conditions. The HDR image (Figure 6.31 right) gives a clearer rendering of the scene and is not affected with reflection of the light on the wall.

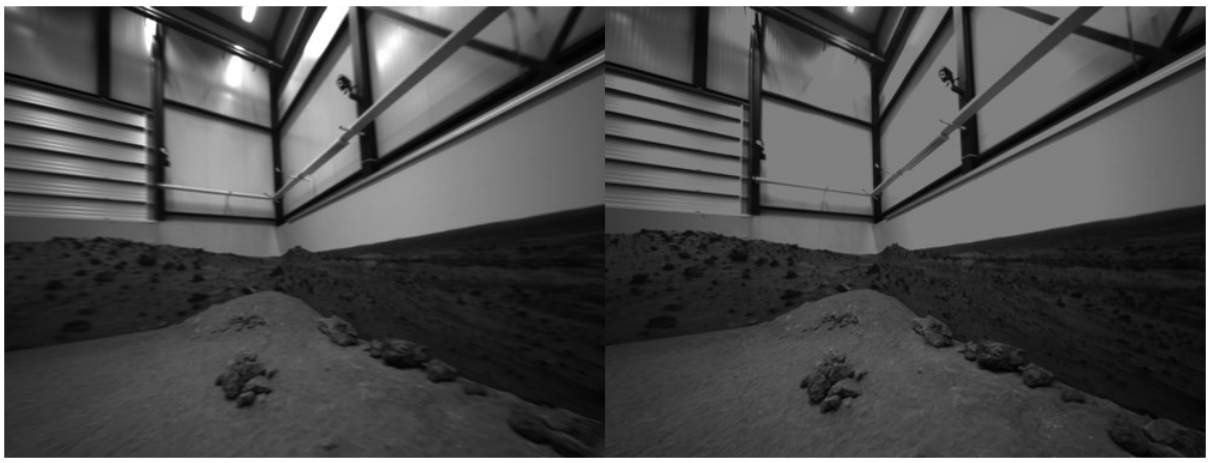


Figure 6.31. *Illustration of the image rendering of the pool without (left) and without (right) HDR mode activated for indoor lighting conditions.*

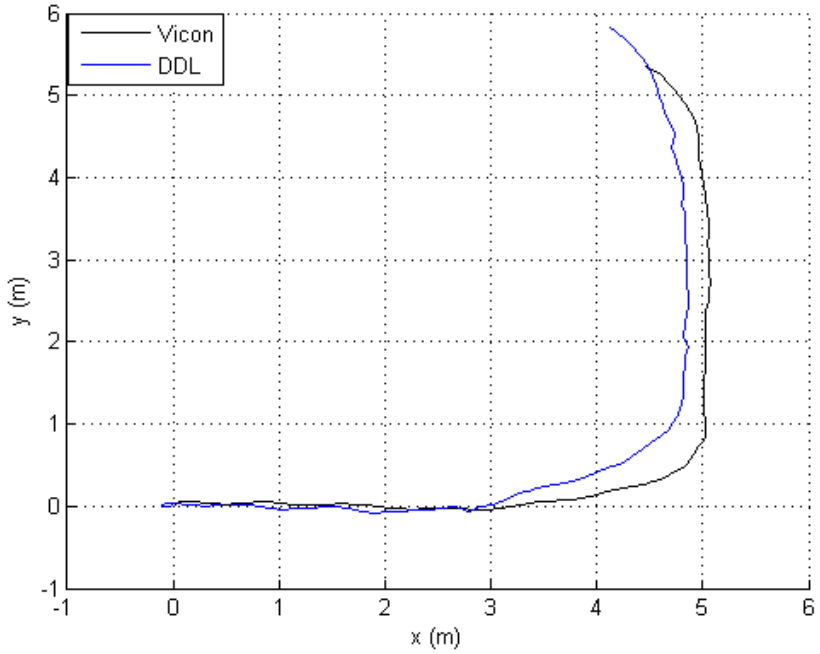


Figure 6.32. *Run 1: 2D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black) with standard images for indoor lighting conditions*

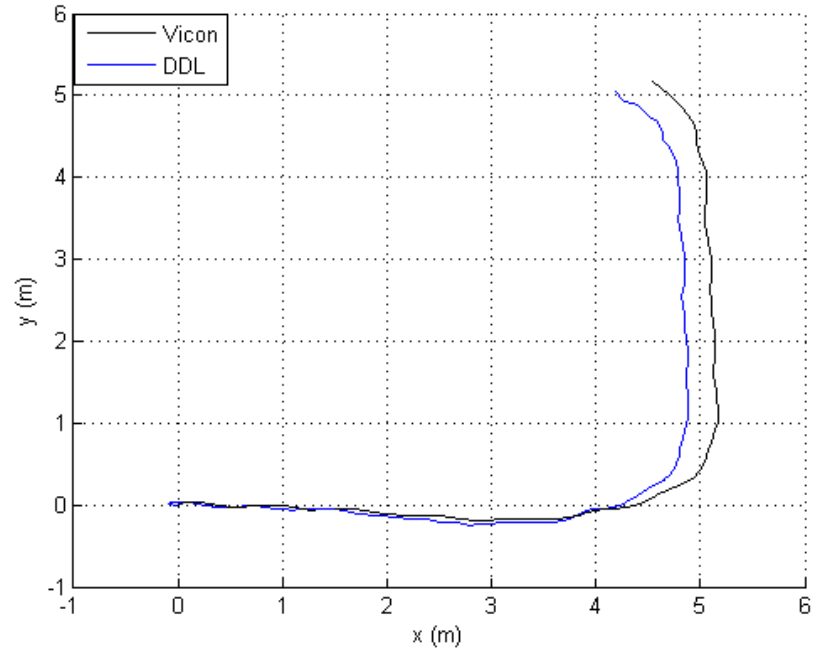


Figure 6.33. *Run 2: 2D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black) with HDR images for indoor lighting conditions*

Figure 6.32, shows the results of our embedded visual odometry based navigation system running with double Dogleg algorithm (in blue) against the reference motion capture trajectory (in black) with standard images (Run 1). On the other hand, Figure 6.33 shows the results of our embedded visual odometry based navigation system running with double Dogleg algorithm (in blue) against the reference motion capture trajectory (in black) with HDR images this time (Run 2).

Table 6.7 *Final Position Relative Error Comparison in Trajectory Generation for indoor lighting conditions.*

Indoor Lighting	2D relative error		3D relative error	
	in m	in %	in m	in %
Run 1 (without HDR)	0.589	5.85	0.909	9.04
Run 2 (with HDR)	**0.387**	**3.81**	**0.457**	**4.51**

Figure 6.34 illustrates the rendering images of the pool with and without HDR mode activated in dark conditions. The HDR image (Figure 6.31 right) gives a much clearer rendering of the scene and while standard image is very dark.

Figure 6.35, shows the results of our embedded visual odometry based navigation system running with double Dogleg algorithm (in blue) against the reference motion capture trajectory (in black) with standard images (Run 3) and Figure 6.36, shows the resulted trajectory using HDR images (Run 4).

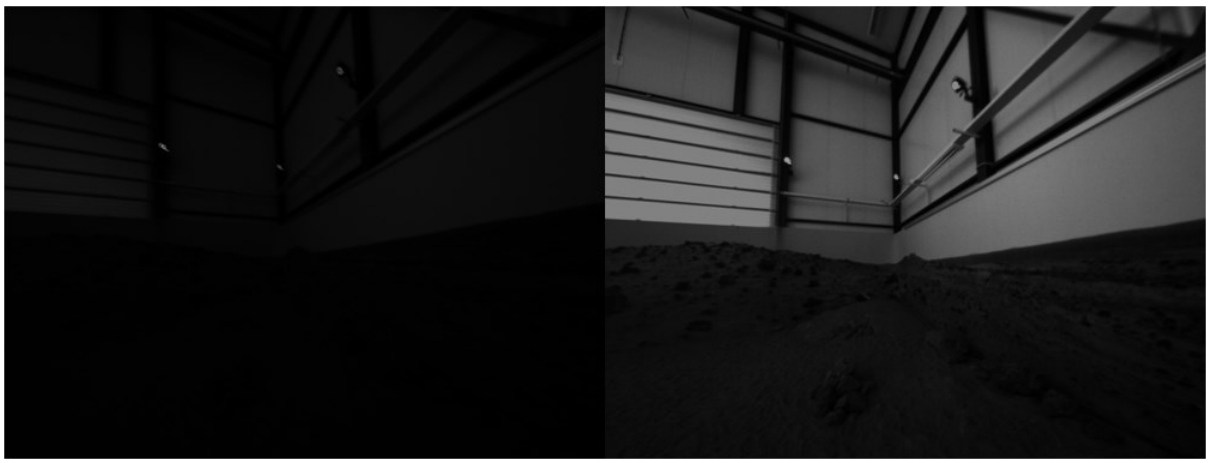


Figure 6.34. *Illustration of the image rendering of the pool without (left) and without (right) HDR mode activated for dark conditions.*

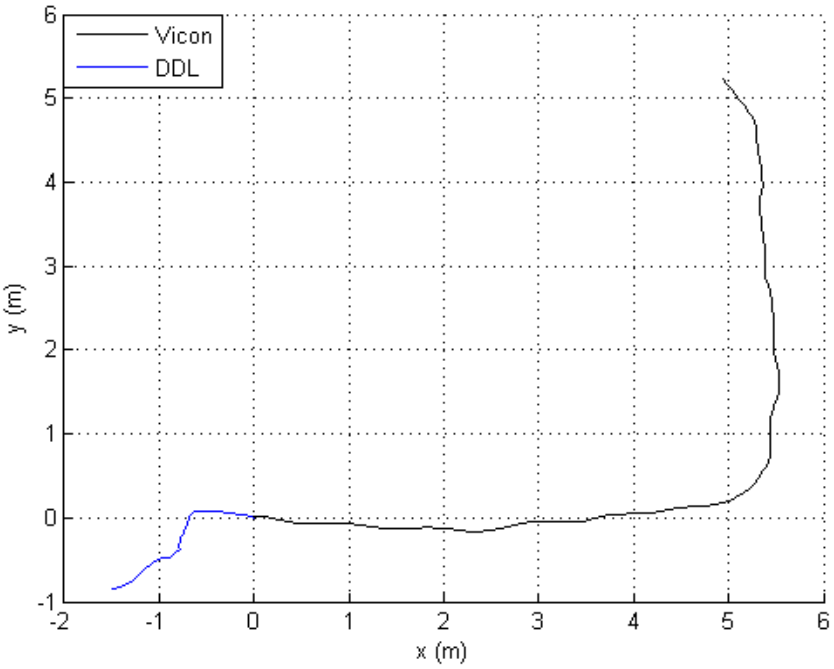


Figure 6.35. *Run 3: 2D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black) with standard images for dark conditions*

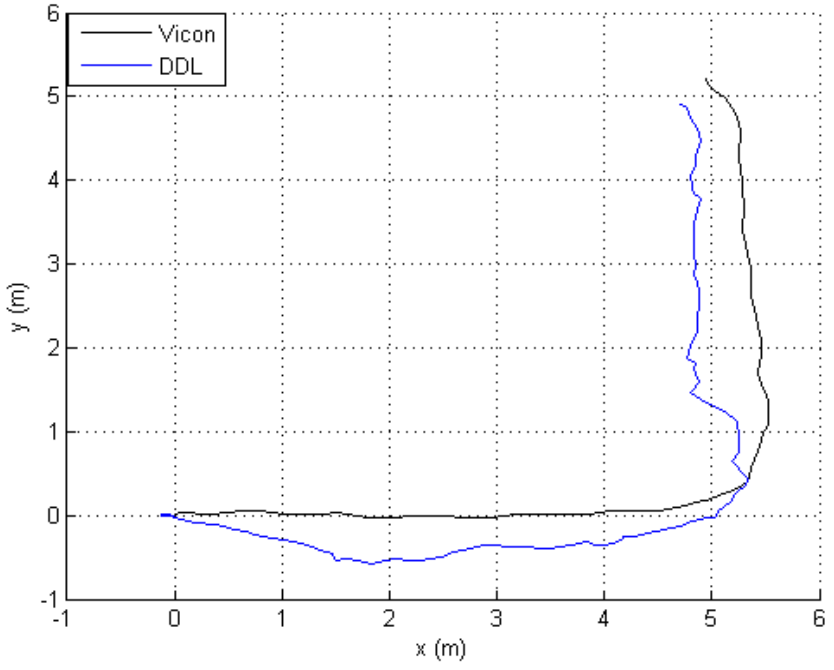


Figure 6.36. *Run 4: 2D plot of the trajectory generated with the navigation sensor using double Dogleg algorithm (blue) compared to Vicon reference (black) with HDR images for dark conditions*

In Figure 6.35 we can clearly observe that the images in dark conditions do not allows our visual odometry algorithm to work as it has been shown in the results presented in this thesis. On the other hand we can see in Figure 6.36 that the generated trajectory by our visual odometry algorithm is remaining relatively close to the reference giving the extreme low illumination conditions. As shown in Table 6.8, the trajectory using HDR images lies at the final position at 38.4 cm while the final positions trajectory using standard images is very far (8.85 m) from the reference.

Table 6.8 *Final Position Relative Error Comparison in Trajectory Generation for dark conditions.*

Dark pool	2D relative error		3D relative error	
	in m	in %	in m	in %
Run 3 (without HDR)	8.856	84.97	8.857	84.97
Run 4 (with HDR)	**0.384**	**3.69**	**1.63**	**15.66**

Figures 6.37 to 6.40 for indoor lighting conditions (Runs 1 and 2) and Figures 6.41 to 6.44 for Dark conditions (Runs 3 and 4) show the relative error in meter as well the relative error over the travelled distance in percentage.

These preliminary results confirm what has been shown in Chapter 3 and highlight the huge potential of using HDR imaging for navigation purpose and more generally in different computer vision applications. For navigation systems, environments presenting extreme illuminations conditions would be less difficult or even now possible to tackle. HDR imaging is certainly on way to answer to extreme illumination problems which are a real concern for a large number of computer vision applications.

Indoor Lighting conditions

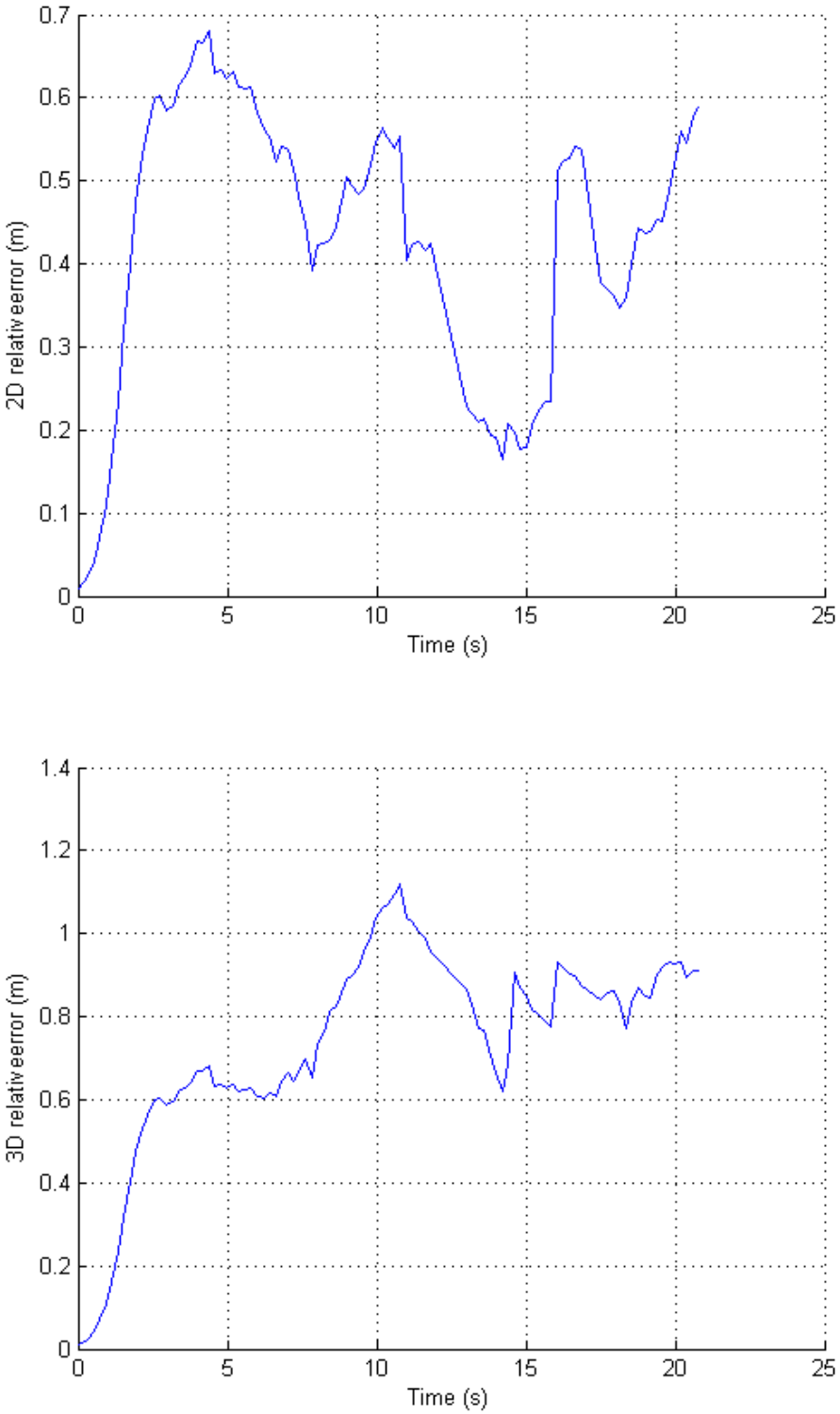


Figure 6.37. *Run 1: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory with standard images for indoor lighting conditions.*

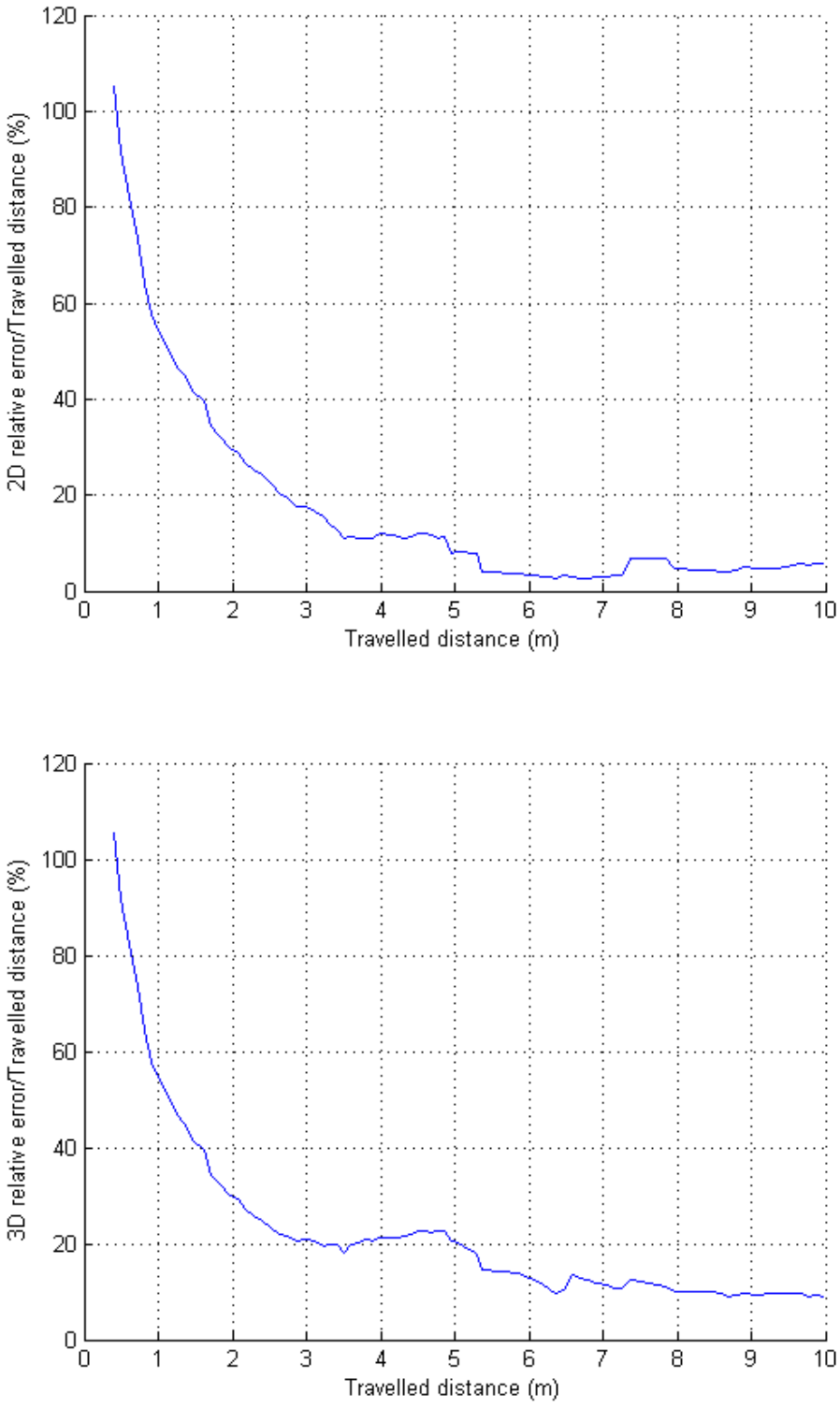


Figure 6.38. *Run 1: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory with standard images for indoor lighting conditions.*

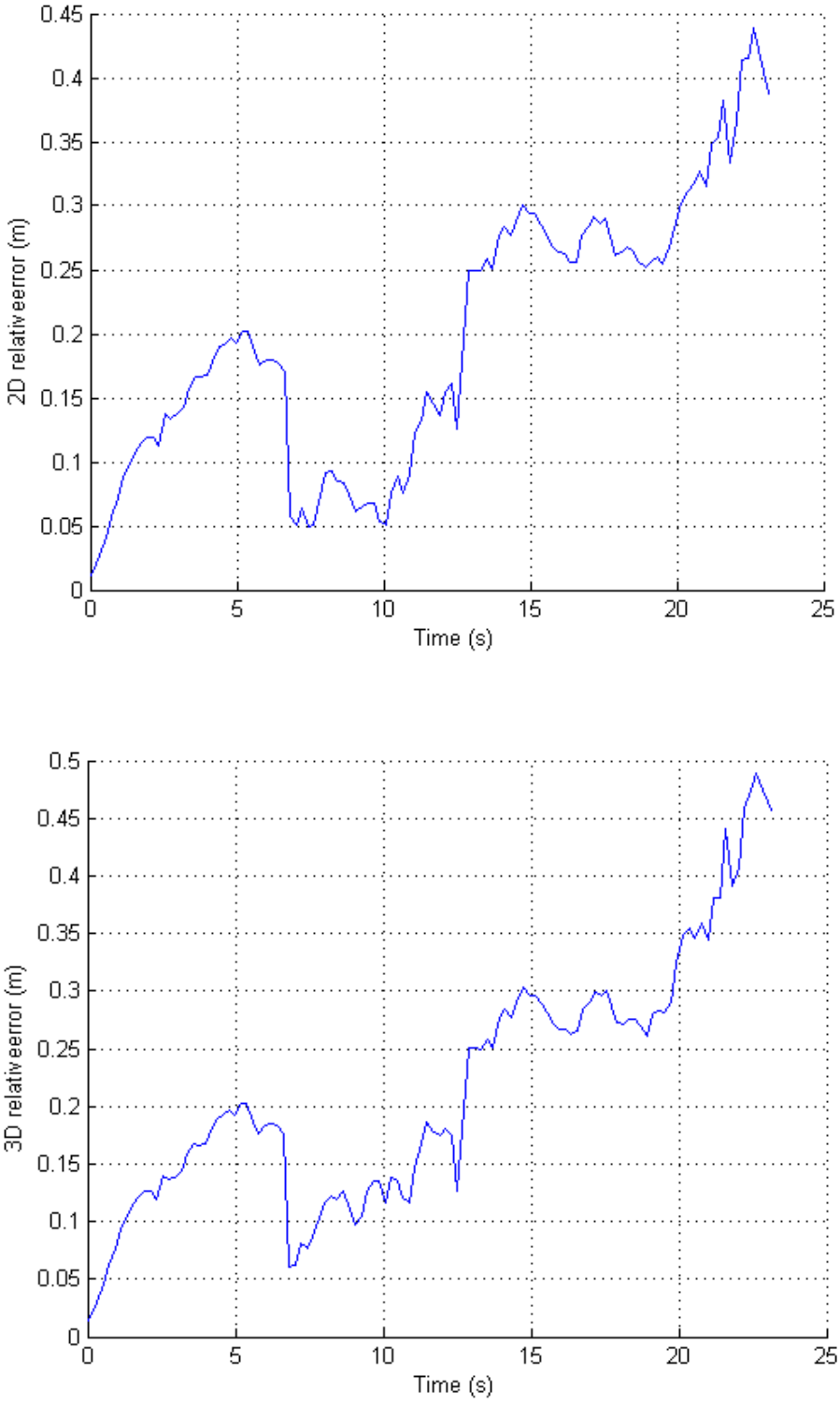


Figure 6.39. *Run 2: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory with HDR images for indoor lighting conditions.*

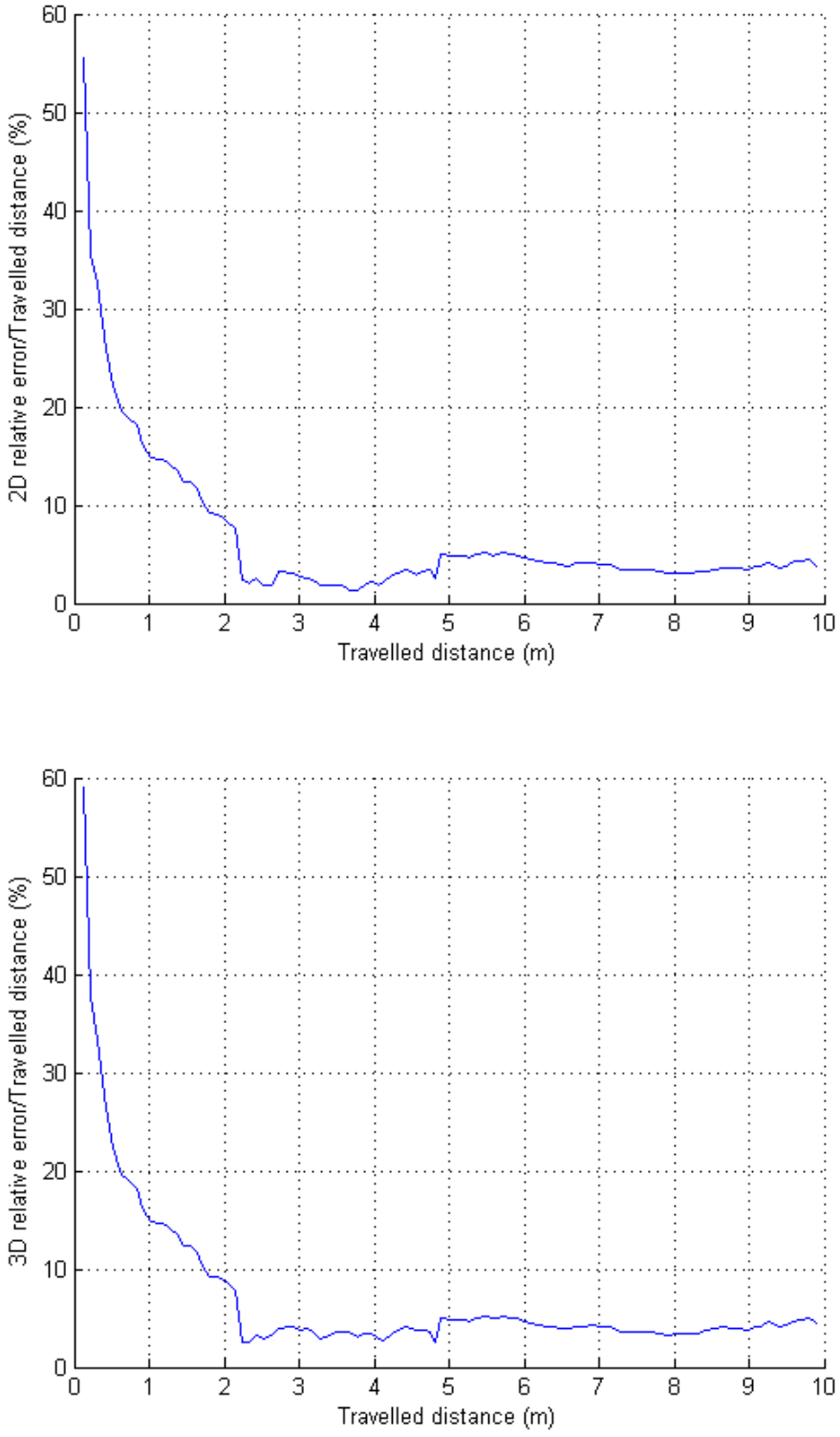


Figure 6.40. *Run 2: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory for with HDR images for indoor lighting conditions.*

Dark conditions

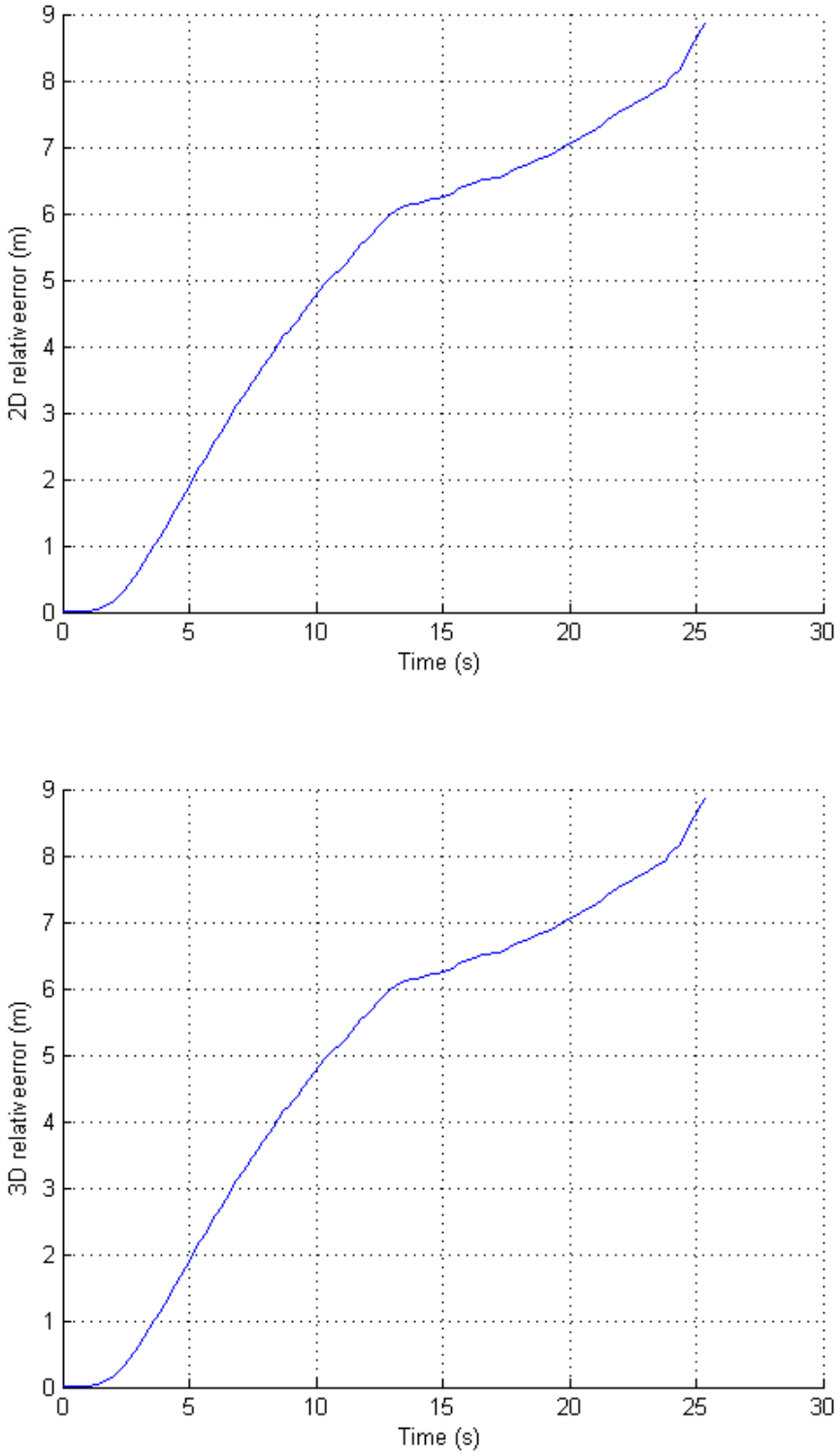


Figure 6.41. *Run 3: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory with standard images for dark conditions.*

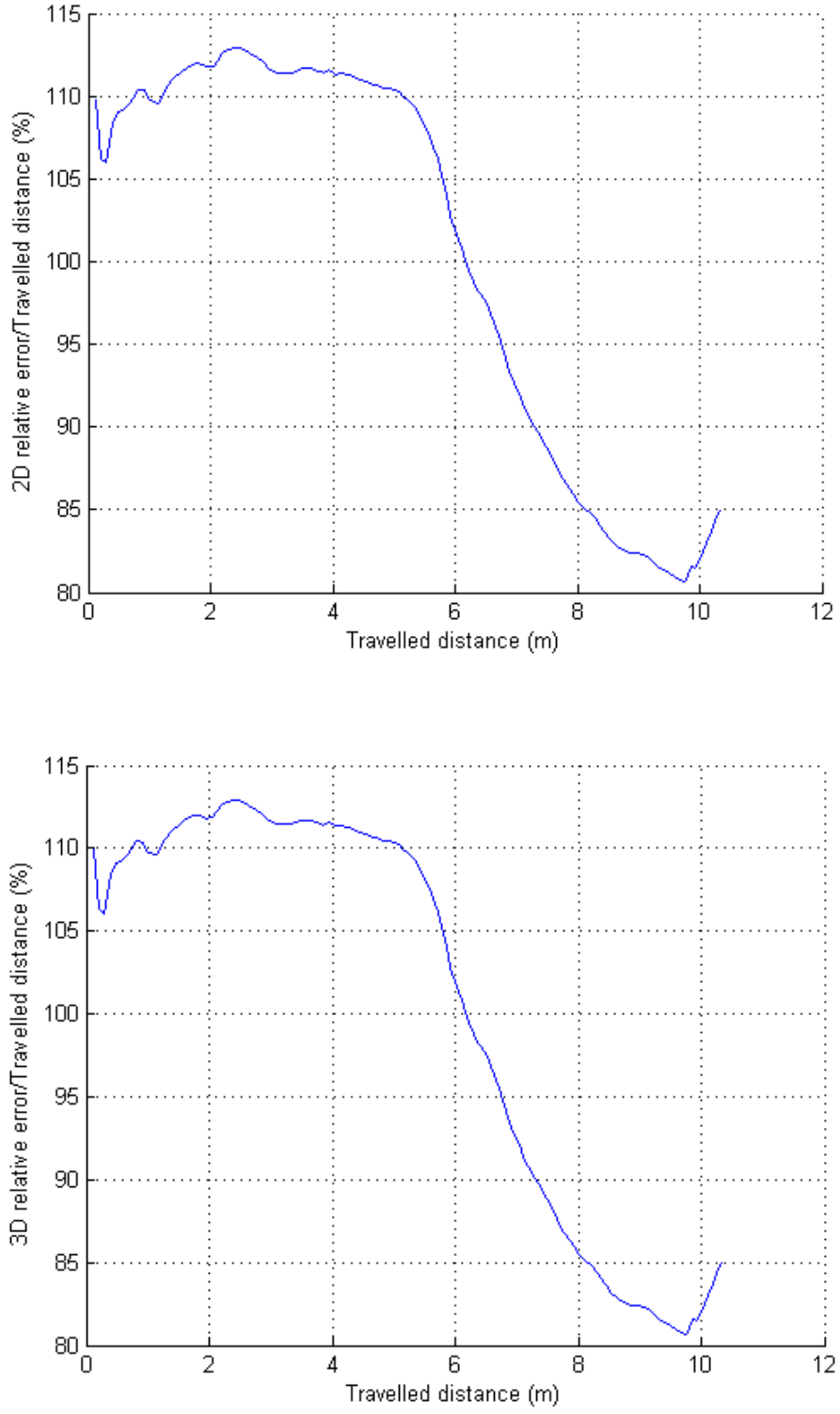


Figure 6.42. *Run 3: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory with standard images for dark conditions.*

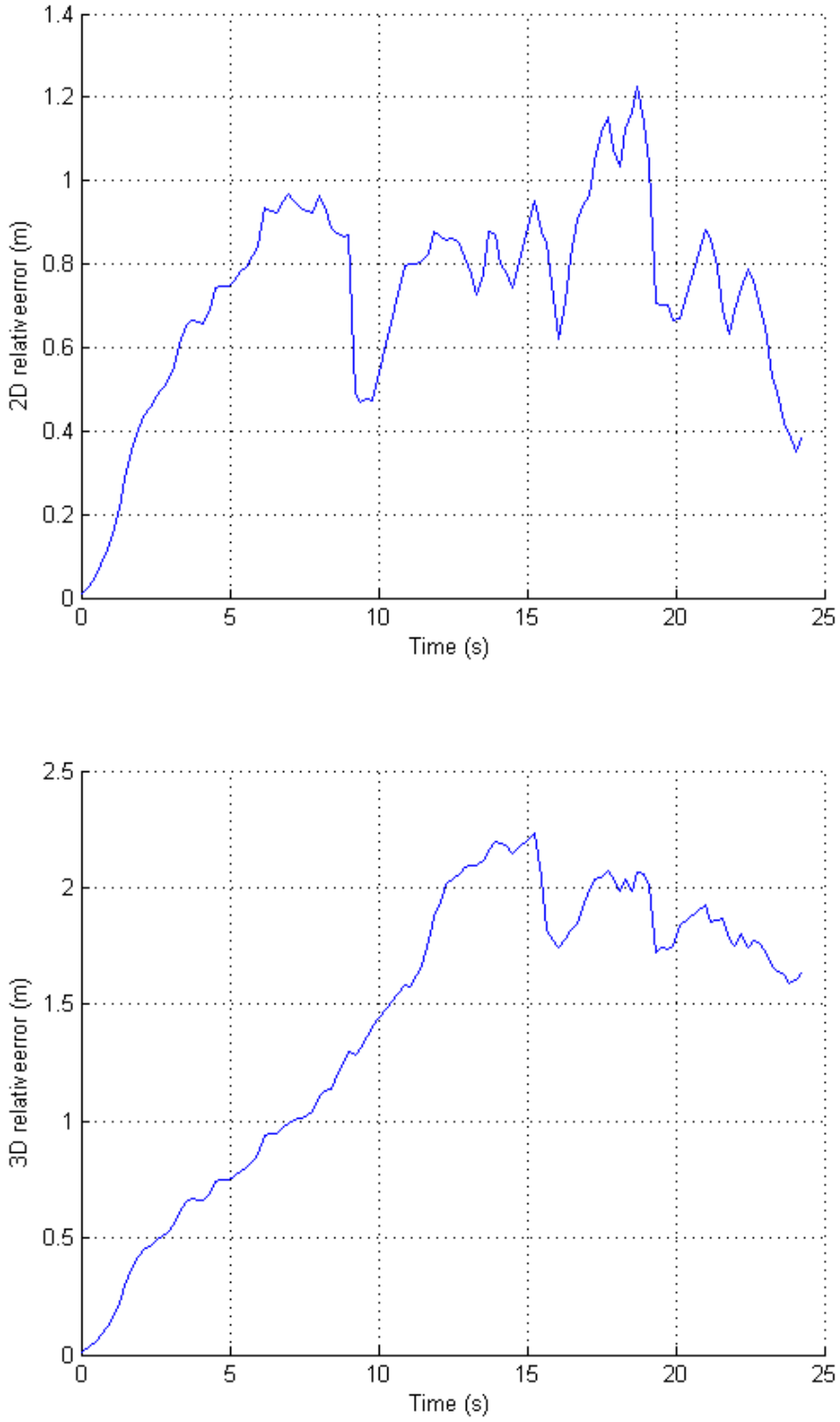


Figure 6.43. *Run 4: Relative squared error2D (top) and 3D (bottom) in meter over the time regarding Vicon reference trajectory with HDR images for dark conditions.*

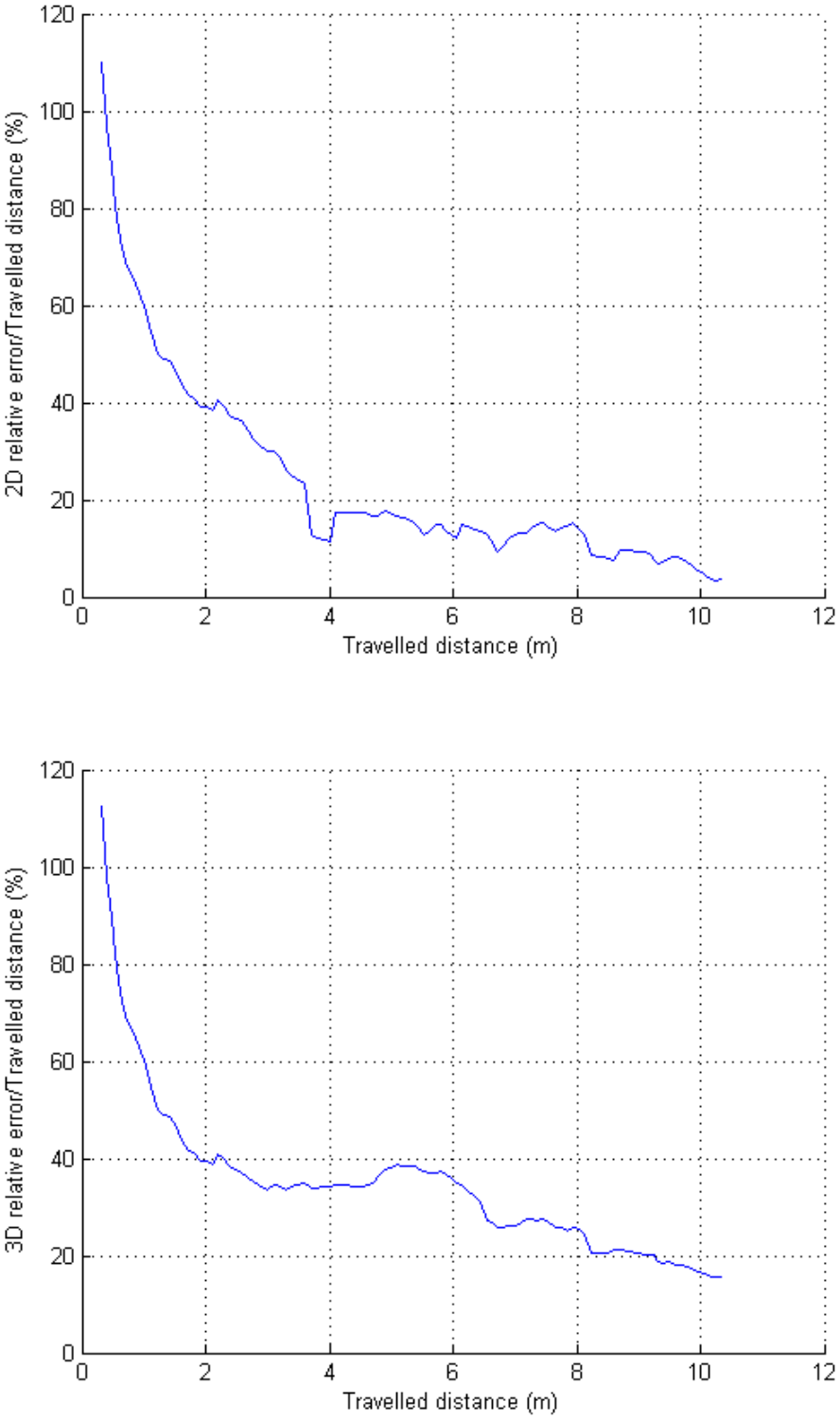


Figure 6.44. *Run 4: Relative squared error2D (top) and 3D (bottom) in percent over the travelled distance regarding Vicon reference trajectory with HDR images for dark conditions.*

6.6 Chapter Conclusion

The Visual/IMU navigation system presented in this chapter is a real-time smart and standalone stereo/IMU ego-motion localisation sensor. We demonstrated through the different sections of this work the great potential of the strategy elaborated and this, from the choice of the components to the choice and the proposed techniques of our visual odometry pipeline.

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

As a result of a balanced combination of hardware and software implementation, we achieved 5fps frame rate processing up to 750 initials features at a resolution of 1280x960. This is the highest reached resolution in real time for visual odometry applications to our knowledge. In addition visual odometry accuracy of our algorithm achieves the state of the art with less than 1% relative error in the estimated trajectories.

Also use of HDR images as input of our visual odometry pipeline showed improvements for normal lighting conditions. For dark conditions it is even more interesting as it provides a reasonably good visual odometry performance where it usually fails completely with standard images.

A great potential is emerging from this work and offers lot of improvement perspectives in software development but also in the setup design. In terms of design, we just started with a compact and comfortable setup without trying to optimise space at the best. This is probably the easiest improvement task to be carried out.

7 Conclusions and Future Work

This thesis has explored robust and efficient techniques to improve the visual odometry pipeline and which resulted into a real time accurate and standalone navigation sensor solution. A significant part of the contributions and efforts put in this thesis were dedicated to the enhancement of feature tracking, the improvement of motion estimation algorithm and the optimisation of the architecture of the visual odometry pipeline. Intersecting these avenues definitely led to porting all of the proposed algorithms to be implemented on a GPU. The benefit here is double. It offers a faster solution while leaving extra computing slots to add further techniques like 3D map reconstruction for instance. The structural design of the navigation sensor can definitely gain in compactness.

In Chapter 3, extreme illumination environments were studied. HDR imaging solution to cope with extreme illumination, showed very promising results. This is very interesting, as HDR imaging extends the possibility of using computer vision algorithms to conditions that are impossible and/or extremely difficult with standard cameras.

In Chapter 4, a 3D correspondence only based motion estimation technique based on quaternion method was examined. Outliers rejection based on 3D geometrical constraints was further investigated in order to provide reasonably good motion estimation results. It also explored the idea of minimising the re-projection error from a spatial feature representation into the 2D image plane in order to reduce errors from 3D approximation. Experimental results, demonstrated the great benefit of using this type of approach

especially with double Dogleg minimisation algorithm. It shows impressive accuracy and stability over dynamic and long range outdoor urban environments.

Chapter 5 presented a novel way to improve feature tracking by combining IMU information using stereoscopy and 3D geometry properties. This enables precise prediction of the features in the forthcoming stereo image pair. Two major benefits have been shown. It reduces wrong tracking over subsequent images. It even enables tracking at low frames rate with challenging image-scaling changes. Results on long range outdoor environment showed clear benefits of using this innovative technique. Visual odometry using IMU assisted feature tracking scheme was even more accurate than GPS/INS in certain conditions.

Chapter 6 materialises the work presented in the previous chapters into a smart standalone visual-IMU navigation system. The full design including the selection of the material, hardware and software implementation were described. GPU porting of the feature detection, stereo matching and feature tracking operations enable us to achieve real-time performance while keeping full size images and a relatively high number of features. Visual odometry performance was experimentally validated through strict criteria and experimental conditions as well as under extreme illumination conditions which validated for visual odometry the conclusions found in Chapter 3 regarding the potential of using HDR imaging in this context.

Future work

As it has been summarised in the previous section, lots of theoretical and experimental works have been carried out during this PhD project. Although there are always ways for improvements, we can humbly say that most of the objectives planned for this thesis have been eventually achieved. Nevertheless, the successful results we ended with would constitute a strong base to exploit in order to begin new projects. As the mountains are made of multitude of stones I wish that my contribution will serve others even just a bit to reach higher levels as previous valuable contributions considerably helped me during my PhD.

Chapter 3: HDR imaging has been studied for feature detection and tracking. Scenario such as low light indoor exploration, outdoor direct sun exposure or route mixing different

illumination conditions would be perfect to test the limit of HDR imaging based navigation.

Chapter 4: Two views local bundle adjustment scheme using double Dogleg minimisation showed impressive accuracy. However, integrating the IMU inter-frame estimation to the motion estimation process could be an interesting way to further improve it. Various approaches can be investigated and a filtering based solution might be one of them.

Chapter 5: IMU assisted feature tracking have been associated with KLT feature tracking. It would be interesting to see how it would behave if it is associated with other tracking approaches such as template matching or descriptors based feature tracking. Additionally, error propagation uncertainty can be also used here as a discarding criterion for inertial predicted features.

Chapter 6: The improvement that can be brought to the designed navigation sensor is porting the remaining part of the visual odometry pipeline to GPU. This mainly includes the adaptive local tracking windows and the motion estimation stages. It is not an easy task, as it requires re-thinking of the architecture and the methodology of serialised processes. However, the potentially reduced time can be invested to add more options to this navigation sensor such as providing dense map reconstruction associated with the explored environments.

Lastly the structure design of the navigation sensor can be rethought in a much compact form and much lighter structure. This can be done simply by changing the form of the structure itself and keeping the same materiel but re-arranging it differently. Of course, there is also the possibility to go for lighter, more compact sensors, board, and battery pack, though it is more a matter of budget.

Bibliography

[1] M. Srinivasan, S. W. Zhang, M. Lehrer, and T. Collett, "Honeybee navigation en route to the goal: visual flight control and odometry," *Journal of Experimental Biology*, pages 237–244, 1996.

[2] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),* pages 652–659, 2004.

[3] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover.," DTIC Document, 1980.

[4] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal of Robotics and Automation,* , pages 239–248, 1987.

[5] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover self localization in planetary-like environments," in *Artificial Intelligence, Robotics and Automation in Space*, pages 433-440, 1999.

[6] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pages 453–458, 2000.

[7] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[8] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, pages 3–20, 2006.

[9] P. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4007–4012, 2004.

[10] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pages 2531–2538,2008.

[11] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, pages 146–157, 1997.

[12] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* , pages 3946–3952, 2008.

[13] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," *Robotics Research*, pages 201–212, 2011.

[14] N. Sünderhauf and P. Protzel, "Stereo odometry–a review of approaches," *Chemnitz University of Technology,* Technical Report, 2007.

[15] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine,*pages 80–92, 2011.

[16] F. Fraundorfer and D. Scaramuzza, "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications," *IEEE Robotics & Automation Magazine*, pages 78–90, 2012.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, pages 91–110, 2004.

[18] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision (ECCV)*, pages 404–417, 2006.

[19] M. Agrawal, K. Konolige, and M. R. Blas, "Censure: Center surround extremas for realtime feature detection and matching," *Computer Vision (ECCV)*, pages 102–115, 2008.

[20] C. G. Harris and J. M. Pike, "3D positional integration from image sequences," *Image and Vision Computing*, pages 87–90, 1988.

[21] J. Shi and C. Tomasi, "Good features to track," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600,1994.

[22] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision (ECCV)*, pages 430–443, 2006.

[23] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[24] J.-Y. Bouguet, "Pyramidal implementation of the affine Lucas kanade feature tracker description of the algorithm," *Intel Corporation*, 2001.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, pages 381–395, 1981.

[26] M. J. Black and A. Rangarajan, "The outlier process: Unifying line processes and robust statistics," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15–22, 1994.

[27] J. A. Ferwerda, "Elements of early vision for computer graphics," *IEEE Computer Graphics and Applications*, pages 22–33, 2001.

[28] M. Ashikhmin, "A tone mapping algorithm for high contrast images," *Eurographics workshop on Rendering*, pages 145–156, 2002.

[29] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski, *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[30] T. Mitsunaga and S. K. Nayar, "Radiometric self calibration," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,1999.

[31] A. Rovid, A. R. Varkonyi-Koczy, T. Hashimoto, S. Balogh, and Y. Shimodaira, "Gradient based synthesized multiple exposure time HDR image," *Ieee Conference on Instrumentation and Measurement Technology (IMTC)*, pages 1–6, 2007.

[32] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *ACM SIGGRAPH 2008 classes*, p. 31, 2008

[33] S. K. Nayar and T. Mitsunaga, "High dynamic range imaging: Spatially varying pixel exposures," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 472–479, 2000.

[34] S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High dynamic range video," *ACM Transactions on Graphics (TOG)*, pages 319–325, 2003.

[35] J. Tumblin and H. Rushmeier, "Tone reproduction for realistic images," *IEEE Computer Graphics and Applications*, pages 42–48, 1993.

[36] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," *ACM Transactions on Graphics (TOG)*, pages 267–276, 2002.

[37] A. A. Goshtasby, "Fusion of multi-exposure images," *Image and Vision Computing*, pages 611–618, 2005.

[38] R. Raskar, A. Ilie, and J. Yu, "Image fusion for context enhancement and video surrealism," *ACM SIGGRAPH 2005 Courses*, p. 4, 2005.

[39] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Transactions on Graphics*, p. 51, 2007.

[40] A. Boschetti, N. Adami, R. Leonardi, and M. Okuda, "High dynamic range image tone mapping based on local histogram equalization," *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1130–1135,2010.

[41] J. Wang, B. Shi, and S. Feng, "Extreme Learning Machine based exposure fusion for displaying HDR scenes," *IEEE International Conference on Signal Processing (ICSP)*, pages 869–872, 2012.

[42] Y. S. Heo, K. M. Lee, and S. U. Lee, "Illumination and camera invariant stereo matching," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[43] N. Sun, H. Mansour, and R. Ward, "HDR image construction from multi-exposed stereo LDR images," *IEEE International Conference on Image Processing (ICIP),* pages 2973–2976, 2010.

[44] A. A. Bell, D. Meyer-Ebrecht, A. Bocking, and T. Aach, "HDR-Microscopy of Cell Specimens: Imaging and Image Analysis," *Asilomar Conference on Signals, Systems and Computers (ACSSC)*, pages 1303–1307, 2007.

[45] Y. Cui, A. Pagani, and D. Stricker, "Robust point matching in HDRI through estimation of illumination distribution," *Pattern Recognition*, pages 226–235, 2011.

[46] B. P\vribyl, A. Chalmers, and P. Zem\vcík, "Feature point detection under extreme lighting conditions," *Conference on Computer Graphics*, pages 143–150, 2012.

[47] L. Chermak, N. Aouf, "A Minimax Solution for Stereo Based Motion Estimation", *IEEE International Conference on Cybernetic Intelligent Systems (CIS)*, pages 112–117, 2012.

[48] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *International journal of computer vision*, pages 335–360, 2011.

[49] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 756–770, 2004.

[50] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62, 1987.

[51] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 580–593, 1997.

[52] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *Pattern Recognition*, pages 236–243, 2003.

[53] W. Chojnacki and M. J. Brooks, "On the consistency of the normalized eight-point algorithm," *Journal of Mathematical Imaging and Vision*, pages 19–27, 2007.

[54] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, pages 629–642, 1987.

[55] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on,Pattern Analysis and Machine Intelligence,* pages 698–700, 1987.

[56] B. K. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *JOSA A*, pages 1127–1135, 1988.

[57] R. M. Haralick, H. Joo, D. Lee, S. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *IEEE Transactions on Systems, Man and Cybernetics*, pages 1426–1446, 1989.

[58] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on pattern analysis and machine intelligence*, pages 376–380, 1991.

[59] T. S. Huang and A. N. Netravali, "Motion and structure from feature correspondences: A review," *Proceedings of the IEEE*, pages 252–268, 1994.

[60] J. C. Chou, "Quaternion kinematic and dynamic differential equations," *IEEE Transactions on Robotics and Automation*, pages 53–64, 1992.

[61] N. Molton and M. Brady, "Practical structure and motion from stereo when motion is unconstrained," *International Journal of Computer Vision*, pages 5–23, 2000.

[62] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Transactions on Robotics and Automation,* pages 804–819, 1989.

[63] Z. Zhang, "Iterative point matching for registration of free-form curves," 1992.

[64] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool, "Flexible 3D acquisition with a monocular camera," *IEEE International Conference on Robotics and Automation*, pages 2771–2776, 1998.

[65] A. Mallet, S. Lacroix, and L. Gallo, "Position estimation in outdoor environments using pixel tracking and stereovision," *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 3519–3524, 2000.

[66] H. Hirschmuller, P. R. Innocent, and J. M. Garibaldi, "Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics," *International Conference on Control, Automation, Robotics and Vision, 2002. (ICARCV),* pages 1099–1104, 2002.

[67] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," *Vision algorithms: theory and practice*, pages 298–372, 2000.

[68] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Technical Report, 2004.

[69] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *IEEE Transactions on Robotics*, pages 1066–1077, 2008.

[70] K. Levenberg, "A method for the solution of certain problems in least squares," *Quarterly of applied mathematics*, pages 164–168, 1944.

[71] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, pages 431–441, 1963.

[72] M. I. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software (TOMS)*, pages 1-30, 2009.

[73] J. E. Dennis Jr and H. H. W. Mei, "Two new unconstrained optimization algorithms which use function and gradient values," *Journal of Optimization Theory and Applications*, pages 453–482, 1979.

[74] M. L. A. Lourakis and A. A. Argyros, "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?," *IEEE International Conference on Computer Vision (ICCV)*, pages 1526–1531, 2005.

[75] M. J. Powell, *A new algorithm for unconstrained optimization*. UKAEA, 1970.

[76] W. Sun and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming*, Springer Science Business Media, 2006.

[77] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International joint conference on Artificial intelligence*, 1981.

[78] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, pages 221–255, 2004.

[79] T. Zinßer, C. Gräßl, and H. Niemann, "Efficient feature tracking for long video sequences," *Pattern Recognition*, pages 326–333, 2004.

[80] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Feature tracking and matching in video using programmable graphics hardware," *Machine Vision and Applications*, pages 207–217, 2011.

[81] H.-K. Lee, K.-W. Choi, D. Kong, and J. Won, "Improved Kanade-Lucas-Tomasi tracker for images with scale changes," *IEEE International Conference on Consumer Electronics (ICCE)*, pages 33–34, 2013.

[82] M. Hwangbo, J.-S. Kim, and T. Kanade, "Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation," *The International Journal of Robotics Research*, pages 1755–1774, 2011.

[83] Y. G. Ryu, H. C. Roh, and M. J. Chung, "Video stabilization for robot eye using IMU-aided feature tracker," *International Conference on Control Automation and Systems (ICCAS)*, pages 1875–1878, 2010.

[84] S. Tanathong and I. Lee, "Translation-Based KLT Tracker Under Severe Camera Rotation Using GPS/INS Data," *IEEE Geoscience and Remote Sensing Letters*, pages 1–1, 2013.

[85] D. Strelow and S. Singh, "Optimal motion estimation from visual and inertial measurements," *IEEE Workshop on Applications of Computer Vision*, pages 314–319, 2002.

[86] M. Agrawal and K. Konolige, "Rough terrain visual odometry," *International Conference on Advanced Robotics (ICAR)*, pages 28–30, 2007.

[87] A. Nemra and N. Aouf, "Robust airborne 3D visual simultaneous localization and mapping with observability and consistency analysis," *Journal of Intelligent and Robotic Systems*, pages 345–376, 2009.

[88] A. B. Chatfield, *Fundamentals of high accuracy inertial navigation*, AIAA, 1997.

[89] A. Makadia and K. Daniilidis, "Correspondenceless ego-motion estimation using an imu," *IEEE International Conference on Robotics and Automation, (ICRA),* pages 3534–3539, 2005.

[90] D. M. Helmick, S. I. Roumeliotis, Y. Cheng, D. S. Clouse, M. Bajracharya, and L. H. Matthies, "Slip-compensated path following for planetary exploration rovers," *Advanced Robotics*, pages 1257–1280, 2006.

[91] Y. Kuroda, A. Sakai, and Y. Tamura, "Six-Degree-of-Freedom Localization using an Unscented Kalman Filter for Planetary Rovers," *Advanced Robotics*, pages 1199–1218, 2010.

[92] T. Zhang, W. Li, K. Kühnlenz, and M. Buss, "Multi-Sensory Motion Estimation and Control of an Autonomous Quadrotor," *Advanced Robotics*, pages 1493–1514, 2011.

[93] L. Chermak, N. Aouf, "A Minimax Solution for Stereo Based Motion Estimation," *IEEE International Conference on Cybernetic Intelligent Systems (CIS),* pages 112–117, 2012.

[94] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, pages 609–631, 2010.

[95] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," IEEE I*ntelligent Vehicles Symposium,* pages 486–492, 2010.

[96] A. I. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3d visual odometry," *IEEE International Conference on Robotics and Automation (ICRA),* pages 40–45, 2007.

[97] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," *IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

[98] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, pages 169–186, 2007.

[99] M. Agrawal and K. Konolige, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS.," *ICPR*, 2006, pages 1063–1068, 2006.

[100] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," *IEEE Intelligent Vehicles Symposium*, pages 963–968, 2011.

[101] S. B. Goldberg and L. Matthies, "Stereo and imu assisted visual odometry on an omap3530 for small robots," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 169–176, 2011.

[102] K. Schmid and H. Hirschmuller, "Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device," *IEEE International Conference on Robotics and Automation (ICRA),* pages 4671–4678, 2013.

[103] http://opencv.org/

[104] http://www.mathworks.co.uk/products/matlab/

[105] http://www.boost.org/

[106] http://www.vision.caltech.edu/bouguetj/calib_doc/

[107] http://www.cvlibs.net/datasets/karlsruhe_sequences/

[108] https://wiki.qut.edu.au/display/cyphy/UQ+St+Lucia.

[109] http://www.hdrsoft.com/images/cs5/comparison.html