Cranfield University

Axel Beauvisage

Robust multispectral image-based localisation solutions for autonomous systems

Cranfield Defence and Security

Centre for Electronic Warfare Information and Cyber

PhD. Thesis

supervisor: **Prof. Nabil Aouf**

November 21, 2019

Cranfield University

Axel Beauvisage

Robust multispectral image-based localisation solutions for autonomous systems

Cranfield Defence and Security

Centre for Electronic Warfare Information and Cyber

PhD. Thesis

supervisor: **Prof. Nabil Aouf**

November 21, 2019

This thesis is submitted for the degree of *Doctor of Philosophy*

I would like to dedicate this thesis
to my grandparents

Abstract

With the recent increase of interest in multispectral imaging, new image-based localisation solutions have emerged. However, its application to visual odometry remains overlooked. Most localisation techniques are still being developed with visible cameras only, because the portability they can offer and the wide variety of cameras available. Yet, other modalities have great potentials for navigation purposes. Infrared imaging for example, provides different information about the scene and is already used to enhance visible images. This is especially the case of far-infrared cameras which can produce images at night and see hot objects like other cars, animals or pedestrians. Therefore, the aim of this thesis is to tackle the lack of research in multispectral localisation and to explore new ways of performing visual odometry accurately with visible and thermal images.

First, a new calibration pattern made of LED lights is presented in Chapter 3. Emitting both visible and thermal radiations, it can easily be seen by infrared and visible cameras. Due to its peculiar shape, the whole pattern can be moved around the cameras and automatically identified in the different images recorded. Monocular and stereo calibration are then performed to precisely estimate the camera parameters.

Then, a multispectral monocular visual odometry algorithm is proposed in Chapter 4. This generic technique is able to operate in infrared and visible modalities, regardless of the nature of the images. Incoming images are processed at a high frame rate based on a 2D-to-2D unscaled motion estimation method. However, specific keyframes are carefully selected to avoid degenerate cases and a bundle adjustment optimisation is performed on a sliding window to refine the initial estimation. The advantage of visible-thermal odometry is shown on a scenario with extreme illumination conditions, where the limitation of each modality is reached.

The simultaneous combination of visible and thermal images for visual odometry is also explored. In Chapter 5, two feature matching techniques are presented and tested in a multispectral stereo visual odometry framework. One method matches features between stereo pairs independently while the other estimates unscaled motion first, before matching the features altogether. Even though these techniques require more processing power to overcome the dissimilarities between

multimodal images, they have the benefit of estimating scaled transformations.

Finally, the camera pose estimates obtained with multispectral stereo odometry are fused with inertial data to create a robustified localisation solution which is detailed in Chapter 6. The full state of the system is estimated, including position, velocity, orientation and IMU biases. It is shown that multispectral visual odometry can correct drifting IMU measurements effectively. Furthermore, it is demonstrated that such multi-sensors setups can be beneficial in challenging situations where features cannot be extracted or tracked. In that case, inertial data can be integrated to provide a state estimate while visual odometry cannot.

Keywords

Multispectral imaging; thermal; infrared; visual odometry; motion estimation; stereovision; optimisation; data fusion; Kalman filter.

Acknowledgements

I would like to take the opportunity to thank my supervisor, Prof. Nabil Aouf, for giving me the chance to undertake this PhD and for his guidance. I am especially grateful for his full support until the end, my research would not have been possible without his aid.

I would like to thank Alessio Balleri for accepting me as a student and for his supervision. I am grateful for his advice and support towards the end of my PhD.

I wish to express my sincere gratitude to Kenan Ahiska and Antonio Scannapieco for stepping in and helping me when I needed it. Thank you for your supervision, your useful feedback and your cooperation. Without them, this thesis would not have been what it is now. Thank you for helping me to raise the level of this thesis.

I would like to thank the staff and students of the unmanned autonomous systems lab for all these enlightening discussions. With a special mention to Alejandro, Duarte, Ray and Shahmi for their everyday support. To Alejandro Dena, thank you for your help and guidance on electronic matters. I would also like to mention, Hugo and Carole who started and struggled with me. thank you for sharing your experience and for those moments spent together.

I would like to thank Leon Kocjancic for helping me to set up other challenges and escape the everyday life of a research student. To my squash and basketball buddies, thank you for helping me to relax.

To Maud Sananikone, thank you for your love and moral support. Thank you for your never ending encouragements in good and bad times, and sorry for having to endure this PhD with me.

Last but not least, I would like to thank my parents for their support and for never stopping to believe in me. I also wish to express my gratitude to all the other members of my family who helped me to achieve my goal.

Contents

List of Figures

List of Tables

List of Abbreviations

VO: Visual Odometry

ME: Motion Estimation

SLAM: Simultaneous Localisation And Mapping

V-SLAM: Visual-Simultaneous Localisation And Mapping

VINS: Visual-Inertial Navigation System

VIO: Visual-Inertial Odometry

IMU: Inertial Measurement Unit

MEMS: Micro-Electro-Mechanical System

GNSS: Global Navigation Satellite System

GPS: Global Positioning System

LWIR: Long-Wave Infrared

SWIR: Short-Wave Infrared

PnP: Perspective N Points

BA: Bundle Adjustment

WBA: Windowed Bundle Adjustment

RANSAC RANdom SAmple Consensus

SSE: Sum of Squared Errors

DoF: Degrees of Freedom

FOV: Field Of View

$\mathbf{SO}(3)$**:** Special orthogonal group

$\mathfrak{so}(3)$**:** Special orthogonal Lie algebra

$\mathbf{SE}(3)$**:** Special Euclidean group

$\mathfrak{se}(3)$**:** Special Euclidean Lie algebra

$\mathbb{R}^N$**:** N-dimensional vector of real numbers

$\mathbb{I}^N$**:** N-dimensional vector of imaginary numbers

$\mathbb{H}$**:** Quaternion group

1 | Introduction

Whether we can see it or not, Robotics and Autonomous Systems (ARS) are taking a greater part in our everyday lives, especially with the recent emergence of Artificial Intelligence (AI), which facilitates greatly the automation of complex and tedious tasks. As we start speaking of a robotic revolution, the ability of an agent to localise itself and to navigate safely in its environment is becoming a major step towards mobile robots complete autonomy.

Autonomous navigation is a rapidly growing field which gathers many research aspects related to the mobility of robots and the way they perceive their surrounding environments. It has attracted a lot of attention in recent years due to its diverse domains of application. This technology could be used for all sorts of vehicles. For example, autonomous cars could transport people or goods from one place to another without the need for a driver. With drones and ground robots, inspections and various other logistic operations could become autonomous. For wearables such as helmets or glasses, some algorithms are already used to provide the user with an augmented reality experience. These are only a few examples, many more applications could be found. Indeed, autonomous navigation is a vast and complex concept including localisation, path planning, control theory and obstacle avoidance. Path planning focuses on finding the optimal way to reach a destination, taking into account the current environment as it is perceived. Control is necessary to guarantee that the commands sent to the robot are executed

properly and that it behaves as intended. In a dynamic environment, obstacle avoidance techniques are used to detect, evaluate and avert potential threats. All these algorithms rely on the vehicle localisation system. To plan a new mission, correct the trajectory if the vehicle drifts from the desired path or estimate obstacles behaviour, it is necessary to know precisely the position and orientation of the vehicle. Therefore, a good localisation solution is required to make sure that the vehicle does not end up far from its intended destination and that it is able to adapt to a dynamic environment.

The localisation challenge consists in estimating the position and the orientation of a vehicle over time. It can be global when defined with respect to a general coordinate system. For example, *Global Navigation Satellite System* (GNSS) coordinates correspond to a specific position around the Earth. Electromagnetic compasses can be used to estimate the heading of the vehicle with respect to the north pole. Navigation can however be relative, when defined with respect to a local coordinate system (e.g. body frame or initial frame). For instance, inertial sensors provide acceleration and angular velocity measurements expressed in the sensor coordinate frame. They can be integrated over a certain time interval to estimate the evolution of the robot's position, velocity and orientation in that interval. Thus, the current state of the robot is computed based on the previous one. However, GNSS receivers and *Inertial Measurement Units* (IMUs) are not the only sensors that could be used for navigation. In fact, a broad range of sensors is available [1].

1.1 Visual localisation

In recent years, image-based localisation has been studied substantially and has led to the creation of a new research field: *Visual Odometry* (VO) [2]. It consists

in estimating the relative motion between camera frames based on visual features and the camera(s) geometric properties. VO is a type of relative estimation, where local transformations are accumulated to generate an estimate of the current pose of the vehicle. Additionally, cameras can be used to build a consistent map of the environment along with the pose estimate. In that case, we speak of *Visual-Simultaneous Localisation And Mapping* (V-SLAM) [3]. With V-SLAM it is then possible to estimate the global position and orientation of the vehicle in that map [4]. Hence, V-SLAM algorithms can benefit from additional optimisation techniques such as loop closure, when the vehicle returns to a previously visited location [5]. The path can be corrected and a more precise localisation is obtained. This, on the other hand, makes V-SLAM computationally heavier and leads to some specific issues such as map/feature data association [6].

Within VO and V-SLAM frameworks, several approaches have been investigated over the years to recover motion from visual cues. For example, the 6 DoF transformations between consecutive frames can be computed by solving a system of homogeneous linear equations using the Direct Linear Transformation algorithm [7, 8], or through the estimation of the *essential* matrix [9]. However, more complex solutions such as recursive numerical optimisation [10, 11, 12, 13] or filtering [14, 15, 16] have become more popular since they can take the noise and errors present in the visual data into account, making them more robust to real world datasets.

Furthermore, the architecture of a visual localisation algorithm depends greatly on the amount of cameras it is using. The main types of visual navigation setups are *monocular*, where a single camera is employed, and *stereo*, where two cameras are utilised [3]. Stereo cameras have the advantage of providing depth information through the use of stereovision, allowing for precise estimation of scaled trajectories. However, such systems are usually affected by the quality of the camera

calibration process and the feature matching between stereo images. On the other hand, monocular setups are more versatile and portable since they only depend on one camera, but they are known to suffer from scale ambiguity.

1.2 Multispectral imaging

1.2.1 Overview

Images are produced by detecting electromagnetic radiations (light) absorbed by a sensor. Most cameras sense radiations emitted in the visible domain, which is the same portion of the spectrum seen by the human eye. The development of cameras was probably initiated by the desire to create an artificial eye and was largely inspired by the human body.

However, visible light is not the only type of electromagnetic radiations that can be perceived by optical sensors. As illustrated in Figure 1.1, the electromagnetic spectrum comprises various types of radiation ranging from radio waves with large wavelengths (up to 10 Km) to gamma waves with particularly small wavelengths (up to 1 pm). Hence, visible light only corresponds to a small part of the full spectrum. While any object whose temperature lies above the absolute zero emits electromagnetic radiations, not all the objects emit the same type and amount of radiations. According to Planck's law, black bodies (perfect emitters) produce a radiation peak at a certain wavelength which depends on their temperature. This is why objects at room temperature ($\sim$ 300 K) emit mostly thermal radiations, also called Long Wave InfraRed (LWIR), which are not seen by human eyes. However, when heated, they start glowing as the peak shifts towards the visible domain. This is what happens to a light bulb when it is heated by an electric current. Another example is the sun whose temperature amounts to approximately 6000 K. Even though it generates all types of radiations, its peak emission is located in the visible

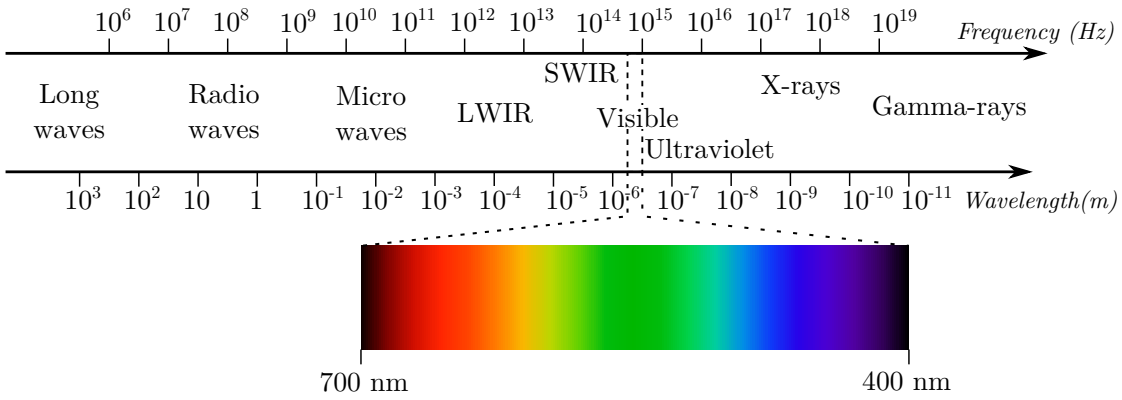


Figure 1.1: The electromagnetic spectrum. The type of light depends on its wavelength (or frequency).

domain. The sun is a natural source of light but what makes infrared radiations particularly interesting is the fact that they are emitted at low temperature and do not require heating or additional light source to be seen by thermal cameras.

Besides the temperature, other factors can affect the appearance of an object. First, the spectral emissivity is a measure between 0 and 1 which describes the resemblance of an object to a black body emitter for a specific wavelength. If the index is close to 1, it emits as much radiations as possible. If close to 0, the emissions are negligible. The emissivity of an object corresponds to the average spectral emissivity for a certain range of wavelengths. Secondly, the reflective property of an object, corresponding to its ability to reflect incoming rays of light, also impacts on the aspect of the object. Finally, the transmission of a material affects the signal travelling through it. Transmission T depends on the absorption and scattering factors of the environment and the distance travelled. They are linked by the following formula: $T = e^{(\alpha+\beta)D}$, where α and β are the absorption and scattering factors, respectively, and D the distance to the emitter. For example, the atmosphere absorbs and scatters certain frequencies due to the presence of CO_2 and H_2O molecules. Emissivity, reflection and transmission always add up to 1. This means that for solids where transmission is null, emissivity and reflection

(a) Visible domain. (b) Thermal domain.

Figure 1.2: Difference between visible and IR images.

are inversely proportional. Low emissivity materials such as polished metals tend to be highly reflective and vice-versa.

These different material properties lead to distinct behaviours depending on the part of the electromagnetic spectrum that is observed. Various types of images, containing different information, can thus be captured. Regarding the scope of this thesis, the following works focus on the visible and LWIR modalities. An example of each modality enlightening their differences can be found in Figure 1.2. As the figure shows, it is possible to recognise text and drawings on a road sign in the visible domain because some wavelengths are absorbed and some others are reflected. However, it is not possible to distinguish between the paint and the background in the IR domain because they both have approximately the same temperature of emissivity. Moreover, it is possible to see through glass with visible cameras because visible light is transmitted, but not with thermal cameras as IR light is reflected. It is also important to note that visible light is mostly reflected. At the exception of a few artificial sources such as headlights or traffic lights, the main source of light is coming from the sun in outdoor environments. This is what makes night navigation so difficult with visible cameras. On the contrary, LWIR radiations are mostly emitted by the scene.

The contrasting nature of the information available makes multispectral setups

powerful. Combining visual information from different part of the electromagnetic spectrum is called multispectral imaging and as it offers several advantages, it also presents few drawbacks. Visible image-based localisation suffers from illumination changes or lack of powerful light source while thermal imaging does not. On the other hand, it is possible to distinguish colours on visible images, which is not the case with IR. Yet, while one modality can be used to enhance the other, it is also harder to process the two information simultaneously. Indeed, no linear relationship can be found between visible and thermal images. It is therefore impossible to predict the pixel intensities in one image based on the other one. This makes all pixel intensity based techniques inadequate.

1.2.2 Applications

Computer vision has many potential applications and limiting its use to the visible spectral band would be a mistake. Looking at other wavelengths can enhance the visual data and provide more valuable information in some cases. For medical applications, multispectral and hyperspectral images are widely used to facilitate the detection of certain diseases [17]. As suggested in [18], various parts of the infrared domain have been explored by the U.S. army for defence applications. They have been utilised to track targets, detect ballistic missiles or reveal land mines.

From a commercial perspective, some multispectral visual systems are already available like ASTROHN Technology's visible-thermal cameras which are used for surveillance. This new market is also exploding in agriculture, to monitor crops based on a multispectral analysis of the soils and plants. A wide variety of sensors can now be purchased and easily mounted on small commercial drones. Amongst them, the Parrot's sequoia, Tetracom's ADC Lite or Sony's MSZ-2001G cameras can be noted.

However, multispectral imaging is relatively new in autonomous navigation and has rarely been used for localisation purposes [19]. A few works have investigated the use of thermal cameras for visual odometry [20, 21] and a few others have made use of truly multispectral stereo images [19, 22, 23, 24]. Mouats was the first one to exploit this idea in order to perform VO from cars [19], following his work on multimodal stereo matching [25]. The KAIST laboratory created a public car dataset for day/night conditions but only used it for pedestrian detection [26]; no work was performed for localisation. In [22], visible and thermal images were fused to create an enhanced version which was then employed in a monocular localisation algorithm. Later, the use of deep learning was investigated in [27] to perform stereo matching between modalities. Following initial works [25, 19], the Cranfield University UASL laboratory has investigated further the challenges of multispectral image registration [28, 29] and multispectral stereo odometry [23, 24].

1.3 Objectives and motivation

The main objectives of this thesis are defined as follows:

- Develop new ways of combining thermal and visible images to perform visual odometry.
- Tackle the lack of similarity between such modalities and the lack of multispectral localisation solutions in the literature.
- Overcome the limitations of state-of-the-art image processing and localisation techniques for extreme illumination scenarios.
- Demonstrate the benefits of multispectral setups on real data, with real driving conditions.

The use of cameras was motivated by the vast potential they offer and the attractiveness of VO/V-SLAM algorithms. Indeed, cameras are becoming smaller, lighter and cheaper every day. It has turned into a truly versatile sensor and can easily be embedded on any kind of unmanned systems, from cars to UAVs. Moreover, visual odometry offers a greater precision (few cm) compared to standard GNSS (few m), especially in urban environments where tall buildings interfere with the GNSS signal. Furthermore, satellites are not always reachable. This is the case inside buildings or in sensitive areas where communications are jammed. Besides being an interesting alternative to GNSS, cameras can also be an appealing option to replace bulky and expensive sensors like radar or LiDar which are less convenient to implant on small robots due to their size, weight and power consumption.

However, VO possesses a major drawback: it drifts over time. Because relative transformations are estimated, the errors produced, as small as they are, can be accumulated over time and become significant after a certain duration. Even if some techniques have been developed to correct and reduce this drift, it cannot be completely removed. Nevertheless, image-based localisation drifts less than inertial based localisation. To compensate for this known issue, VO and inertial data are usually combined with other sensors to create robust localisation solutions able to work for extended periods of time. For that reason and because it also has the advantage of being small, cheap and is already present in most autonomous systems nowadays, an IMU was added to the setup.

Despite the fact that other types of camera exist and provide information of a different kind, most of the research found in the literature focus on the visible modality as it corresponds to the most common type of cameras. However, with other visual sensors becoming widely available and being used for various applications, there is now a clear lack of resources and algorithms to process visual data

acquired by multispectral systems. This is why this thesis aims at providing new solutions to perform multispectral visual odometry.

In order to take advantage of the additional information provided by multispectral setups and overcome the lack of multispectral odometry solutions in the literature, the study of visible and thermal images was preferred. The choice of these modalities was motivated by their dissimilarity and the challenge they represent. Indeed, combining such images encourages the development of new image processing methods which do not rely on pixel intensities. Such techniques would need to be more generic and it would be easier to adapt them to other spectral bands. Finally, visible-thermal imaging offers several advantages as the range of information available is extended. For example, the infrared domain is more appropriate for pedestrian detection or to distinguish the different elements of the scene at night since it is sensitive to heat generated by human bodies and other objects (see Figure 1.3) [26]. On the other hand, the visible modality is more helpful to detect road signs and ground features. Therefore, such multispectral setups are interesting because they can not only be used for localisation purposes, but for other aspects of autonomous navigation such as obstacle detection/avoidance, target recognition, etc...

Furthermore, a stereo setup was preferred because it allows to compute the depth of the scene quickly and efficiently, but more importantly, it allows to estimate scaled trajectories. As it will be seen later, it does not necessarily have to be used in a stereo manner, especially since one of the objectives is to look at extreme conditions. The images produced might not always be suitable for motion estimation and it might be better to only work with one camera.

Finally, a VO approach was preferred over SLAM because the main objective of this thesis is to focus on the multimodality aspect of the localisation solution. Hence, it has been deemed not necessary to build and update a map of the en-

(a) Visible domain. (b) Thermal domain.

Figure 1.3: Visible and thermal images at night. The visible image is dark because no significant source of light is available. Street lights can be spotted but they are not bright enough to light the scene. On the contrary, the thermal camera is able to interpret the infrared radiations emitted by trees and other elements of the scene.

vironment. This way, more resources could be used to process the multispectral images.

1.4 Tools and Softwares

1.4.1 Experimental setup

To fulfil the goals set by this thesis, the following multispectral setup is used. It is made of several sensors mounted on top of a car, including:

- a Tau2 thermal camera (Flir)
- a BlueFOX visible camera (Matrix Vision)
- an MTi-G-710 IMU (Xsens)

Both cameras are attached to a rigid structure, facing forward, and are separated by a baseline of around 50 *cm*. The baseline was selected in accordance to the distance of the scene and the amount of overlap between stereo images. It

was assumed that in a navigation context, most features would be located between a few meters and a few hundred meters from the cameras. Hence, such baseline provides significant disparity between features while maintaining enough overlap between stereo images.

The thermal camera is uncooled and provides a high image acquisition frame rate. Its resolution is relatively high for a thermal camera of such a size (see Table 1.1). The small size and the light weight of the *Tau2* camera make it convenient to embed on vehicles such as cars or UAVs. The BlueFOX camera is even smaller, lighter and possesses a higher acquisition rate. To match the one channel characteristic of the thermal camera, a grayscale visible camera was selected which acquires 8-bit images. An High Dynamic Range (HDR) mode is available, for this camera, to ensure a good illumination of the whole image and to guarantee pixel consistency between consecutive frames. This is an important feature to have because, as it will be seen later, the image processing techniques used in this thesis are based on optical flow, which assumes pixel consistency between the images. More details regarding the camera characteristics can be found in Table 1.1. Unlike common stereo rigs which contain identical cameras, multispectral setups use camera with different attributes. Hence, they must be chosen to have the most similar properties as possible. Because thermal cameras possess sensors with a larger pixel size than visible cameras, due to the longer wavelength of the radiations they are measuring, the lens of each device needs to be selected carefully so that both field-of-view are similar. To acquire images at the exact same time, an external hardware trigger is employed. It generates a short pulse sent to both cameras at a 30 Hz frequency. This way, the thermal camera is used at its full capability.

The Mti-G is an Micro-Electro-Mechanical Sytem (MEMS) inertial measurement unit. It is equipped with a GPS receiver to provide a full INS/GNSS local-

Table 1.1: Camera characteristics.

Camera	Tau 2	BlueFox
Max. framerate	30 fps	90 fps
Resolution	640×512 px	752×480 px
Spectral band	$7.5 - 13.5$ μm	$7 - 14$ nm
Pixel size	17 μm	6 μm
Focal length	7 mm	9 mm
Dimensions	$40 \times 40 \times 15$ mm	$45 \times 45 \times 30$ mm

isation solution. It is important to note that this information is not used in the algorithms developed as the aim of this work is to provide a solution which does not rely on GPS. Instead, the sensor is employed to record the precise trajectory of the vehicle which is considered to be the ground truth in all experiments. Additionally, raw acceleration and angular velocity measurements are recorded. They are fused in the visual-inertial framework presented in Chapter 6.

It is important to note that even though the INS/GNSS trajectory provided by the MTi-G is used to evaluate the performance of the VO algorithms presented in this thesis, it remains a position estimate based on noisy inertial and GNSS data. By fusing these information, the system is able to provide precise localisation with a standard deviation of about 1 m at a high rate (300 Hz). Since the length of the experimental trajectories is much bigger than a few meters, this precision is acceptable. Nevertheless, this level of noise present in the ground truth must be kept in mind when analysing the results obtained in the following chapters.

For clarity, a picture of the full setup is shown in Figure 1.4. It was used to acquire several datasets by driving the car around the Defence Academy of the UK at Shrivenham, in a semi-urban environment. These datasets contain trajectories of a few hundred meters. They represent real driving conditions, with dynamic elements such as pedestrians or other cars.

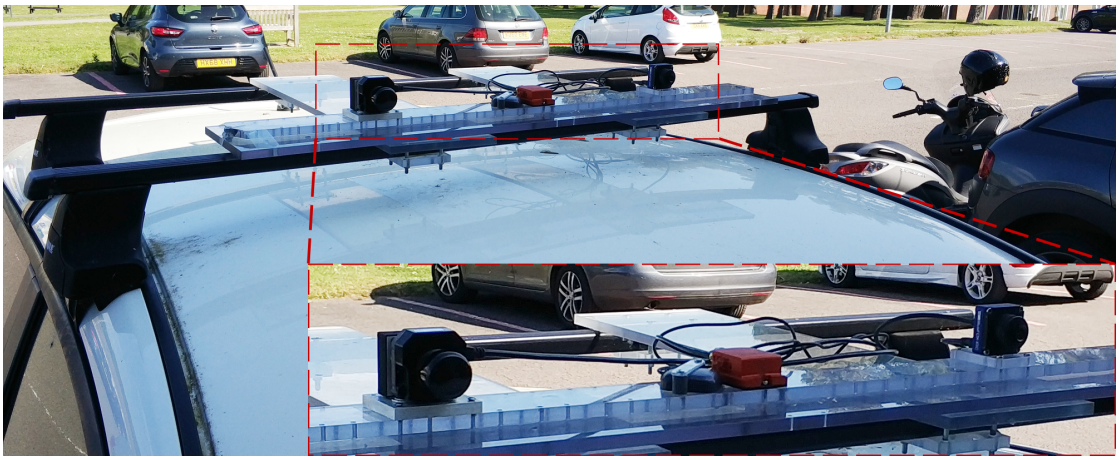


Figure 1.4: Experimental setup embedded on top of a Nissan Leaf car. The thermal camera can be seen on the left and the visible camera on the right. The IMU is located at the centre.

1.4.2 Softwares

All the algorithms presented throughout this thesis were implemented and tested in C++. This programming language was chosen for its efficiency and versatility. The computers used for both acquisition and processing run on a Linux environment. Image processing is performed using the tools and functions provided by the open-source *OpenCV* library [30]. Concerning matrix manipulation and linear algebra, the *Eigen* library [31] was preferred. Least-squares optimisation is performed using the framework provided by the *ceres-solver* library [32]. Finally, the rest of the code was entirely written in C++.

1.5 Outline of the thesis

As stated previously, the main goal of this thesis is to investigate the use of visible and LWIR images for navigation purposes. It explores different ways of combining visible and thermal information and presents novel multispectral visual odometry solutions. The issues addressed are related to feature detection and matching, camera calibration, stereovision, non-linear optimisation and data fusion.

The thesis is divided into seven chapters, including the introduction and conclusion. The details and contributions of each chapter are presented as follows:

Chapter 2 introduces the notion of projective geometry. It aims at explaining how to map 3D points expressed in a global frame into pixel locations in an image. It also defines the camera model, coordinate systems and different rotation parametrisations used in this thesis. This is a background chapter which is necessary to fully comprehend the topics discussed in this thesis.

Chapter 3 presents a novel calibration pattern and an automatic calibration algorithm to calibrate both visible and infrared cameras as well as stereo rigs. Making use of LEDs, this crucial step prior to any visual localisation algorithm can be performed accurately with both modalities. This calibration technique was published in **Paper 1** and is used throughout the thesis, for all the experiments carried out to evaluate the performance of each VO algorithm. The contribution of this chapter comes from the pattern which was designed with a limited number of LEDs, powerful enough to create bright visual features in both visible and thermal images. But with an overall low power consumption to keep the calibration board as portable as possible. Furthermore, the specific shape of the pattern was designed to be detected automatically and to track its orientation.

Chapter 4 focuses on monocular navigation, as a first approach to perform visual odometry. The use of optical flow for multispectral feature tracking is investigated and an automatic keyframe selection algorithm is proposed. A full VO framework is presented and the trajectories obtained with visible and thermal images are compared. The main contribution of this chapter resides in the alternating combination of visible and LWIR images which shows that the VO process can be robustified to deal with extreme illumination conditions. Presented in **Paper 2**, this work is a first attempt to solve the multispectral localisation challenge and is reused in Chapter 5 and Chapter 6.

Chapter 5 tackles the feature matching problem between multispectral images. Two innovative methods are proposed: a local matching technique presented in **Paper 3**, where each feature is matched independently, and a global matching technique presented in **Paper 4**, where all features are reprojected in each stereo image and matched altogether. The matches are then exploited to compute scaled motion, solving the scale ambiguity which remains unanswered after Chapter 4.

Finally, Chapter 6 proposes a fusion scheme which combines IMU and VO measurements in order to robustify the motion estimation. Presented in **Paper 4**, an error-state Kalman filter fuses inertial data with the VO trajectories obtained from Chapter 5. It is demonstrated in **Paper 4** and **Paper 5** that VO can successfully correct IMU measurements and overcome the drifty nature of IMU integration. Moreover, the use of inertial data can compensate for VO failures and provide a pose estimate when pure VO cannot. Thus, the main contribution of this chapter comes from the fusion of inertial and multispectral VO data to create a robust multispectral visual-inertial localisation solution.

1.6 Published and submitted manuscripts

Conferences

Paper 1: **A. Beauvisage** and N. Aouf, “Low cost and low power multispectral thermal-visible calibration” in *IEEE Sensors*, Glasgow, UK, oct 2017, pp. 1-3. (published)

Paper 2: **A. Beauvisage**, K. Ahiska and N. Aouf, "Multimodal tracking framework for visual odometry in challenging illumination conditions" in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. (submitted)

Paper 3: **A. Beauvisage**, N. Aouf, and H. Courtois, “Multispectral Visual Odometry for Unmanned Air Vehicles” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, oct 2016, pp. 1994-1999. (published)

Paper 5: **A. Beauvisage** and N. Aouf, “Multimodal visual-inertial odometry for navigation in cold and low contrast environment”, in *European Conference on Mobile Robots (ECMR)*, Paris, France, sept 2017, pp. 1-6. (published)

Journals

Paper 4: **A. Beauvisage**, K. Ahiska and N. Aouf, "Robust multimodal visual-inertial navigation with visual odometry failure recovery" in *IEEE Transactions on Intelligent Transportation Systems.* (submitted)

2 | Projective geometry

In order to reconstruct the geometry of a scene and localise a camera or a vehicle in its environment, it is essential to understand the image acquisition process and more precisely, how a 3D point of the scene is mapped into the image. This section aims at introducing the notion of projective geometry and presenting the different models and series of transformations used to create a 2D image from the 3D scene. First, an introduction regarding the physical aspect of image acquisition will be given. The pinhole camera model will then be presented as well as the projection function which computes the position of a point in the image from its 3D location in the scene. Different ways to parametrise rotations and represent the camera attitude will be discussed. Finally, details about the way to describe camera motion over time will be given.

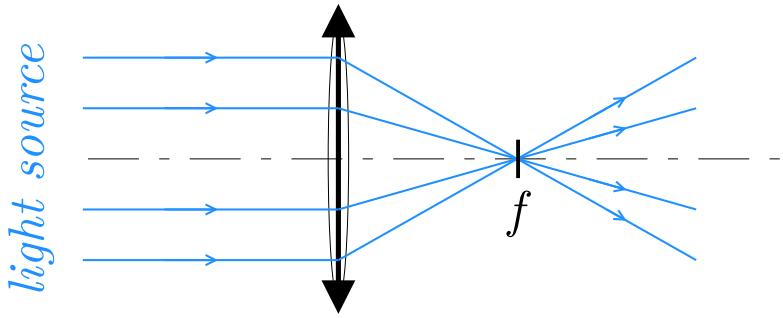


Figure 2.1: Lens focusing rays coming from the scene. All parallel rays pass through the focal point f.

2.1 Image formation

A camera creates an image by focusing rays of light onto the sensor where photons are collected and converted into an electric signal. In most cases, image sensors consist of an $n \times m$ array of small rectangular pieces. These elements are sensitive to a specific part of the electromagnetic spectrum. CCD (Charged Coupled Device) or CMOS (Complementary Metal-Oxide Semiconductor) for example, will react to visible light ($400 - 700$ nm), whereas bolometers will be affected by LWIR ($7 - 15$ μm). By successively opening and closing a shutter for a certain period of time (called *exposure time*), the sensor is exposed to light for a brief moment. The light flooding the sensor then generates a charge for each one of its element. After exposition, the electric signals created by each element are read out and assembled into an $N \times M$ image I where each pixel value $I(i, j)$, $\{i, j\} \in \{[1, N], [1, M]\}$ corresponds to the magnitude the signal generated by the sensor at $s(k, l)$, $\{k, l\} \in \{[1, n], [1, m]\}$. The magnitude of the signal and thus, the pixel intensity, is defined by the quantity of photons hitting the sensor during the acquisition. The more photons being detected, the higher the intensity. In order to obtain enough photons to generate a good signal, rays of light need to be concentrated onto the sensor using a lens (see Figure 2.1).

2.2 Pinhole camera model

A lot of different parameters can affect the quality of the image produced: the distortions induced by the lens, whether or not the sensor lies in the focal plane, the exposure time, etc... All these factors make the image formation a rather complex phenomenon that is not always easy to handle. Fortunately, it can be abstracted into simpler mathematical models, much more convenient to manipulate. The most common representation used for image-based navigation is the *pinhole camera model*. This model specifies that rays coming from different objects of the scene intersect in a single point named the *centre of projection*. This way, a camera can be considered as a point in space and rays can be drawn from the centre of projection to any other point in the scene. As illustrated in Figure 2.2, all lines intersect at the centre of projection and a 2D representation of the scene is created in the image plane. With this level of abstraction, certain effects such as focus or lens thickness can be ignored. It is important to notice that lenses are inverting the image (Figure 2.3a) but this can be easily rectified by rotating the sensor upside-down. As a result, the representation described by Figure 2.3b, where the image plane is located in front of the camera, is also a valid depiction of the pinhole camera model.

Point representation

Throughout the thesis, a point in a 3D Euclidean space will be represented as $\mathbf{X} = [X, Y, Z]^T$ and a point belonging to a 2D Euclidean space will be expressed as $\mathbf{x} = [x, y]^T$. To simplify calculations, the homogeneous representation could be adopted. In that case, vectors $\mathbf{X}$ and $\mathbf{x}$ will be extended with an additional positive scalar component λ:

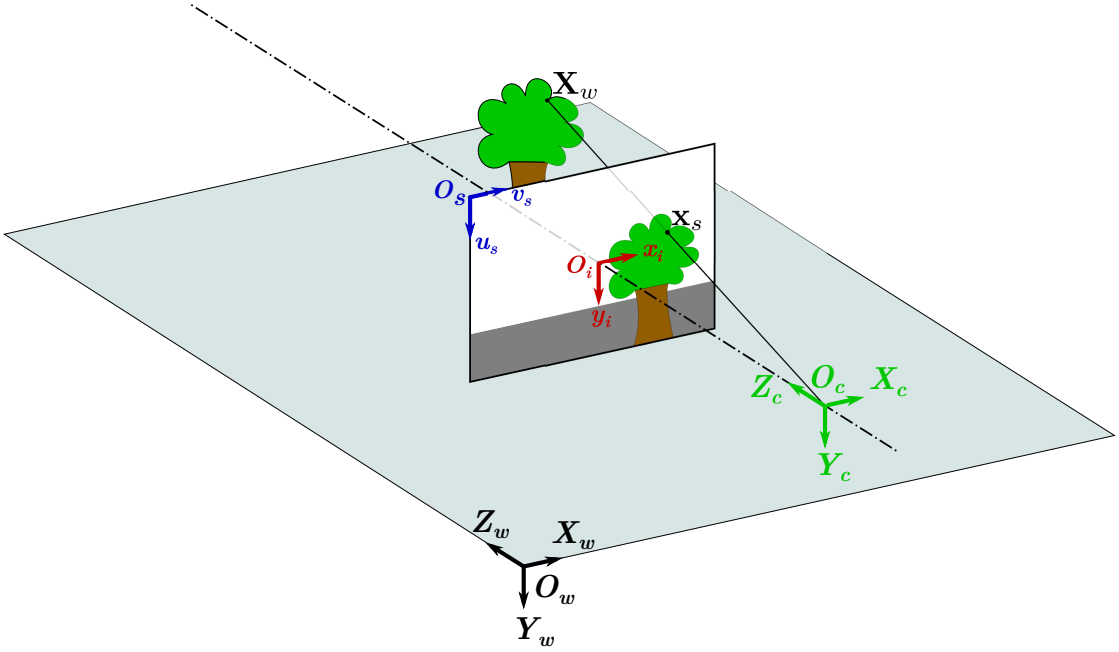
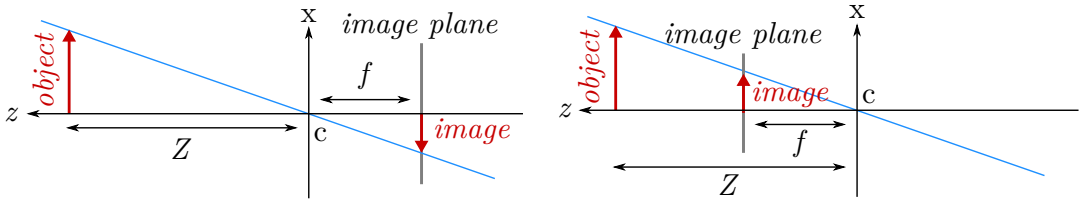


Figure 2.2: Pinhole camera model.



(a) Pinhole projection. The image is inverted.

(b) Simplified model where images are not rotated.

Figure 2.3: Different ways to represent a pinhole camera.

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ \lambda \end{bmatrix} \qquad \hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \lambda \end{bmatrix} \tag{2.1}$$

The inhomogeneous coordinates $\mathbf{X}, \mathbf{x}$ of $\hat{\mathbf{X}}, \hat{\mathbf{x}}$ corresponds to the first 3 and 2 components of the normalized homogeneous coordinate, respectively:

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \qquad \hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{2.2}$$

It can be noted that a singularity occurs when $\lambda = 0$. In projective geometry, it is used to represent points at infinity. This means a point located on the line passing through the centre of projection and the point $\mathbf{X}$ or $\mathbf{x}$, at an infinite distance from the centre of projection.

Coordinate systems

In order to derive the expression of the projection function π which maps a point in the scene to a pixel in the image, the following four coordinate systems need to be defined:

World coordinate frame $\{O_w, X_w, Y_w, Z_w\}$**:** reference frame of the scene, where the 3D objects lie. It can be a global frame if the vehicle's pose is known at the start of the acquisition, or it is usually selected to coincide with the first camera frame for relative navigation. Units: *m,rad.*

Camera coordinate frame $\{O_c, X_c, Y_c, Z_c\}$**:** coordinate system attached to the camera. It defines the position and orientation of the camera in the world coordinate frame. The origin corresponds to the centre of projection of the

camera. Units: *m,rad*.

Image plane $\{O_i, x_i, y_i\}$**:** plane where 3D points are projected to form the 2D view of the scene. The origin of the image plane is the centre of the plane, where the optical centre is projected. Unit: *m*.

Sensor plane $\{O_s, u_s, v_s\}$**:** coordinate system representing the image. Usually, the origin is located at the top left corner of the image. A pixel location is represented by the pair (u, v) where u corresponds to a row and v to a column. Unit: *pixel*.

To avoid confusions, any coordinate stated in this chapter will be followed by a subscript corresponding to the frame in which it is expressed $\{w, c, i, s\}$.

2.2.1 Projection into the image

The pinhole camera model maps a point from a 3D Euclidean space $\in \mathbb{R}^3$ to a point in a 2D Euclidean space $\in \mathbb{R}^2$:

$$\pi(\mathbf{X}) = \mathbf{x} : \quad \mathbb{R}^3 \mapsto \mathbb{R}^2 \tag{2.3}$$

where π is the projection operator. A 3D point in the scene can be expressed in the sensor frame by applying a series of transformations as summarised in Figure 2.4. It is important to notice that the central projection is unidirectional. This means information is lost during the projection from 3D space to 2D space. Indeed, only a line (direction) can be recovered from a 2D point. The depth of the object is lost during the projection. It is however possible to recover the missing information with stereovision or with a prior knowledge of the scene (size of an object, height/orientation of the camera, etc...). In the following sections detailed information will be given about each transformation.

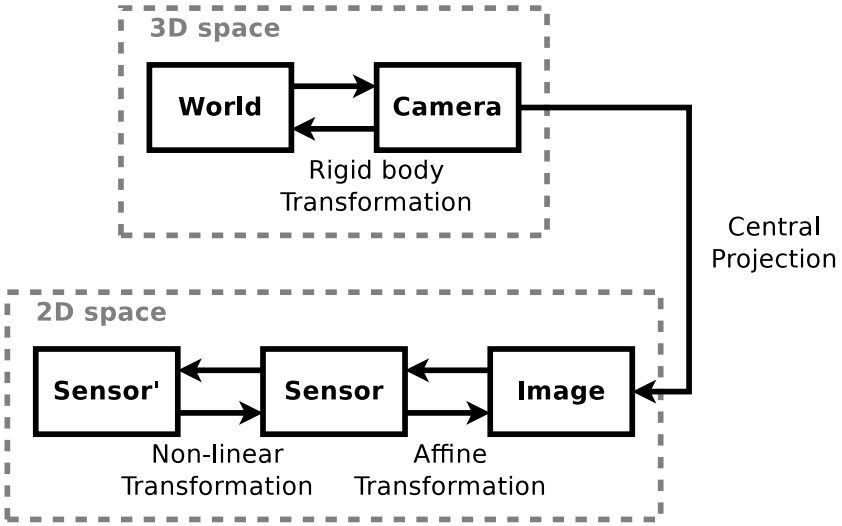


Figure 2.4: Projection pipeline from 3D scene points to 2D image features.

Rigid body transformation

Assuming a camera can move and rotate in every direction (6 DoF), its pose can be expressed in a matrix form with a combination of rotation and translation $T = [R|\mathbf{t}] \in \mathrm{SE}(3)$, where R is a 3×3 matrix $\in \mathrm{SO}(3)$ and $\mathbf{t}$ a 3×1 vector $\in \mathbb{R}^3$. Converting the coordinate of a point from the world coordinate frame to the camera frame can be performed by simply applying the rigid transformation T:

$$\begin{aligned} \mathbf{X}_c &= R\mathbf{X}_w + \mathbf{t} \\ \begin{bmatrix} \mathbf{X}_c \\ 1 \end{bmatrix} &= \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_w \\ 1 \end{bmatrix} = T \begin{bmatrix} \mathbf{X}_w \\ 1 \end{bmatrix} \end{aligned} \tag{2.4}$$

Similarly, the inverse transformation T^{-1} converts coordinates from the camera frame to the world frame:

$$\begin{bmatrix} \mathbf{X}_w \\ 1 \end{bmatrix} = \begin{bmatrix} R^T & -R^T\mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_c \\ 1 \end{bmatrix} \tag{2.5}$$

Central projection

As shown in Figure 2.3, a 3D point $\mathbf{X} = [X, Y, Z]^T$ can be projected into $\mathbf{x} = [x, y, z]$ with the following relationship:

$$x = \frac{f}{Z}X, \quad y = \frac{f}{Z}Y, \quad z = f \tag{2.6}$$

where f is the focal length. It corresponds to the distance between the centre of the lens and the image plane, where the image is in focus. This relation holds when the image plane is located in front of the lens like in Figure 2.3b. In the case where the focal plane is located behind the lens (Figure 2.3a), all components of $\mathbf{x}$ will be negative. Using homogeneous coordinates, the relation in (2.6) can be expressed as a linear system:

$$\mathbf{x}_i = \begin{bmatrix} x \\ y \\ \lambda \end{bmatrix}_i = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_c \tag{2.7}$$

λ is the scaling factor corresponding to the depth of the feature $\mathbf{x}_i$. When λ is equal to 1, the coordinate $\mathbf{x}_i$ is normalized and lies in the image plane.

Affine transformation

In the previous section, the central projection has been introduced for the simplest case, as described in Figure 2.3. However, other factors such as the difference in scale between the axes or the physical shape of pixel sensors need to be accounted for.

Usually, lenses are not completely spherical and some changes in scale can be measured between the horizontal and vertical axis. This difference is represented by adding a constant m to the focal length in the vertical axis. Moreover, pixels

do not necessarily have a square shape. In such cases, an additional skew factor s is employed to represent the asymmetry. The projection matrix becomes:

$$\mathbf{x}_i = \begin{bmatrix} x \\ y \\ \lambda \end{bmatrix}_i = \begin{bmatrix} f & s & 0 & 0 \\ 0 & f+m & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}_c \tag{2.8}$$

As illustrated in Figure 2.2, the origin of the image is usually chosen to match one of its corners. In this thesis, the origin is defined at the top left corner of the image, following the OpenCV library representation [30]. However, the centre of projection lies on the optical axis which is usually placed around the centre of the sensor. This results in the optical axis intersecting the image plane somewhere in the middle of the image. The position of the centre of projection in the image is called the principal point $pp = (c_u, c_v)$. To compensate for this different origin, a translation corresponding to the coordinate of the principal point is added to the projection matrix as follows:

$$\mathbf{x}_s = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix}_s = \begin{bmatrix} f & s & c_u & 0 \\ 0 & f+m & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{X}_c \tag{2.9}$$

Finally, positions in the image are expressed in pixels because it is the smallest element that can be resolved in the image. On the contrary, positions in the world and in the image plane are expressed in metric units (meters or millimeters). To convert a position from pixels to meters and vice-versa, one can simply apply the ratio (s_x, s_y) between a pixel and its physical size, usually given by the manufacturer:

$$\mathbf{x}_s = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix}_s = \begin{bmatrix} \frac{f}{s_x} & s & c_u & 0 \\ 0 & \frac{f+m}{s_y} & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{X}_c \tag{2.10}$$

(2.10) can be expressed with two matrices K and $\hat{P}$. K encapsulates the focal length and other parameters intrinsic to the camera. It is thus called the *intrinsic matrix*. Most of the time, the skew factor s is close to zero and can be neglected. $\hat{P}$ corresponds to the mathematical projection which convert a 4-dimensions homogeneous vector into a 3-dimensions homogeneous vector:

$$\begin{bmatrix} \frac{f}{s_x} & s & c_u & 0 \\ 0 & \frac{f+m}{s_y} & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{f}{s_x} & s & c_u \\ 0 & \frac{f+m}{s_y} & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = K\hat{P} \tag{2.11}$$

To summarise, the final projection that transforms a 3D point expressed in the world coordinate frame into a 2D point in the image can then be written as:

$$\mathbf{x}_s = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix}_s = \begin{bmatrix} \frac{f}{s_x} & s & c_u \\ 0 & \frac{f+m}{s_y} & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} [R|\mathbf{t}]\mathbf{X}_w \tag{2.12}$$

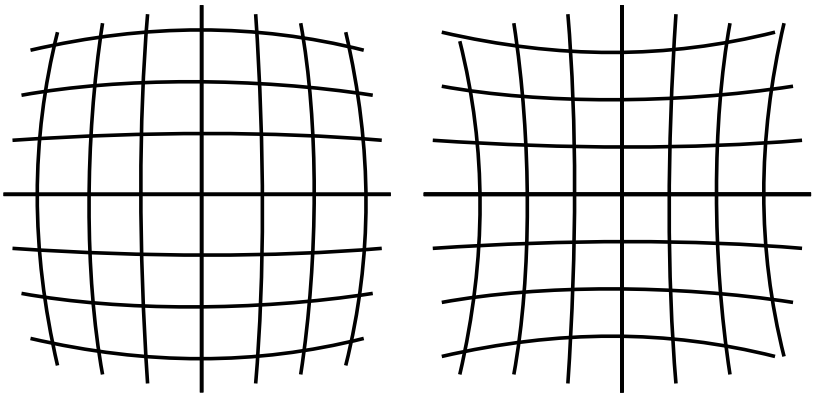
$$\mathbf{x}_s = K\hat{P}[R|\mathbf{t}]\mathbf{X}_w \tag{2.13}$$

To simplify calculation, the linear projection function can be expressed in a more compact form, with a single matrix P:

$$\mathbf{x}_s = P\mathbf{X}_w \tag{2.14}$$

where:

$$P = K\hat{P}[R|\mathbf{t}] \tag{2.15}$$



(a) Barrel distortion. (b) Pincushion distortion.

Figure 2.5: Example of radial distortions.

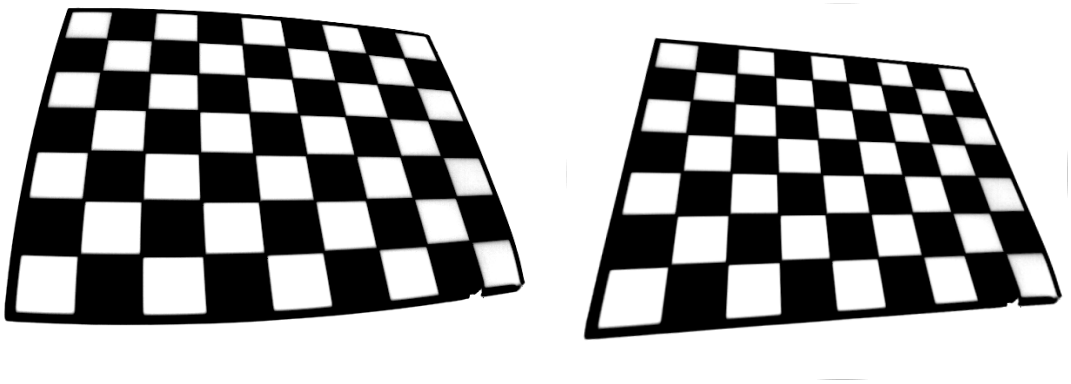
2.2.2 Lens distortions

In the previous section, the linear projection of a 3D point into a pixel position in the image has been presented, but there is one last phenomenon that has to be taken into account: non-linearities generated by the lens. These non-linearities lead to distortions in the image and can produce large errors if they are not compensated. Distortions can be due to the imperfections of the lens, we then speak of *barrel distortions* if due to the curvature of the lens; or *tangential distortions* if due to the non-planarity of the sensor. An example of barrel distortion is given in Figure 2.5.

For more information about the distortion model used in this thesis, the reader can refer to Section 3.3.

There are two options to take lens distortions into account in the projection function. The first one consists in adding the non-linear error to in the intrinsic parameters matrix K:

$$K = \begin{bmatrix} f_u & 0 & c_u + \delta x(x,q) \\ 0 & f_v & c_v + \delta x(y,q) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$



(a) Original image. (b) Undistorted image.

Figure 2.6: Example of distortions seen on a calibration board (IR) and removed after calibration. The lines have been straightened.

where $\delta x(x, q)$ corresponds to the displacement produced by the mapping of a measured point x into its corresponding undistorted position.

The second option is to create undistorted images by remapping all pixels into their undistorted positions. Undistorted images are then used as input for further processing. This way, non-linearities are removed from K, keeping the projection function linear. An example of undistorted image is given in Figure 2.6.

2.3 Rotation parametrisation

Rotations are used to represent the attitude (orientation) of a camera, or the vehicle on which the camera is mounted. Rotations can be expressed in many ways. In the following section, the representations relevant to this thesis are presented.

2.3.1 Angle-Axis and rotation vectors

Any 3D rotation can be thought of a rotation by an angle θ around a unit vector $\mathbf{u}$. As illustrated in Figure 2.7, a vector $\mathbf{x}$ is rotated around $\mathbf{u}$ into vector $\mathbf{x}'$. $\mathbf{x}$

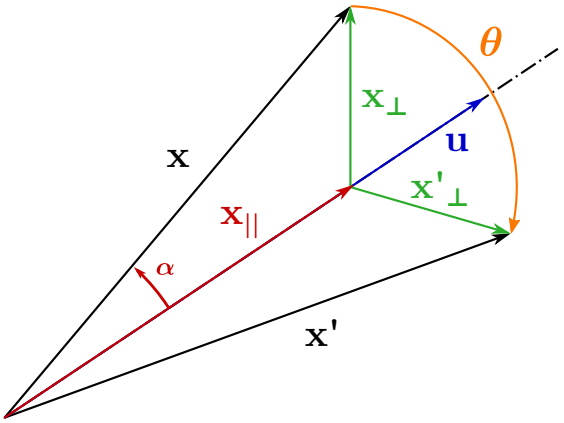


Figure 2.7: Angle-Axis representation.

can be decomposed into two components: $\mathbf{x}_{\parallel}$ parallel to $\mathbf{u}$, and $\mathbf{x}_{\perp}$ perpendicular to $\mathbf{u}$. This way:

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_{\perp} + \mathbf{x}_{\parallel} \\ \mathbf{x}_{\parallel} &= \mathbf{u}(\|\mathbf{x}\| \cos\alpha) = \mathbf{u}\mathbf{u}^T\mathbf{x} \\ \mathbf{x}_{\perp} &= \mathbf{x} - \mathbf{x}_{\parallel} = \mathbf{x} - \mathbf{u}\mathbf{u}^T\mathbf{x} \end{aligned} \tag{2.17}$$

Similarly,

$$\mathbf{x}' = \mathbf{x}'_{\perp} + \mathbf{x}'_{\parallel} \tag{2.18}$$

$\mathbf{x}_{\parallel}$ is not affected by the rotation, which means $\mathbf{x}_{\parallel} = \mathbf{x}'_{\parallel}$, and $\mathbf{x}_{\perp}$ undergoes a planar rotation in the plane normal to $\mathbf{u}$. A new orthonormal base can be defined to represent this plane, with basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ such as:

$$\begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{\perp} \\ \mathbf{u} \times \mathbf{x}_{\perp} \end{bmatrix} \tag{2.19}$$

Since $\mathbf{x}_{\parallel}$ and $\mathbf{u}$ are parallel, $\mathbf{u} \times \mathbf{x}_{\parallel} = \mathbf{0}$, which means:

$$\mathbf{u} \times \mathbf{x}_{\perp} = \mathbf{u} \times (\mathbf{x} - \mathbf{x}_{\parallel}) = \mathbf{u} \times \mathbf{x} - \mathbf{u} \times \mathbf{x}_{\parallel} = \mathbf{u} \times \mathbf{x} \tag{2.20}$$

Hence, the basis vectors can be expressed as:

$$\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\end{bmatrix} = \begin{bmatrix}\mathbf{x}_\perp\\ \mathbf{u}\times\mathbf{x}_\perp\end{bmatrix} = \begin{bmatrix}\mathbf{x}_\perp\\ \mathbf{u}\times\mathbf{x}\end{bmatrix}, \quad \text{satisfying } \|\mathbf{e}_1\| = \|\mathbf{e}_2\| \tag{2.21}$$

And the planar rotation θ in this plane can be expressed as:

$$\mathbf{x}'_\perp = \mathbf{e}_1\cos\theta + \mathbf{e}_2\ \sin\theta \tag{2.22}$$
$$\mathbf{x}'_\perp = \mathbf{x}_\perp\cos\theta + (\mathbf{u}\times\mathbf{x})\sin\theta \tag{2.23}$$

The final expression of the rotated vector $\mathbf{x}'$ becomes:

$$\mathbf{x}' = \mathbf{x}_\parallel + \mathbf{x}_\perp\cos\theta + (\mathbf{u}\times\mathbf{x})\sin\theta \tag{2.24}$$

The angle-axis parametrisation possesses 4 components (1 angle + 3 axis). However, it is possible to use a more compact representation called *rotation vector*. A rotation vector $\boldsymbol{\theta}$ corresponds to the scaling of the unit axis vector with the angle of rotation:

$$\boldsymbol{\theta} = \theta\mathbf{u} \tag{2.25}$$

Because the angle of rotation θ corresponds to the norm of the rotation vector $\boldsymbol{\theta}$, the angle-axis representation can simply be recovered from $\boldsymbol{\theta}$:

$$\begin{aligned}\theta &= \|\boldsymbol{\theta}\| = \sqrt{\boldsymbol{\theta}_x^2 + \boldsymbol{\theta}_y^2 + \boldsymbol{\theta}_z^2}\\ \mathbf{u} &= \frac{\boldsymbol{\theta}}{\theta}\end{aligned} \tag{2.26}$$

2.3.2 Rotation matrices

Rotation matrices are special linear transformations whose multiplication rotates a rigid body while preserving its length, angles and relative orientation (handedness). It is thus considered a rigid transformation. The set of all 3×3 rotation matrices is called *special orthogonal group* and denoted by SO(3). Elements of this group possess 9 parameters but have only 3 DoF. This arises specific constraints:

$$RR^T = I \;\Leftrightarrow\; R = R^T \tag{2.27}$$
$$det(R) = +1 \tag{2.28}$$

(2.27) corresponds to the orthogonal condition. Transformations that satisfy this requirement are part of the orthogonal group $O(3)$. However, this condition does not guarantee a rigid transformation. To avoid reflections, a second requirement is necessary. (2.28) ensures that the determinant is positive and unitary. Such matrices are called *proper* or *special* matrices and thus form the *special orthogonal group* SO(3).

There are two ways to represent the attitude of a camera. One can use the rotation that maps a point coordinate in the camera frame into its corresponding world coordinate; or the inverse rotation that maps a world coordinate into the camera coordinate. Either way, the conversion from one representation to the other is trivial with rotation matrices because each representation is the transpose of the other. In this thesis, the first convention is used, which gives:

$$\mathbf{X}_c = R\mathbf{X}_w \tag{2.29}$$

where $\mathbf{X}_w$ represents a point in the world reference frame, whereas $\mathbf{X}_c$ corresponds to its coordinate in the camera frame.

Properties

As a group, SO(3) possesses the following properties:

- An identity element

$$I_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.30}$$

- An inverse operation:

$$R^{-1} = R^T \tag{2.31}$$

- A composition operation:

$$R_{123} = R_1 R_2 R_3 \tag{2.32}$$

The composition is associative, but not commutative:

$$\begin{aligned} R_1 R_{23} &= R_{12} R_3 \\ R_1 R_2 &\neq R_2 R_1 \end{aligned} \tag{2.33}$$

Non-commutativity means that the order is important. Rotation R_{21} apply R_1 first, then R_2. Applying R_2 first would result in a different transformation. Additionally, the inverse of a matrix composition reverses the order in which matrices are multiplied. for example:

$$\begin{aligned} R_{123}^{-1} &= R_{123}^T \\ (R_1 R_2 R_3)^{-1} &= R_3^T R_2^T R_1^T \end{aligned} \tag{2.34}$$

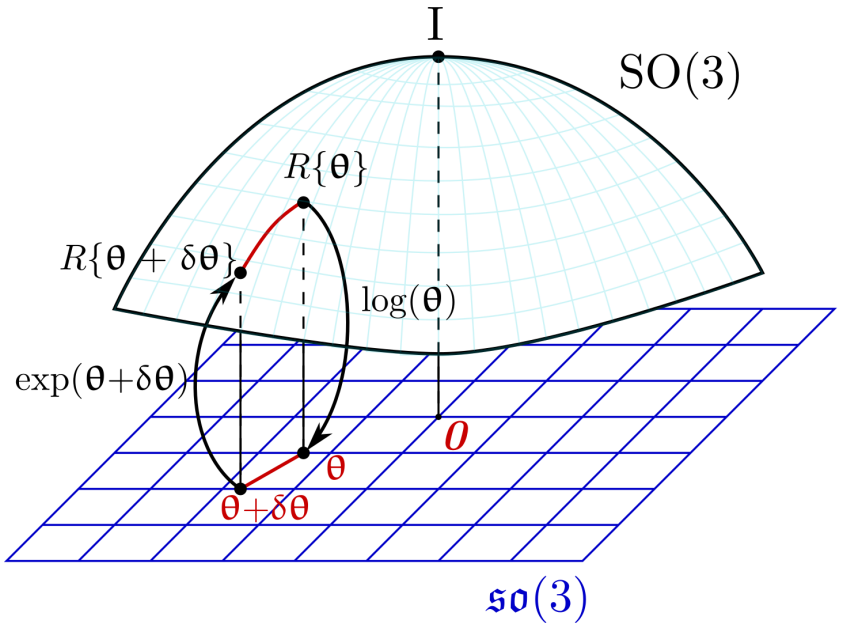


Figure 2.8: Illustration of the exponential and logarithmic maps.

Exponential and logarithmic maps

SO(3) also corresponds to a smooth manifold and forms a *Lie group* [33]. It means that a trajectory can be defined in SO(3) which continuously rotates a rigid body from an initial orientation $R\{\boldsymbol{\theta}\}$ to a new orientation $R\{\boldsymbol{\theta}+\delta\boldsymbol{\theta}\}$ (see Figure 2.8). With this property, time derivatives can be defined and infinitesimal calculus can be used in the 3D rotation space. The SO(3) group is associated to a *Lie algebra* denoted by $\mathfrak{so}(3)$. It coincides with the set of 3×3 skew-symmetric matrices and corresponds to the tangent space of SO(3) at identity. Skew-symmetric matrices are defined as:

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.35}$$

where $\boldsymbol{\omega}\in\mathbb{R}^3$ and $[\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$.

As illustrated in Figure 2.8, it is possible to go back and forth between SO(3)

and $\mathfrak{so}(3)$ using functions called *exponential* and *logarithmic maps*.
The exponential map is defined as:

$$\begin{aligned} &\exp : \mathfrak{so}(3) \mapsto \mathrm{SO}(3) \\ &\exp(\boldsymbol{\theta}) = e^{[\boldsymbol{\theta}]_\times} = R\{\boldsymbol{\theta}\} \end{aligned} \tag{2.36}$$

From a rotation vector $\boldsymbol{\theta} = \theta\mathbf{u} \in \mathbb{R}^3$, it can be rewritten as:

$$R = e^{\theta[\mathbf{u}]_\times} = I + \sin\theta[\mathbf{u}]_\times + (1 - \cos\theta)[\mathbf{u}]_\times^2 \tag{2.37}$$

This equation is also named the *Rodrigues rotation formula*.
The logarithmic map is defined as:

$$\begin{aligned} &\log : \mathrm{SO}(3) \mapsto \mathfrak{so}(3) \\ &\log(R) = [\boldsymbol{\theta}]_\times = \theta[\mathbf{u}]_\times \end{aligned} \tag{2.38}$$

with

$$\begin{aligned} \theta &= \frac{\mathrm{trace}(R) - 1}{2} \\ \mathbf{u} &= \frac{(R - R^T)^\vee}{2\sin\theta} \end{aligned} \tag{2.39}$$

where $[\cdot]^\vee$ is the inverse of $[\cdot]_\times$

2.3.3 Euler angles

Coordinate rotations

Rotations around a single coordinate axis are called *coordinate rotations* and are represented by specific rotation matrices:

$$R_i(\alpha) : \mathbb{R} \mapsto \mathrm{SO}(3); \quad i \in \{1, 2, 3\} \tag{2.40}$$

where i corresponds to the axis of rotation, respectively x, y and z. In navigation context, rotations around these axes are commonly referred to as *roll*, *pitch* and *yaw* with respective angles ϕ, θ and ψ [34]. Their corresponding rotation matrices are defined as follows:

$$R_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.41}$$

$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.42}$$

$$R_3(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.43}$$

Rotation sequence

3D rotations can be defined by a succession of three coordinate rotations. Many different sequences can be utilised such as (123), (121) or (231). The sequence (123) for instance is commonly used in aerospace engineering and computer graphics [34]. As illustrated in Figure 2.9, a rigid body rotated in this order undergoes a ψ coordinate rotation (yaw) around the z-axis first, followed by a θ coordinate rotation (pitch) around the y-axis, and a ϕ coordinate rotation (roll) around the x-axis. These rotations can be grouped into a single vector $\mathbf{u}$ containing the angle of each coordinate rotation. This vector is called *Euler angle vector*:

$$\mathbf{u} = [\phi, \theta, \psi]^T, \quad \mathbf{u} \in \mathbb{R}^3 \tag{2.44}$$

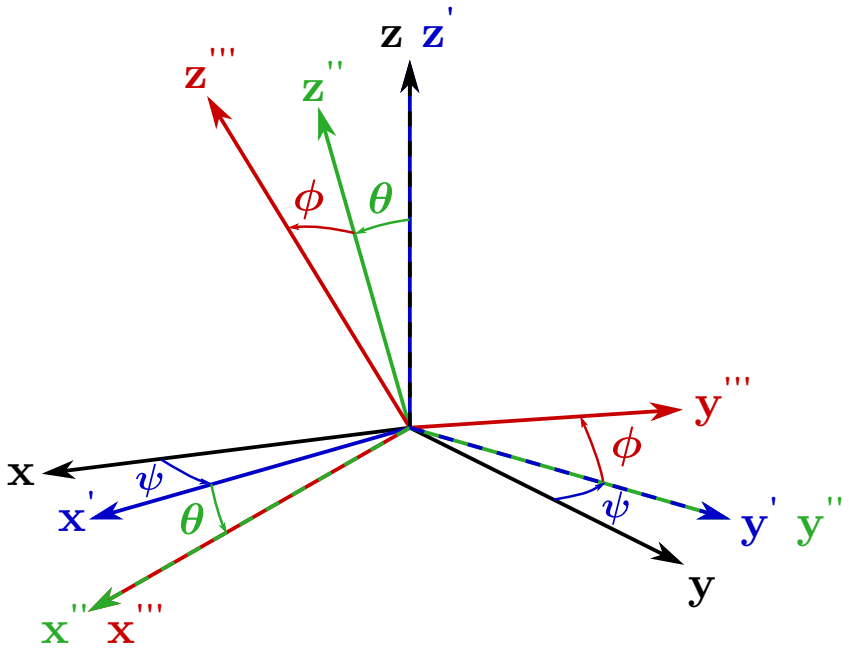


Figure 2.9: 3D rotation following the sequence (123). Blue, green and red rotations are applied respectively.

Euler angle vectors can be mapped to a single rotation matrix by concatenating the three successive coordinate rotations:

$$R(\mathbf{u}) : \mathbb{R}^3 \mapsto \mathrm{SO}(3) \tag{2.45}$$

For the sequence (123), the corresponding rotation matrix is:

$$R_{123} = R_1(\phi)R_2(\theta)R_3(\psi)$$

$$R_{123} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.46}$$

2.3.4 Quaternions

Definition

Another popular rotation parametrisation is the use of unit quaternions. Invented in the 19^{th} century by mathematician William Rowan Hamilton, quaternions are composed of four elements:

$$q = a + bi + cj + dk, \quad \{a, b, c, d\} \in \mathbb{R}, \quad \{i, j, k\} \in \mathbb{I}, \quad q \in \mathbb{H} \tag{2.47}$$

where $\{i, j, k\}$ are imaginary unit numbers forming a basis such as:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1 \\ \Rightarrow ij = -ji = k \; jk = kj = i, \; ki = -ik = j \end{aligned} \tag{2.48}$$

It is well known that complex numbers of unit length can rotate vectors in the 2D plane. For example, $z = e^{i\theta} \rightarrow \mathbf{x}' = z\mathbf{x}$, but this representation is limited to 2D spaces. To represent rotations in 3D spaces, a more complex representation is needed, and this is exactly what quaternions offer: an extended version of complex numbers to represent rotations in the 3D space. For simplicity, quaternions can be expressed as vectors:

$$q = [q_w, q_x, q_y, q_z]^T = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} = q_w + q_x i + q_y j + q_z k \tag{2.49}$$

In (2.49), q_w is called the scalar or real part of the quaternion, whereas $\mathbf{q}_v$ is named vector or imaginary part. The main advantage of using the vector form is that linear algebra can be applied to quaternion operations. A quaternion whose vector part is null is called *real quaternion*:

$$q = \begin{bmatrix} q_w \\ \mathbf{0}_{3\times 1} \end{bmatrix} \in \mathbb{H}_r \tag{2.50}$$

On the contrary, a quaternion whose scalar part is null is called *pure quaternion*:

$$q = \begin{bmatrix} 0 \\ \mathbf{q}_v \end{bmatrix} \in \mathbb{H}_p \tag{2.51}$$

Properties

- The sum of two quaternions is performed by adding both components:

$$p + q = \begin{bmatrix} p_w + q_w \\ \mathbf{p}_v + \mathbf{q}_v \end{bmatrix} \tag{2.52}$$

The quaternion addition is commutative and associative:

$$p + q = q + p \tag{2.53}$$
$$p + (q + r) = p + q + r \tag{2.54}$$

- Multiplication between two quaternions is more complex. Using the Hamilton representation, it can be expressed as:

$$p \otimes q = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix} \tag{2.55}$$

which can be simplified using the vector and scalar parts:

$$p \otimes q = \begin{bmatrix} p_w q_w - \mathbf{p}_v q_v \\ p_w \mathbf{q}_v + q_w \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{bmatrix} \tag{2.56}$$

The presence of cross-product makes the quaternion multiplication non com-

mutative when $\mathbf{p}_v \times \mathbf{q}_v \neq 0$. The cross-product appears to be null when one of the quaternions is pure ($p = \left[\begin{smallmatrix} p_w \\ \mathbf{0}_{3\times1} \end{smallmatrix}\right]$ or $q = \left[\begin{smallmatrix} q_w \\ \mathbf{0}_{3\times1} \end{smallmatrix}\right]$) or when both vector part are parallel ($\mathbf{p}_v \parallel \mathbf{q}_v$). The quaternion product is however associative.

In the same way rotation matrices are concatenated, composition between quaternions consist in multiplying them together:

$$q_{12} = q_1 \otimes q_2 \tag{2.57}$$

- The conjugate of a quaternion is defined as:

$$q^* = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} \tag{2.58}$$

$$(q \otimes q^*) = (q^* \otimes q) = \begin{bmatrix} \|q\|^2 \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{2.59}$$

$$(p \otimes q)^* = q^* \otimes p^* \tag{2.60}$$

- The norm of a quaternion is:

$$\|q\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \in \mathbb{R} \tag{2.61}$$

$$\|p \otimes q\| = \|q \otimes p\| = \|p\| \times \|q\| \tag{2.62}$$

- The identity quaternion is a pure quaternion such as:

$$q_I \otimes q = q \otimes q_I = q \tag{2.63}$$

$$q_I = \begin{bmatrix} 1 \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{2.64}$$

• The inverse quaternion is defined so that the multiplication between a quaternion and its inverse returns the identity:

$$q^{-1} \otimes q = q \otimes q^{-1} = q_I \tag{2.65}$$

$$\Rightarrow q^{-1} = \frac{q^*}{\|q\|} \tag{2.66}$$

Rotation operator

Quaternions can also be used as a rotation operator, to rotate vectors by an angle α around a vector $\mathbf{u}$. The rotation operator is defined as a double quaternion product:

$$\mathbf{x}' = q \otimes \mathbf{x} \otimes q^* \tag{2.67}$$

where $\mathbf{x}$ is a pure quaternion $\mathbf{x} = \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix} \in \mathbb{H}_p$, and q the quaternion representing the rotation α around $\mathbf{u}$.

Unit quaternion

Unit quaternions, whose norm $\|q\| = 1$, have the special property of being reversible:

$$q^{-1} = q^* \tag{2.68}$$

This is useful to parametrise rotations or orientations because it means that an inverse rotation can be performed using the conjugate quaternion.

Conventions

There are a lot of different ways to define a quaternion, and for this reason, the literature can be confusing. Although it does not make a big difference in the quaternion expression, it can produce dramatic changes in its interpretation and implementation. Thus, it is important to know which representation is used when dealing with quaternions.

- First, the order of the elements is crucial, and especially the real part. It can be first or last:

$$q = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} \quad \text{or} \quad q = \begin{bmatrix} \mathbf{q}_v \\ q_w \end{bmatrix} \tag{2.69}$$

- Secondly, the multiplication formula that have been presented earlier can be defined in two different ways:

$$\begin{gathered} ij = -ji = k \quad \text{or} \quad ji = -ij = k \\ right - handed \quad \text{or} \quad left - handed \end{gathered} \tag{2.70}$$

As suggested above, these equations determine the handedness of the system, which means that q_{right} will rotate a vector with the right-hand rule whereas q_{left} will rotate a vector with the left-hand rule.

- Additionally, the rotation operator (defined in Section 2.3.4) can either be passive or active. If passive, the operator rotates frames, otherwise, it rotates vectors.

- Finally, in the passive case where coordinate frames are modified, the direc-

tion of the operator can be different. It can convert a vector coordinates from the local frame ($\mathbf{x}_L$) to the global frame ($\mathbf{x}_G$):

$$\mathbf{x}_G = q \otimes \mathbf{x}_L \otimes q^* \tag{2.71}$$

Or from the global to the local frame:

$$\mathbf{x}_L = q \otimes \mathbf{x}_G \otimes q^* \tag{2.72}$$

Due to their robust mathematical nature, quaternions became popular in numerous fields such as mathematics, engineering or robotics [35] and this difference in representation has led to various conventions. Nevertheless, two dominant conventions have emerged: the *Hamilton* representation, mostly used in robotics and the *JPL* (Jet Propulsion Lab) representation, more suited for aerospace. In this thesis, the Hamilton representation is preferred as it was adopted by the libraries used for this work (Eigen [31], Google ceres [32], ROS [36], ...).

Exponential and logarithmic maps of unit quaternion

The set of unit quaternions is referred as S^3 and lies in a unit $3-sphere$, the 4-dimensional equivalent of a unit sphere. The tangent space of S^3 corresponds to the set of pure quaternions.

$$\begin{aligned} \exp &: \mathbb{H}_p \mapsto S^3 \\ \exp(\mathbf{q_\theta}) &= e^{\mathbf{q_\theta}} = \mathbf{q} \end{aligned} \tag{2.73}$$

However, unlike SO(3), a rotation vector in $\mathbb{R}^3$ cannot be mapped directly into a unit quaternion. Because of the double product involved in the quaternion rotation (see (2.67)), half of the rotated angle is used instead. The exponential

map becomes:

$$\mathbf{q} = e^{\boldsymbol{\theta}/\mathbf{2}} = e^{(\theta\mathbf{u})/2} = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2} = \begin{bmatrix} \cos\theta/2 \\ \mathbf{u}\sin\theta/2 \end{bmatrix} \tag{2.74}$$

The logarithmic map corresponds to the function that converts a unit quaternion into a pure quaternion:

$$\begin{aligned} \log : S^3 &\mapsto \mathbb{H}_p \\ \log(\mathbf{q}) = \mathbf{q}_{\boldsymbol{\theta}} &= \begin{bmatrix} 0 \\ \theta\mathbf{u} \end{bmatrix} \end{aligned} \tag{2.75}$$

with

$$\begin{aligned} \theta &= 2\ \arctan(\|\mathbf{q}_v\|, q_w) \\ \mathbf{u} &= \frac{\mathbf{q}_v}{\|\mathbf{q}_v\|} \end{aligned} \tag{2.76}$$

2.3.5 Comparison

Euler angles are probably the most common parametrisation used to represent 3D rotations due to their intuitive nature. Because they represent rotations around a physical axis, they are easy to interpret and to visualise. On top of that, they can easily be converted into rotation matrices using sine and cosine functions. However, Euler angles possess a major drawback: singularities. Singularities arise when one of the angles reaches a critical value. Changes in the other two angles then become undistinguishable. This phenomenon is known as *Gimbal lock* and is different for every sequence. Sequence 123, for example, produces a gimbal lock when the pitch angle reaches 90° up. In that case, it is impossible to distinguish between roll an yaw (see Figure 2.10). The solution to this problem consists in using a different sequence representation when the attitude of the system comes

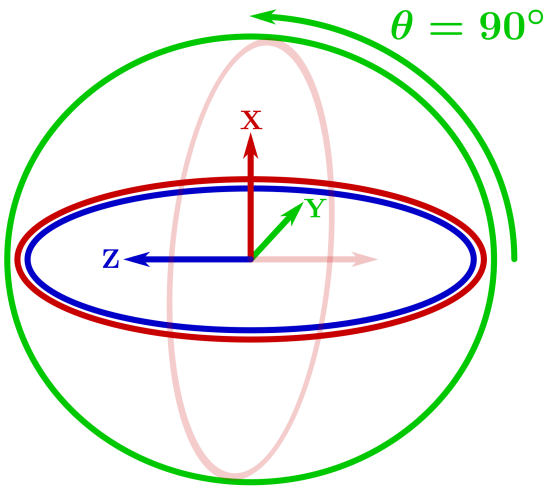


Figure 2.10: Example of Gimbal lock. Roll and Yaw rotations cannot be distinguished.

close to a singularity, but this is not ideal. Mathematically speaking, singularities could be seen when certain elements of the rotation matrix become null. Using the same sequence (123) defined in (2.46), $\cos(90°) = 0$ which leads to $r_{23} = r_{33} = r_{12} = r_{11} = 0$. But the conversion between rotation matrices and Euler angles is defined as:

$$\mathbf{u}_{123}(R) = \begin{bmatrix} \phi_{123} \\ \theta_{123} \\ \psi_{123} \end{bmatrix} = \begin{bmatrix} \arctan(\frac{r_{23}}{r_{33}}) \\ -\arcsin(r_{13}) \\ \arctan(\frac{r_{12}}{r_{11}}) \end{bmatrix} \tag{2.77}$$

ϕ_{123} and ψ_{123} are thus undefined.

Rotation vectors are another compact representation but also suffer from singularities (at 180°). To avoid this, one can employ rotation matrices, but such representation possesses many components, which makes it less computationally efficient and require more resources.

Quaternions on the other hand, provide a good compromise between stability and efficiency. Because they are expressed in 4 dimensions and are made of only

4 elements, they do not suffer from singularities when representing 3D rotations. They are however less intuitive. It is indeed hard to visualise how quaternions work and how they are affecting a rigid body. Their temporal derivative is very stable though, which makes quaternion the best alternative for integrating inertial data. That is why they have become highly popular for navigation purposes in recent years. The main drawback of quaternions is the unity norm constraint which is quadratic and thus hard to ensure. This makes it impossible to use with standard optimisation algorithms. Two solutions are then possible. The first one consists in renormalising the quaternion parameter at each iteration. The second option is to use a local 3D parametrisation, such as rotation vectors, in the optimisation which then updates the attitude, stored in a quaternion form [37]. In this thesis, most of the algorithms were developed using quaternions and rotations matrices. Rotation vectors were utilised as local parametrisation. Even though they were used in the early stages, Euler angles have been replaced by quaternions.

2.4 Camera motion

2.4.1 Rigid body transformation

A *camera pose* is the rigid transformation relating a camera position and attitude to its reference frame. It comprises a 3D rotation and a 3D translation. As seen in the previous section, there are several ways to parametrise the attitude. Using a rotation matrix $R \in SO(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, a point coordinate from the reference frame can be expressed in the camera frame such as:

$$\mathbf{X}_c = R\mathbf{X}_r + \mathbf{t} \tag{2.78}$$

R and $\mathbf{t}$ can be combined into a single homogeneous transformation matrix

$T \in \mathrm{SE}(3)$. Such representation requires point coordinates to be extended to homogeneous coordinates. The previous relationship then becomes:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_c = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_w \tag{2.79}$$

The set of all transformation matrices T, defined as above, form a group of affine rigid motion. It corresponds to a Lie group called *special Euclidean group* and denoted SE(3). SE(3) has the structure of a 6-dimensional manifold and is associated to a lie algebra $\mathfrak{se}(3)$ whose base is made of six 4×4 matrices, corresponding to infinitesimal rotations or infinitesimal translations around each axis. $\mathfrak{se}(3)$ can thus be represented by a 6-element vector.

Exponential and logarithmic maps

As it has been seen previously with other Lie groups, camera poses expressed in $\mathfrak{se}(3)$ can be converted to SE(3) and vice versa using the exponential and logarithmic maps. Let $\mathbf{v} = [\boldsymbol{\theta}, \mathbf{t}]^T \in \mathfrak{se}(3)$ be the 6-dimensional vector where $[\boldsymbol{\theta}]_\times \in \mathfrak{so}(3)$ corresponds to the rotation parameters and $\mathbf{t} \in \mathbb{R}^3$ corresponds to the translation parameters. The exponential map is the function defined such as:

$$\begin{aligned} &\exp : \mathfrak{se}(3) \to \mathrm{SE}(3) \\ &\exp(\mathbf{v}) = T_{4\times4} \end{aligned} \tag{2.80}$$

Its closed-form is:

$$e^{\mathbf{v}} = \begin{bmatrix} e^{[\boldsymbol{\theta}]_\times} & V\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{2.81}$$

where $e^{[\boldsymbol{\theta}]_\times}$ is the SO(3) exponential map defined in (2.36) and:

$$V = I_3 + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2 \tag{2.82}$$

The logarithmic map is defined as:

$$\begin{aligned} &\log : \mathrm{SE}(3) \to \mathfrak{se}(3) \\ &\log(T) = \mathbf{v} = \begin{bmatrix} V^{-1}\mathbf{t} \\ \log(R) \end{bmatrix} \end{aligned} \tag{2.83}$$

with V defined in (2.82) and $\log(R)$ the SO(3) logarithmic map defined in (2.38).

2.4.2 Dead reckoning

Now that it has been seen how to represent a camera pose at a certain point in time, its evolution with time can be modelled with the use a process called dead reckoning. This refers to the process of estimating the current camera pose based on previously determined poses. Camera motion becomes a function of time, corresponding to the accumulation of relative transformations between the frames. The camera pose at a specific time t is then represented by:

$$T(t) = \begin{bmatrix} R(t) & \mathbf{t}(t) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \quad T \in \mathrm{SE}(3) \tag{2.84}$$

where $R(t)$ is the rotation matrix representing the orientation of the camera at time t, and $\mathbf{t}(t)$ corresponds to its position. As the camera is moving, point coordinates in the camera frame change as well. Assuming the initial camera pose coincides with the world coordinate frame, points in the world coordinate frame can then be expressed in the current camera frame at time t with the following formula:

$$\mathbf{X}(t) = R(t)\mathbf{X}_0 + \mathbf{t}(t) \tag{2.85}$$

$$\begin{bmatrix} \mathbf{X}(t) \\ 1 \end{bmatrix} = T(t) \begin{bmatrix} \mathbf{X}_0 \\ 1 \end{bmatrix} \tag{2.86}$$

where $\mathbf{X}_0 = \mathbf{X}(t_0)$ corresponds to any point in the initial frame. But the first camera frame does not necessarily have to coincide with the world coordinate frame. If it is not the case, $T(t_0) \neq I_{3\times3}$ represents the initial pose. $\mathbf{X}(t)$ is estimated recursively by concatenating relative transformation between poses:

$$\mathbf{X}(t_i) = T(t_{i-1}, t_i)\mathbf{X}(t_{i-1}) \tag{2.87}$$

with $T(t_{i-1}, t_i)$ the relative rigid body transformation between camera poses at time t_{i-1} and t_i. This means that any point can be converted from its initial estimate at t_0 to its current estimate at t_c by:

$$\mathbf{X}(t_c) = T(t_0, t_c)\mathbf{X}_0 \tag{2.88}$$

with

$$\begin{aligned} T(t_0, t_c) &= T(t_{c-1}, t_c)T(t_{c-2}, t_{c-1}) \dots T(t_0, t_1) \\ &= \prod_{i=1}^{c} T(t_{i-1}, t_i) \end{aligned} \tag{2.89}$$

Reciprocally, if a point has been estimated in the current camera frame, it can be expressed in the initial frame following:

$$\begin{aligned} \mathbf{X}_0 &= T(t_c, t_0)\mathbf{X}(t_c) \\ &= T^{-1}(t_0, t_c)\mathbf{X}(t_c) \\ &\Rightarrow TT^{-1} = I \end{aligned} \tag{2.90}$$

By detecting and matching 2D features in consecutive images, it is possible to

estimate the relative transformation between two camera poses $T(t_{i-1}, t_i)$, which is then accumulated over time from a known location to the current location. As the camera moves and the current estimate is propagated, errors are also accumulated, introducing a drift which grows with time. This is a recurrent problem with dead reckoning processes, which can be even more significant with other systems like inertial navigation. This drift can be reduced by fusing the pose estimate with other sensors such as IMUs, or compensated by the addition of global positioning sensors such as GPS receivers.

2.5 Conclusion

In this background chapter the image acquisition process was explained and the projection function introduced. It has been seen that the projection function maps a 3D point from the scene into a 2D pixel location in the image. The function can be expressed as a linear transformation represented by the matrix P when processing rectified images. If the original images are used instead, the function then becomes a non-linear transformation due to the lens distortions. The distortions and camera parameters included in P should be estimated during the calibration process, a necessary step prior to any localisation algorithm or, more generally, prior to any image processing technique requiring the use of projective geometry.

Rotations are used to represent the camera orientation and they bring additional non-linearities to the process. Unlike lens distortions that can be removed by rectifying the images, the camera orientation cannot be changed. Thus, it is important to use the best suited parametrisation for the task being performed. Several representations were presented in this chapter and will be used in the rest of this thesis. Rotation matrices are utilised in the image processing algorithms, to

project and triangulate points, for example. They are also used in the calibration process presented in Chapter 3. This choice was made based on the OpenCV library used to manipulate images and which offers powerful tools relying on rotation matrices. However, quaternions are preferred to accumulate transformations and to estimate the current camera pose due to their better numerical stability. This is also the parametrisation used by the Kalman filter in Chapter 6. Finally, rotation vectors are utilised as a local parametrisation for bundle adjustment because they possess only 3 DoF and are therefore best suited for the optimisation.

3 | Camera calibration across modalities

Camera calibration is the primary step to many computer vision applications. In this chapter, the process is presented and its importance for 3D reconstruction is demonstrated. An exhaustive list of state-of-the-art multispectral calibration patterns and calibration techniques is given. The specificity of stereo setups is presented and the different parameters to be estimated are introduced. Then, a new calibration pattern with specific shape and materials is proposed. It has been designed for multispectral stereo cameras, with features distinguishable in both visible and infrared domains. But it can be also be utilised for monocular, stereo visible or stereo infrared (SWIR/LWIR) calibrations. A novel multispectral calibration process is also presented. It estimates both the intrinsic parameters of single cameras and the extrinsic parameters of stereo rigs. Calibration and reconstruction accuracy are tested to justify the use of this technique in the rest of the thesis, for both monocular and stereo setups.

3.1 Introduction

Camera calibration is essential to many computer vision related applications such as stereo feature matching and depth map computation or more generally autonomous navigation [23]. The precision of such systems depends greatly on the accuracy of the calibration. Thus, it is important to have a precise and reliable calibration process. Calibrating consists in estimating the optical properties of a camera (intrinsic parameters), and in the case of stereo setups, their relative orientation and position (extrinsic parameters) as well. Camera calibration is a problem that has been extensively studied, but a significant interest in infrared and multispectral imaging has grown over the past few years as infrared sensors are providing additional information about the scene and are more robust to illumination changes [19, 26, 38, 27, 23, 39]. The need to calibrate such systems has emerged and a few specific calibration techniques have been developed. However, these methods are usually less precise than those used in the visible domain [40].

Most of the calibration patterns used with visible cameras are chessboards because they can be printed easily and provide a large contrast between black and white regions, which makes feature detection straightforward [41]. However, this is not the case in the thermal domain, where ink and paper cannot be distinguished. Several alternative solutions have been proposed to tackle this problem. In [42] for example, it was suggested to heat the chessboard with a flood lamp to make use of the difference in emissivity between the black and white regions. However, Vidas criticised this method as it fails to produce sharp corners and lacks of precision [43]. Instead, he suggested a mask-based approach with a pattern cut out of cardboard and placed in front of a heated backdrop. In [44], Kim proposed to use thin lines of copper instead of squares. To avoid heating the whole pattern the authors of [45] have placed resistors at the centre of each square. Corners can thus be detected

in the visible domain and the centres can be precisely localised in the thermal domain. Finally, [40] used light bulbs to create a grid of points instead of utilising a chessboard.

This work focuses on the calibration of multispectral setups and more specifically on the combination of visible and far infrared images [46]. For these types of systems, the challenge comes from selecting materials with different properties in order to build a visual calibration pattern that can be easily seen in both modalities. This means that the pattern should be made of elements with a high difference in emissivity, although components with low emissivity tend to be highly reflective and cameras cannot be placed directly in front of the calibration pattern because their reflection interferes with the pattern detection [23]. As explained earlier, heating the pattern could be a solution but it is not practical and results in blurry images. Thus, a new approach based on LED light and similar to [40] is adopted.

The development of a new multispectral calibration process was motivated by the lack of convenient visible-thermal calibration patterns and the absence of precise method to detect and identify similar features in multimodal images. Moreover, all the current patterns proposed in the literature possess certain drawbacks making them impractical. Reflective materials create noise when the camera passes in front of the calibration board; heated patterns take time to set up and produce blurry edges; resistors and light bulbs require a certain power to function and are less portable. Therefore, a new, simple and easy-to-use calibration pattern is proposed. It can be utilised with both infrared and visible cameras separately or together in a stereo manner. The pattern is made of low-cost elements and requires only a few Light Emitting Diodes (LED) to produce distinguishable features. Hence, it can be powered by a small battery attached to the calibration board. This allows the board to be moved freely in front of the cameras and it

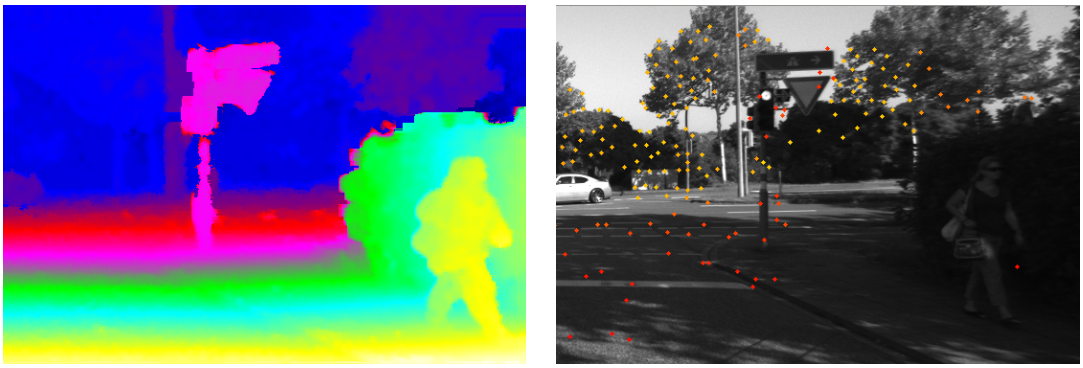
can be used immediately, without requiring any heating system. The proposed calibration pattern is identified in the images within an automatic calibration procedure. Indeed, the algorithm is able to automatically detect the different elements of the calibration pattern and to compute its orientation. The cameras intrinsic and extrinsic parameters are then estimated in an optimisation scheme.

3.2 The stereovision principle

To fully understand the calibration process of stereo cameras it is important to know what a stereo setup is and how it is used. Thus, the stereovision principle is introduced first.

A stereo system is made of two synchronised cameras attached to a common structure. It is used to reconstruct the depth of a scene by triangulation. As explained in Chapter 2, the projection function is unidirectional and its inverse can only recover a line (direction) from a 2D feature in the image. This is due to the fact that the depth information is lost during the projection from 3D to 2D. However, if the feature is observed in a stereo pair of images and both intrinsic and extrinsic parameters are known, one ray can be estimated from each image. The 3D position of the observed feature can then be estimated at the location where the two rays intersect. Therefore, in the same way humans are estimating the depth of a scene based on the different views perceived by both eyes, the depth of a feature can be recovered from the shift produced by stereo images, called *disparity*. Even though, the depth estimation is intuitive for human eyes, precise knowledge of the intrinsic and extrinsic parameters is however required for a computer to build a depth map.

As illustrated in Figure 3.1, depth images can be either dense or sparse. Dense maps contain the depth of every pixel in the image but take more time to generate



(a) Dense (b) Sparse

Figure 3.1: Example of dense and sparse depth maps.

because every pixel needs to be processed. Moreover, it usually requires post processing (smoothing) if used for scene segmentation or recognition as depth can change drastically between neighbouring pixels. Sparse depth maps on the other hand are faster to process because only the depth of a limited number of points is computed. In both cases, the matching process between stereo images is a crucial step and the accuracy of the resulting depth map depends greatly on the quality of the matching process.

Another important aspect of stereovision is the synchronisation. To guarantee that the relative transformation between the stereo camera frames has been properly estimated during calibration and is constant throughout the acquisition, images need to be acquired at the exact same time. A slight delay during the acquisition might have dramatic effects on the calibration and/or the motion estimation. For example, if the stereo setup is embedded on a car moving forward, a delay will make one image appear in front of the other and will create the wrong disparity between features, leading to an incorrect depth estimation.

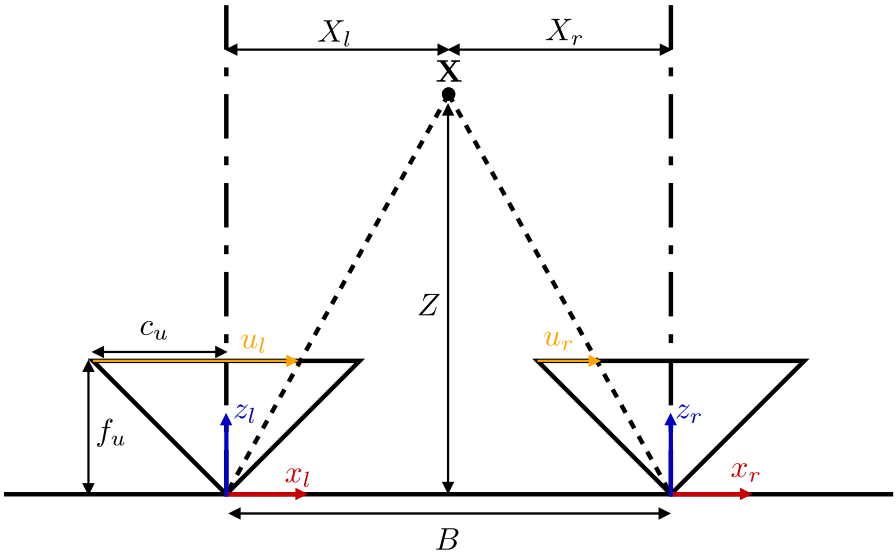


Figure 3.2: Simple triangulation in the ideal case where both cameras are aligned.

The ideal case

Let us consider the special case where images are acquired at the exact same time and the optical axis of both cameras are aligned, as illustrated in Figure 3.2. It is also assumed that both cameras share the same focal length (f_u, f_v) and principal point (c_u, c_v). Because they have the same orientation, the cameras are only related by a translation called the *baseline* of the system, denoted by B. In this specific case, a point $\mathbf{X}$ is mapped to $\mathbf{x}_l$ in the left image and to $\mathbf{x}_r$ in the right image, with the following simplified projection:

$$\mathbf{x}_l = \begin{bmatrix} u_l \\ v_l \end{bmatrix} = \begin{bmatrix} f_u \dfrac{X_l}{Z_l} + c_u \\ f_v \dfrac{Y_l}{Z_l} + c_v \end{bmatrix} \tag{3.1}$$

$$\mathbf{x}_r = \begin{bmatrix} u_r \\ v_r \end{bmatrix} = \begin{bmatrix} f_u \dfrac{X_r}{Z_r} + c_u \\ f_v \dfrac{Y_r}{Z_r} + c_v \end{bmatrix} \tag{3.2}$$

where $\mathbf{X}_l = [X_l \ Y_l \ Z_l]^T$ is the 3D point expressed in the left camera coordinate system and $\mathbf{X}_r = [X_r \ Y_r \ Z_r]^T$ is the same point expressed in the right camera coordinate system. Because the camera coordinate frames are related by a single translation along the x-axis, it can be noted that $Z_l = Z_r = Z$. Hence, subtracting equations in (3.1) leads to the following disparity equation:

$$d = u_l - u_r = f_u \frac{X_l - X_r}{Z} \tag{3.3}$$

Moreover, in this special configuration, $X_l - X_r$ corresponds to the baseline of the stereo setup. (3.3) can thus be rewritten:

$$d = f_u \frac{B}{Z} \tag{3.4}$$

$$Z = f_u \frac{B}{d} \tag{3.5}$$

where B is the baseline and d the disparity of the observed feature. Once Z has been computed, its value can be substituted in (3.1) to retrieve the X and Y components of $\mathbf{X}_l$:

$$X_l = (u_l - c_u) \frac{Z}{f_u} \tag{3.6}$$

$$Y_l = (v_l - c_v) \frac{Z}{f_v} \tag{3.7}$$

(3.4) shows that the depth Z is directly proportional to the baseline and inverse proportional to the disparity. It means that the baseline impacts directly on the depth accuracy, but a baseline too wide will not provide enough overlap between the images for close objects to be seen by both cameras. The baseline should then be adapted depending on the application and the distance of the features to be observed. Similarly, a mismatch between two features will lead to the wrong

disparity measurement and thus to an erroneous depth.

It is important to notice that in the ideal case where cameras are aligned, stereo matching can be improved. Indeed, with the configuration epipolar lines are horizontal, which means corresponding features are located on the same image row. As it will be explained in more details in Chapter 4, this simplifies considerably the search for matches which is then limited to a single row. Besides, it reduces the risk of incorrect matches by eliminating potential wrong candidates elsewhere in the image. Therefore, horizontal epipolar lines is a useful property of stereo systems as the accuracy of the motion estimation depends on the quality of the matching. But it also means that a good calibration is required in order to rectify the images properly and to legitimately make this assumption.

3.3 Calibration parameters

3.3.1 Monocular Calibration

As explained in Section 2.2, some specific information about the camera are required to accurately project 3D points into the image. These information are called intrinsic parameters because they are specific to the camera used. The role of monocular calibration consists in estimating these parameters as precisely as possible. Using the pinhole camera model to represent the projection of 3D points into the 2D image plane, the intrinsic parameters of a camera are represented with matrix K and vector $\mathbf{D}$:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$\mathbf{D} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & p_1 & p_2 \end{bmatrix}^T \tag{3.9}$$

K encapsulates the geometric properties such as the focal length and principal point of the camera, whereas $\mathbf{D}$ contains information about the lens distortions. Radial and tangential distortions, with respective parameters k_i and p_j, are approximated with the following model [47]:

$$\begin{aligned} u' &= u\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_1uv + p_2(r^2 + 2u^2) \\ v' &= v\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + p_1(r^2 + 2v^2) + 2p_2uv \end{aligned} \tag{3.10}$$

where (u, v) is the feature location in the original image, $r^2 = u^2 + v^2$ and (u', v') the corresponding location of the undistorted feature. These distortions need to be accounted for, but rather than including them in the intrinsic parameters and adding non-linearities to the projection process, it is preferred to rectify the images first, so (u', v') can be measured directly. This way, the projection of a 3D points is kept linear and can simply be represented by the projection matrix P.

3.3.2 Stereo calibration

Stereo rigs are made of two cameras attached together and are used to recover the depth of the 3D scene. As explained previously in Section 3.2, features can be triangulated easily in the ideal case where both images are aligned and the baseline is known. Moreover, with this special configuration all epipolar lines are horizontal which simplifies the search for stereo matches. But in reality images are not completely aligned because the cameras are not identical and their sensors are not physically lined up perfectly. This is especially the case when considering multispectral setups with cameras made by different manufacturers and having different optical properties. Therefore, to be able to rectify and align the images from a software point of view, it is important to know precisely the camera information. To estimate the intrinsic properties of each camera, monocular calibration can

be performed separately. But in order to recover the extrinsic parameters, stereo calibration needs to be run on a series of synchronised images of a known calibration pattern. These calibration parameters encapsulate the geometric properties of the stereo rig, which corresponds to the relative transformation between the two camera poses. Therefore, stereo calibration refers to the process which consists in finding the correct relationship between the two cameras.

By setting the left camera pose as the origin of system, the extrinsic parameters are made of a 3D rotation representing the difference in orientation between the two optical axes, and a 3D translation representing the distance between the two camera centres (baseline) and their potential misalignment. Knowing these parameters is essential to construct the projection matrix of each camera. An homogeneous point $\mathbf{X}$ is then projected into $\mathbf{x}_l$ and $\mathbf{x}_r$ in the left and right image respectively as follows:

$$\begin{aligned} \mathbf{x}_l = \begin{bmatrix} u_l \\ v_l \\ \lambda \end{bmatrix} &= P_l \mathbf{X} = K_l I_{3\times 4} \begin{bmatrix} X \\ Y \\ Z \\ \lambda \end{bmatrix} \\ \mathbf{x}_r = \begin{bmatrix} u_l \\ v_l \\ \lambda \end{bmatrix} &= P_r \mathbf{X} = K_r [R|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ \lambda \end{bmatrix} \end{aligned} \tag{3.11}$$

After rectification, both cameras have the same orientation and are only separated by a translation along the x-axis, which corresponds to the baseline of the stereo rig B (see Section 3.2). The projection matrix for the second camera becomes:

$$\mathbf{x}_r = K \begin{bmatrix} 1 & 0 & 0 & -B \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ \lambda \end{bmatrix} \tag{3.12}$$

For the sake of simplicity, it is also common to transform the images during stereo rectification in such a way that both cameras have the same focal length and principal point. The intrinsic calibration matrix then coincides and are represented by K in the equation above.

3.4 Multispectral Calibration

To tackle the challenges arisen by multispectral setups, a novel monocular and stereo calibration process for visible and thermal images is presented.

3.4.1 Approach

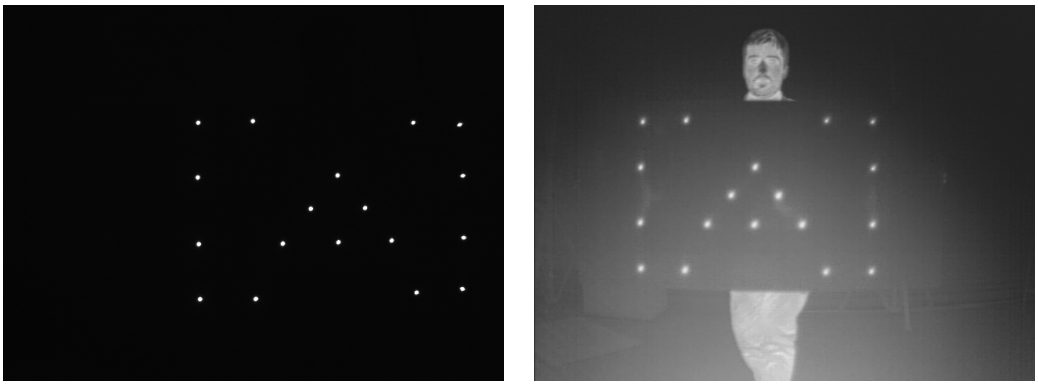
The main issue for multispectral setups is the choice of materials that needs to be easily detected in both image modalities. A lighting-based approach was chosen as it is highly discriminative in the visible domain and it produces heat which can also be captured by thermal cameras. Standard calibration patterns require numerous pattern points to be able to estimate the calibration parameters properly [40]. This means that such systems need a lot of power to light all the bulbs and this can be problematic. Therefore, a limited number of LEDs (less than twenty) was preferred for the design of the calibration board.

The main advantage of using LEDs is that it produces bright light with low power consumption so it can be lit with a small and portable battery. Moreover, LEDs are very small (3mm in diameter) so their centre can be detected precisely, ensuring high accuracy in the calibration parameters estimation. First experiments were carried out with classic LEDs but showed that this source of light is not producing enough heat to be detected in infrared images. On the contrary, high power LEDs dissipate more energy, resulting in a significant amount of heat being

generated. Hence, they were deemed more appropriate for the purpose of creating multispectral features. The ones used in this work can dissipate up to 1W but the current is limited in order to avoid saturating the camera sensor and to keep small and sharp features. By arranging the lights in 6 series of 3 LEDs with a 120Ω resistor, each LED is dissipating around 0.4W which produces enough heat for them to be seen in the infrared images. To keep the setup as simple as possible, each block of LEDs is connected directly to the battery, in a parallel circuit. As a result, the total power required by the board amounts to 2.4W which is small enough to be powered by a small 12V battery.

A wooden board was selected to hold the lights because wood is neither reflective in the visible nor in the thermal domain. This reduces the risk of including noise when moving the pattern. It also provides a good contrast with the lights. Holes were drilled at specific locations to create the pattern and the LEDs were placed inside. The battery was then installed at the back of the board.

For visible cameras, these LEDs might be too bright and produce blurry blobs. In order to be able to detect and identify the calibration points accurately, the exposure time is diminished. This has the advantage of reducing the size of the blobs and improving their localisation, but it also avoids detecting unwanted features in the background as the rest of the image becomes dark (see Figure 3.3). A simple blob detection algorithm is then used to detect their position in the image. The high contrast in emissivity and temperature between the light and the board makes the detection straightforward in thermal images. However, if a person is holding the pattern, other bright features could appear in the image. Nevertheless, they can be filtered out according to their size and shape so only small and circular blobs are retained. Another solution consists in placing the board against a wall or on the ground and moving the cameras around. Once each point has been identified, its centre is estimated with sub-pixel accuracy by computing the



(a) visible (b) thermal

Figure 3.3: Calibration pattern seen by visible and thermal cameras.

centre of mass of the corresponding blob.

The calibration parameters are then estimated in a non-linear optimisation scheme, minimising the distance between the projection of each calibration point into the image and the corresponding detected feature [41]. The cost function of the least-squares optimisation problem can be expressed as:

$$\underset{K,\mathbf{D},R,\mathbf{t}}{\arg\min} \sum_{i=1}^{N} \sum_{j=1}^{M} \|\mathbf{x}_{ij} - P_j \mathbf{X}_i\|_2 \tag{3.13}$$

where $\mathbf{x}_{ij}$ is the i^{th} 2D point observed in the j^{th} image, $\mathbf{X}$ the corresponding 3D point expressed in the calibration pattern coordinate system. P, K, $\mathbf{D}$, R and $\mathbf{t}$ represent the projection matrix, intrinsic and extrinsic camera parameters, respectively (defined in (2.13), (2.15) and (2.16)).

The calibration toolbox provided by OpenCV is used to perform the optimisation. Based on [48], the algorithm takes as an input the location of the pattern's features in each image and the physical geometry of the pattern. After optimisation, it returns the calibration parameters. Additionally, OpenCV also provides a function to automatically rectify new images with the estimated calibration parameters.

3.4.2 Pattern Design

The calibration pattern has been designed to be easily identifiable and to use as few LEDs as possible. It is made of four corners (3 points each) shown with green and blue triangles in Figure 3.4, and a central pyramid (6 points) shown with red triangles in Figure 3.4. Corners spread features around the calibration board whereas the central pyramid gives an orientation to the pattern. As a consequence, whatever the orientation of the board or the cameras, the direction in which the pyramid is pointing can always be estimated.

To identify all the elements of the pattern, the mean of all LED positions is first computed. Features are then sorted out by their distance to the mean. This way, points belonging to the pyramid can easily be distinguished from the corner points. As the base of the pyramid is larger than its height, it is possible to discern the former from the two other edges of the external triangle. The top of the pyramid can then be identified as the vertex opposing the base. A white line segment is displayed in Figure 3.4 to show the orientation of the pyramid. Once the orientation has been determined, a new coordinate system is defined at the centre of the pyramid. All the features are then sorted according to the angle they form with the origin of the new coordinate system. This guarantees that the order of the features is always the same, regardless of the orientation of the pattern.

3.4.3 Experimental Validation

The proposed calibration pattern was tested with the multispectral stereo setup presented in Section 1.4.1. To do so, the calibration pattern was moved in front of the cameras and recorded with different positions and inclinations. Two methods are proposed to assess the quality of the calibration. The first one consists in analysing the reprojection errors and the second one compares the depth of the

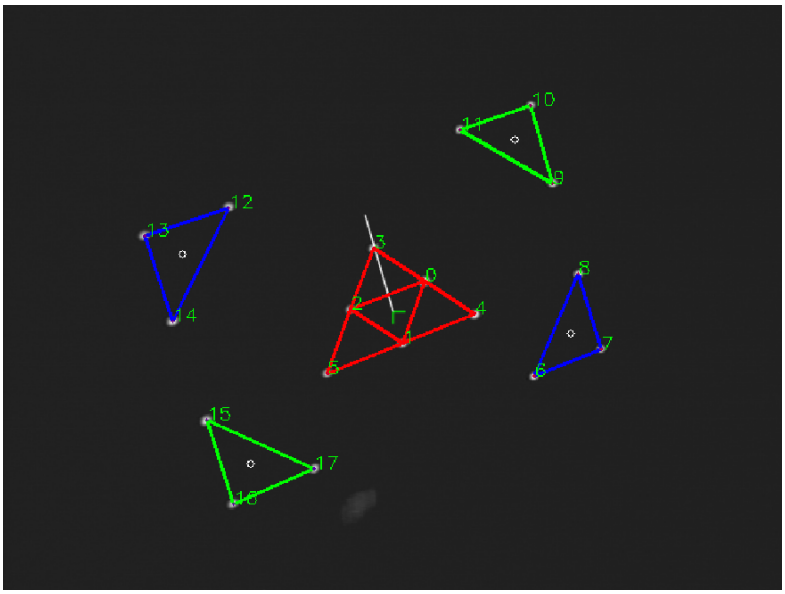


Figure 3.4: Automatic detection of the elements of the pattern.

triangulated features to a ground truth.

Calibration results

In order to prove the robustness of the pattern detection and calibration process, five datasets were collected. On average, the calibration pattern was detected in **84** visible images and **80** thermal images per dataset, corresponding to a total of 1512 visible and 1440 thermal features. These images were then used to estimate the intrinsic parameters of each camera, independently. The accuracy of the calibration is evaluated by computing the *Mean Reprojection Error* (MRE), which corresponds to the average reprojection error for all the images. Thus, it corresponds to the value of the cost function defined in (3.13), divided by the total number of observations. The MRE then gives an estimate of how well the calibration parameters fit the observations. For the sake of completeness, minimum and maximum reprojection errors are also computed.

As summarised in Table 3.1, monocular calibration produced average reprojec-

Table 3.1: Reprojection errors obtained for each monocular and stereo calibration. Minimum MRE/STDEV, minimum reprojection error and maximum reprojection error over all datasets are highlighted in bold.

dataset	A	B	C	D	E
camera	visible				
MRE	**0.226**	0.238	0.233	**0.226**	0.240
STDEV	0.022	0.029	**0.020**	0.021	0.027
MIN	0.205	0.214	0.210	0.187	**0.186**
MAX	0.248	0.253	0.258	0.260	**0.283**
camera	infrared				
MRE	**0.279**	0.281	0.285	0.292	0.358
STDEV	0.035	**0.032**	0.049	0.036	**0.032**
MIN	**0.232**	0.236	0.233	0.260	0.276
MAX	0.314	0.319	0.329	0.333	**0.423**
camera	stereo				
MRE	1.044	1.025	1.045	**1.011**	1.126
STDEV	0.240	**0.212**	0.286	0.261	0.286
MIN	0.805	0.794	**0.792**	0.812	0.890
MAX	1.541	**1.820**	1.751	1.393	1.778

tion errors of **0.23px** and **0.29px** over all calibration datasets, with the BlueFOX and Tau2 cameras respectively. This shows that thermal images produce larger errors and thus, less accurate calibration parameters. It can be explained by the fact that unlike visible light, LEDs heat the chessboard around them and produce blurrier features in the thermal domain. Nevertheless, the difference between visible and infrared MRE remains small and sub-pixel accuracy is achieved with both modalities. These results show a slight improvement in MRE compared to [43], where 0.35px was achieved for the infrared camera and 0.48px for the visible. Moreover, a more accurate estimation was obtained compared to [44], where MRE of 0.73px and 0.78px were recorded for the visible and the thermal camera, respectively.

Among the images selected for monocular calibration, around **63** images per dataset were common to both cameras. These images were thus utilised to refine

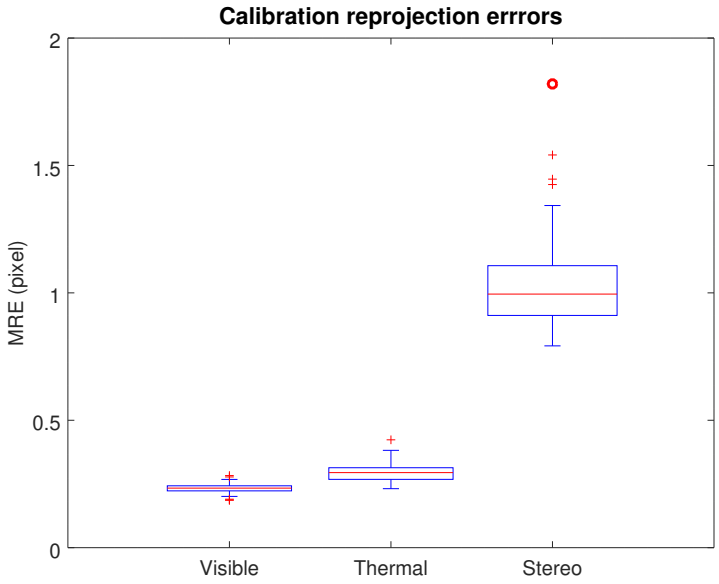


Figure 3.5: Reprojection errors boxplot for all datasets.

the intrinsic parameters of each camera and to estimate the extrinsic parameters of the stereo rig. It can be seen in Table 3.1 that stereo calibration is less precise than monocular calibration. This is expected because more parameters need to be estimated and more constraints arise from the fact that the same points are observed in two different images. Globally, the accuracy of the stereo calibration is equivalent to **1.05px** pixel. Here, the multispectral stereo calibration is more accurate than [43], where a 1.91px MRE was obtained. In [44], no results was provided for the stereo calibration. All reprojection errors were gathered in a boxplot (see Figure 3.5) to clearly visualise and compare the accuracy of each type of calibration.

Another way of checking the validity of the calibration process consists in analysing the epipolar lines in stereo rectified images. After calibration, new images of the calibration pattern were acquired with the same setup and were stereo rectified with the corresponding calibration parameters. As it can be seen in Fig-

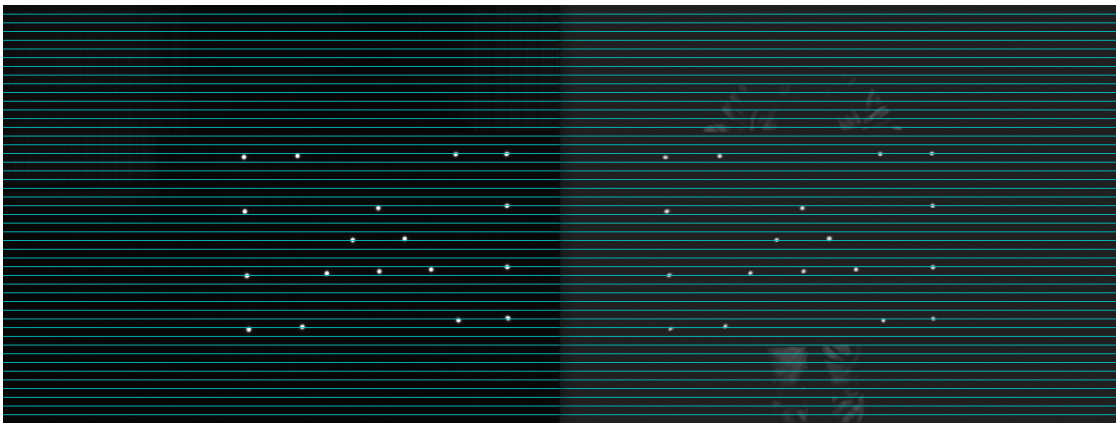


Figure 3.6: Stereo rectified images. Calibration points are located on the same row.

ure 3.6, distortions have been removed and all calibration points are now located on the same row. This means that the epipolar lines are horizontal and the images aligned, as in Section 3.2.

Finally, the distance between the two cameras was measured manually and was estimated at **51** cm. The stereo rig was designed so that both cameras face the same direction (no relative orientation) and their camera centres are aligned (only translated in the x-axis). The average translation and rotation vectors obtained from all calibrations are:

$$\mathbf{t} = \begin{bmatrix} 510 \pm 4mm \\ 2 \pm 3mm \\ 0 \pm 13mm \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} 8.4e^{-3} \pm 7.0e^{-3}rad \\ -1.3e^{-2} \pm 8.9e^{-3}rad \\ -1.0e^{-2} \pm 6.5e^{-4}rad \end{bmatrix}$$

This confirms that the extrinsic parameters of the stereo rig were also estimated correctly, because the baseline corresponds to the 51cm measured and the rotation vector is almost null.

3D reconstruction

To show the suitability of the calibration for 3D reconstruction, the calibration pattern was placed at different distances from the cameras, from 2m to 10m. Its depth was then recovered by triangulating the elements of the pattern detected in the stereo images and computing their average depth. Figure 3.7 shows that a maximum of **2cm** error was obtained when the board was located at 10m from the cameras. The smallest error, **3.6mm**, was obtained at a distance of 5m. It can be noticed that the minimum error was not obtained at the closest distance. This can be explained by the large baseline used and the level of precision at which features could be extracted. Indeed, when the cameras were only a few meters from the board, blobs of a few pixels were generated in the images. Even though their centres were estimated with sub-pixel accuracy, the more distant the cameras, the smaller the blobs and the more precise the feature extraction. On the contrary, when the cameras were located too far away, the size of the blobs became smaller than a pixel their centres could no longer be estimated precisely. Moreover, as the distance increases the disparity between stereo images decreases due to the formula in (3.4) and the noise generated in the feature extraction becomes more significant in the triangulation process.

Since the experiment was not reproduced, it is not possible to evaluate the repeatability of the calibration. However, the calibration was performed in a controlled environment and by looking at the precision at which the calibration parameters were estimated, it is highly probable that the repeatability of the calibration would be affected by the accuracy of the feature extraction algorithm. Nevertheless, it is clear that such calibration allows to triangulate stereo features with a few cm accuracy and that its precision tends to decrease with the distance.

Additionally, the experiment does not represent real driving conditions since it was performed in a lab. With real data, an important number of features would

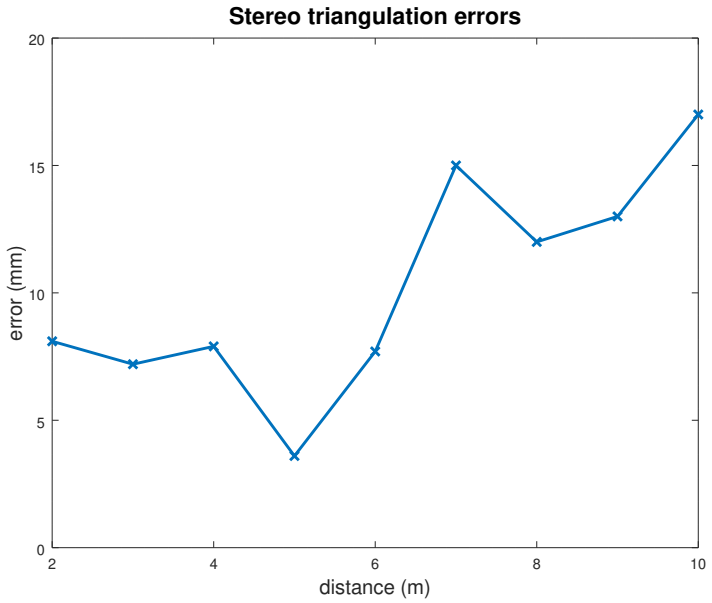


Figure 3.7: Depth errors depending on the distance where it was computed.

be located more than 10m away from the camera. However, it would be hard to repeat the experiment with larger distances since it was performed indoor and the pattern would become so small in the images that it would be almost impossible to distinguish the features due to the resolution of the images.

3.5 Conclusion

In this chapter, a new pattern designed to calibrate multispectral cameras was presented. It makes use of high-power LEDs to create small circular features that can be seen in both visible and infrared spectra. The board is easy to build, made of low-cost and low-power elements and thus, it is convenient and simple to use. The calibration process described consists of an automatic feature detection part which precisely localises the LED centres; and a second estimation part which utilises these points in a non-linear optimisation process to estimate both intrinsic

and extrinsic calibration parameters. The calibration of the visible-thermal stereo setup was achieved with a small reprojection error (**1.05px**) and its robustness was tested by reconstructing the 3D position of the calibration board. A maximum error of **2cm** was obtained when the board was positioned at 10m from the cameras. This shows the suitability of the calibration process and justifies its use for 3D reconstruction and visual odometry in the following chapters.

4 | Monocular visual odometry for visible and LWIR images

With projective geometry, it is possible to evaluate the attitude and position of a calibrated camera from the images generated. To do so, a monocular visual odometry algorithm is introduced. This technique is able to estimate the pose of a single moving camera, regardless of the nature of the images. First, the main components of the VO process are detailed, namely feature detection, feature matching and motion estimation. Then, a step selection algorithm is presented to ensure sufficient translation between consecutive frames and to avoid degenerate cases. The monocular VO framework is tested in its entirety with both visible and LWIR images. Two motion estimation methods are compared, 2D-to-2D estimation which uses only image features and 3D-to-2D estimation where both the structure of the scene and the image features are optimised. Finally, the benefit of multispectral imaging is demonstrated with the alternative use of both modalities on a dataset with challenging illumination conditions. Throughout this chapter, special consideration is given to avoid tuning the algorithm parameters for each modality.

4.1 Introduction

Monocular VO consists in estimating the 6 DoF ego-motion of a single moving camera. 2D features are detected in the different images acquired and matched to produce correspondences. The relative transformations between camera poses can then be recovered with 2D-to-2D motion estimation, by estimating and decomposing the essential matrix solely based on the correspondences in the images [7]. alternatively, 3D-to-2D motion estimation could be performed. It consists in triangulating 3D points based on previously known camera poses and reprojecting them in the latest image plane [49, 50]. Transformations between consecutive images can then be estimated with optimisation-based techniques by minimising some error metrics between the observed and reprojected features.

In such cases, two general approaches can be distinguished: *feature-based methods* which minimise the geometric error between an observed feature (obtained after matching) and its reprojected position in the image [13, 7]; and *direct methods* which minimise the photometric error between images, or in other words the difference in pixel intensities between local regions [51, 52]. Because pixel intensity is not an acceptable comparison metric for multispectral setups due to the different nature of the images, the use of feature-based methods was preferred for this work and will be detailed in the following sections.

4.2 Feature detection and matching

In order to find 2D corresponding features in several images, two main approaches can be considered:

- Tracking methods
- Feature-based matching

Feature-based matching is usually more robust to large geometric and spatial changes between consecutive images because features are described in a unique way with descriptors, but it is also more costly when the number of features increases. Tracking techniques are however relatively fast to compute, which make them more suitable for fast motion such as sharp turns or when driving over speed bumps [19].

In any case, a set of 2D distinctive image points is required to start with. Therefore, Section 4.2.1 first introduces the most common feature detection algorithms used to extract salient information from the images. Section 4.2.2 then describes the feature description/matching process and Section 4.2.3 focuses on tracking methods.

4.2.1 Feature detection

Feature detection consists in finding points of interest in an image. In order to easily identify them, these points need to be salient and highly discernible from their neighbourhood. Corners and small blobs offer the best localisation properties. Edges are also good image features because they have an important gradient and provide a good contrast along their principal axis, but an ambiguity remains along the edge. Some edge-based techniques have been developed in the recent years, but they are usually more complex, time-consuming, and still require the help of other point features [53, 54, 55].

Feature detectors are evaluated based on the following characteristics:

Localisation accuracy: it defines how precisely a feature can be localised. Nowadays, position can be refined to sub-pixel accuracy. For scale invariant and orientation invariant features, the accuracy with which scale and orientation are computed is also important.

Repeatability: it measures how likely features are to be detected again in other

images. In order to perform descriptor-based matching, each feature needs to be detected in the other images, otherwise it will be associated with a wrong counterpart. Thus, it is important to extract identical features during the detection step in order to maximise the number of correct matches during the matching part.

Computational efficiency: depending on the application (whether precision or real-time processing is more important), computation time could be a decisive factor. It directly describes how fast the detection can be achieved and reversibly how many features can be detected in a limited period of time. It is best to have as many features as possible because wrong matches or weak corners can always be discarded later, but motion cannot always be computed properly when the number of decent features is limited.

Robustness: it refers to the ability of a detector to extract features, regardless of the quality of the images (e.g. with noisy images). The more robust the detector, the less error will be made during the matching process.

Invariance: it is a property similar to robustness and describes how well a detector performs when the characteristics of the environment change. This could refer to illumination changes, or geometric changes (e.g. size, orientation) when the camera rotates or when it is moving closer/away from the feature.

The most common corner detectors in the literature are Harris [56] and Shi-Tomasi (Good Feature To Track)[57]. Even though many other detectors exist, these are widely used for VO [50]. Concerning blob detection, more advanced detectors are employed: FAST [58], SIFT [59], SURF [60]. They are more robust to geometric changes because they are scale and orientation invariant, as well as their respective descriptors.

4.2.2 Feature description and matching

Feature-based matching techniques are designed to detect 2D points in all the images, to identify them uniquely with descriptors (meant to describe the neighbourhood around the feature) and then to match these features across the images, based on the similarity of their descriptors.

Feature description

A descriptor is a vector with a reduced number of elements that describes a patch around the detected feature and that can easily be compared to other descriptors. For instance, it could contain the intensities of all pixels in the patch. This would however be terribly inefficient as it will include a lot of elements and would not compensate for illumination and geometric changes. Hence, more complex descriptors are required. They can be divided into two categories: binary and histogram-based descriptors. Binary descriptors such as BRIEF [61], or scale/rotation invariant ORB [62], BRISK [63] and FREAK [64] are designed to compute a binary version of the patch, which values can then be compared pairwise. Such descriptors are extremely fast to compute and compare, making them ideal for real-time applications and navigation purposes. Histogram-based descriptors on the other hand, divide the patch into several sub-regions and describe their aspect with histograms. SIFT descriptors for example, decompose the region in 8 sub-regions (4×4) and compute an histogram of 8 orientations for each one of them. These histograms are then combined together to form a 128-element vector which can then be compared to other descriptors [59]. Histogram-based descriptors are therefore more robust but require more processing power. To reduce the effect of illumination conditions and improve their robustness, histograms can also be normalised.

Feature matching

Two sets of features obtained from different images are matched by computing distances between their descriptors. For each feature in one set, matches are found by selecting the feature from the other set whose descriptor returns the smallest distance. Depending on the descriptor, different norms can be used for this distance computation. The L_2-norm, for example, is known to be a suitable measure for SIFT and SURF. The *Hamming* distance is however preferred for binary descriptors such as ORB or BRISK [50]. To test every feature, several approaches can be used. Brute-force consists in comparing one feature in an image with all features from the other images. This is an extremely time-consuming approach and not very efficient but the computation load can be reduced by limiting the search to specific features (e.g. features located on the epipolar line if it is known). K-nearest neighbour is a more efficient approach to compare only neighbouring features. With a K-dimensional tree structure, these features and descriptors can be accessed and compared rapidly. Once every feature has been tested, the distance obtained can be checked and only the matches with small distances are retained to avoid producing outliers.

4.2.3 Feature tracking

Unlike descriptor-based methods which require a costly matching process, tracking methods are directly estimating the position of a feature in the next image based on its *optical flow*. The main advantage of feature tracking is its speed, even when the number of features grows, which is an asset for navigation applications [65]. In the following sections, the optical flow equations are derived and the *Lucas-Kanade* method is presented.

Optical flow

Optical flow refers to the 2D apparent velocity of the objects in the image due to camera movements. It relates the motion between two frames at different time (t and $t + \Delta t$). Assuming that brightness is consistent between the images, a certain point in one image $\mathbf{p} = [x, y]^T$ will be shifted to another point $\mathbf{p'} = [x+\Delta x, y+\Delta y]^T$ in the next image, with $\Delta\mathbf{p} = [\Delta x, \Delta y]^T$ corresponding to the displacement along the x and y axes, respectively. However, the pixel intensities of $\mathbf{p}$ and $\mathbf{p}'$ remain the same:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{4.1}$$

This equation is called the *brightness consistency constraint*. For small motion, it can be approximated with Taylor series as follows:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \simeq I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{4.2}$$

from which the following optical flow equations are derived:

$$\begin{aligned} & \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \\ & \frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial t} = 0 \\ \Rightarrow \quad & \frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y = -\frac{\partial I}{\partial t} \end{aligned} \tag{4.3}$$

where (V_x, V_y) is the velocity or optical flow of $\mathbf{p}$. Solving these equations results in finding the velocity of $\mathbf{p}$ or the displacement corresponding to a new point $\mathbf{p}'$ where the brightness remains the same.

Lucas-Kanade method

The *Lucas-Kanade* method [65] estimates the position of a previously detected feature in a new image by solving the optical flow equations for each pixel in its neighbourhood. This makes the process more robust than solving it for a single point. Assuming the local flow to be constant in the surrounding area, a system of equations can be generated for the point $\mathbf{p} = [x, y]^T$:

$$\begin{aligned} \frac{\partial I}{\partial x}_{|\mathbf{p}_i=\mathbf{p}_1} V_x + \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_1} V_y &= -\frac{\partial I}{\partial t}_{|\mathbf{p}_i=\mathbf{p}_1} \\ \frac{\partial I}{\partial x}_{|\mathbf{p}=\mathbf{p}_2} V_x + \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_2} V_y &= -\frac{\partial I}{\partial t}_{|\mathbf{p}=\mathbf{p}_2} \\ &\vdots \\ \frac{\partial I}{\partial x}_{|\mathbf{p}_i=\mathbf{p}_N} V_x + \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_N} V_y &= -\frac{\partial I}{\partial t}_{|\mathbf{p}_i=\mathbf{p}_N} \end{aligned} \tag{4.4}$$

where $p_i, i \in [1, N]$ corresponds to the location of the i^{th} pixel in the neighbourhood. This system can be written in a matrix form:

$$A\mathbf{v} = \mathbf{b}$$

$$A = \begin{bmatrix} \frac{\partial I}{\partial x}_{|\mathbf{p}_i=\mathbf{p}_1} & \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_1} \\ \frac{\partial I}{\partial x}_{|\mathbf{p}_i=\mathbf{p}_2} & \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_2} \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}_{|\mathbf{p}_i=\mathbf{p}_N} & \frac{\partial I}{\partial y}_{|\mathbf{p}_i=\mathbf{p}_N} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{-\partial I}{\partial t}_{|\mathbf{p}_i=\mathbf{p}_1} \\ \frac{-\partial I}{\partial t}_{|\mathbf{p}_i=\mathbf{p}_2} \\ \vdots \\ \frac{-\partial I}{\partial t}_{|\mathbf{p}_i=\mathbf{p}_N} \end{bmatrix} \tag{4.5}$$

Because the number of pixels is larger than the number of parameters being estimated (2D velocity), the problem is overestimated and a least-squares approximation can be obtained by solving:

$$\begin{aligned} & A^T A\mathbf{v} = A^T\mathbf{b} \\ \Rightarrow \quad & \mathbf{v} = (A^T A)^{-1} A^T \mathbf{b} \end{aligned} \tag{4.6}$$

However, the differential equations presented in (4.3) only hold for small displacements, to refine a previous estimation for example. To guarantee a reliable tracking even when the images are shifting significantly, a pyramidal approach was used in [66]. A pyramid is built from the original image by reducing its resolution in half at each level. An initial optical flow is then estimated from the image at the top of the pyramid. This image possesses the smallest resolution of the pyramid which corresponds to $\frac{1}{2^N}$ the resolution of the original image, N being the height of the pyramid. Optical flow is then refined iteratively with images of higher resolution, until the lowest level is reached (full resolution). This method is more robust and does not usually necessitate to have many levels in the pyramid. Yet, a trade-off needs to be found to obtain the best results without preforming more calculation than necessary.

4.3 Epipolar Geometry

The epipolar geometry corresponds to the projective geometry that links two views together. It is intrinsic to these views as it only depends on the structure of the scene, the camera calibration parameters and the cameras relative pose. To find the relationship between two images, corresponding points of interest are detected and matched using one of the techniques presented in Section 4.2. With the addition of the 3D points corresponding to the physical representation of the observed features, a series of special geometric elements can be defined:

The baseline: it is the line segment connecting the two camera centres. It also corresponds to the relative translation between the two camera poses.

The epipole: it corresponds to the point where the image plane and the baseline intersect. Therefore, each view possesses one epipole. It can also be

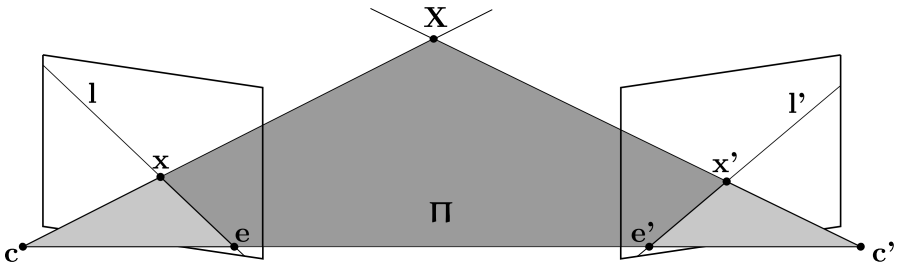


Figure 4.1: Depiction of the epipolar geometry between two views. ($\mathbf{x}$,$\mathbf{x}'$) represents the central projection of $\mathbf{X}$ through the camera centres $\mathbf{c}$ and $\mathbf{c}'$, respectively. The epipoles ($\mathbf{e}, \mathbf{e}'$) and epipolar lines ($\mathbf{l}$,$\mathbf{l}'$) can be seen in each image plane. $\mathbf{\Pi}$ represents the epipolar plane.

interpreted as the projection of one camera centre into the image plane of the other view. Epipoles can sometimes appear outside of the image, if the other view is not located in the FOV of the camera.

The epipolar plane: The plane formed by a 3D point and the two camera centres is called epipolar plane. It contains not only the 3D feature but also the projected points (x, x') in both image planes and the epipoles of each camera, thus its name.

The epipolar line: it corresponds to the line where the epipolar plane intersects an image plane. This line includes the epipole of a camera and the projection of any feature located on the epipolar plane.

As illustrated in Figure 4.1, the epipolar plane links both views, which means that for each feature in the left or right image, an epipolar plane can be defined. Therefore, the epipolar plane contains the feature, its actual 3D location and the baseline. The corresponding feature in the other image plane is inevitably located along the line where the epipolar plane and the image plane intersect.

These constraints implied by the epipolar geometry apply to any point or line located in the image planes. Therefore, it can be represented in a algebraic form with some specific matrices:

The fundamental matrix: represents the epipolar geometry in the image plane. The intrinsic parameters of the camera are not taken into account.

The essential matrix: represents the epipolar geometry of rectified images. The essential matrix relates normalised points, taking into account the intrinsic parameters of the cameras.

4.3.1 The Fundamental matrix

Definition

Given a pair of images, the *fundamental matrix* associates a point in one image plane to the corresponding epipolar line in the other image plane: $\mathbf{x} \mapsto \mathbf{l}'$. It can be noticed that this mapping is unidirectional and it is not possible to obtain a point from an epipolar line, the result would be another line. The mapping is thus projective.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = F \begin{bmatrix} u \\ v \\ \alpha \end{bmatrix} \tag{4.7}$$

The fundamental matrix is defined as the unique 3×3 homogeneous matrix of rank 2 which satisfies:

$$\mathbf{x}^T F \mathbf{x}' = 0 \tag{4.8}$$

where $\mathbf{x} \leftrightarrow \mathbf{x}'$ are corresponding points in two images. This equation is called the *correspondence condition* and translates the fact that corresponding points are located on epipolar lines. Indeed, using homogeneous coordinates, a point $\mathbf{x}'$ is located on a line $\mathbf{l}'$ only if their product is null:

$$\mathbf{x}'\mathbf{l}' = 0 \tag{4.9}$$

For a pair $\mathbf{x} \leftrightarrow \mathbf{x}'$, $\mathbf{l}'$ is related to $\mathbf{x}$ by the fundamental matrix:

$$\mathbf{l}' = F\mathbf{x} \tag{4.10}$$

Similarly, $\mathbf{x}'$ corresponds to $\mathbf{x}$ if $\mathbf{x}'^T F\mathbf{x} = 0$, or in other words, F is the fundamental matrix between two views if it satisfies $\mathbf{x}'^T F\mathbf{x} = 0$ for every correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$ between the image planes. This is an important result as it means that correspondences between images can be characterised without the use of projection matrices P. This also means that F can be computed from a minimum set of 7 matches. Several algorithms exist to estimate the parameters of F [67, 7, 68], the most popular probably being the 8-point algorithm [68].

Properties of F

Transpose

Let F be the fundamental matrix between two camera views which associates a point $\mathbf{x}$ to its corresponding epipolar line:

$$F\mathbf{x} = \mathbf{l}' \tag{4.11}$$

Then, F^T corresponds to the epipolar geometry in the opposite order:

$$F^T\mathbf{x}' = \mathbf{l} \tag{4.12}$$

Null space

The epipoles are spanning the null space of F. Indeed,

$$\mathbf{e}'^T\mathbf{l}' = 0 \Leftrightarrow \mathbf{e}'^T(F\mathbf{x}) = 0 \Leftrightarrow (\mathbf{e}'^T F)\mathbf{x} = 0, \quad \forall\mathbf{x} \tag{4.13}$$

$\mathbf{e}'$ is thus called the left null space of F. Similarly, $F\mathbf{e} = 0$ implies that $\mathbf{e}$ is the

right null space of F.

4.3.2 The Essential matrix

The *essential matrix* is derived from the fundamental matrix. It can be seen as the specialisation of the latter for normalised coordinates. Unlike the fundamental matrix which does not require any knowledge about the calibration parameters of the cameras, the essential matrix encapsulates these parameters. Thus, it possesses fewer degrees of freedom and additional properties.

Definition

As seen in Chapter 2, a point is projected in an image with the projection matrix:

$$\mathbf{x} = P\mathbf{X} \quad \Rightarrow \quad \mathbf{x} = K[R|\mathbf{t}]\mathbf{X} \tag{4.14}$$

If the intrinsic calibration matrix K is known, it can be removed from the projective transformation by multiplying the coordinates of $\mathbf{x}$ with the inverse of K:

$$\hat{\mathbf{x}} = K^{-1}\mathbf{x} \quad \Rightarrow \quad \hat{\mathbf{x}} = [R|\mathbf{t}]X \tag{4.15}$$

$\hat{\mathbf{x}}$ is called the normalized coordinate of $\mathbf{x}$ and the corresponding projection matrix becomes $\hat{P} = [R|\mathbf{t}]$, which depends entirely on the extrinsic parameters of the camera. Any mention of the intrinsic parameters has disappeared. $\hat{P}$ is called *normalised projection matrix*. As stated earlier, the essential matrix translates the epipolar geometry between two normalised views. Therefore, in the same way the fundamental matrix associates an epipolar line to a point, the essential matrix associates an epipolar line to a normalised point:

$$E\hat{\mathbf{x}} = \hat{\mathbf{l}}' \tag{4.16}$$

Different techniques could be utilised to compute the essential matrix. For instance, it can be estimated from the fundamental matrix, or by using normalised features with methods such as the *normalised 8-points* [69, 7] or *5-points* algorithms [9].

Properties of E

Structure

E possesses a specific form:

$$E = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = [t]_\times R \tag{4.17}$$

where $[.]_\times$ denotes a skew symmetric matrix.

Epipolar constraint

Like F, E satisfies:

$$\begin{aligned} \hat{\mathbf{x}}'^T E \hat{\mathbf{x}} &= 0 \\ \mathbf{x}'^T K^{-T} E K^{-1} \mathbf{x} &= 0 \\ \mathbf{x}'^T F \mathbf{x} &= 0 \end{aligned} \tag{4.18}$$

from which it can be derived:

$$E = K'^T F K \tag{4.19}$$

Singular values

The Essential matrix possesses three singular values. Two of them are equal and non-zero while the third one is zero.

Degrees of freedom

As mentioned before, the essential matrix contains information about the extrinsic parameters. This means that E is closely related to the relative rotation

R and translation $\mathbf{t}$ between the views. It nevertheless possesses only 5 degrees of freedom: 3 for the rotation, 3 for the translation, but one degree less due to the scale ambiguity.

4.4 2D-to-2D Motion estimation

Due to the nature of the essential matrix it is possible to retrieve the relative rotation and translation between two views by *Singular Value Decomposition (SVD)*. E can be decomposed as follows:

$$E = U\ diag(\lambda, \lambda, 0)\ V^T = [\mathbf{t}]_\times R \tag{4.20}$$

where $U, V \in \mathrm{SO}(3)$ and λ is the non-zero singular value of E.

Because $[\mathbf{t}]_\times$ is a skew-symmetric matrix it can be decomposed as:

$$[\mathbf{t}]_\times = kUZU^T, \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.21}$$

where k corresponds to the scaling factor.

For the sake of simplicity and because it cannot be estimated directly from the essential matrix, the scale can be ignored by selecting the singular values as 1. The resulting translation is then estimated up to a scale. As a result, Z can now be expressed up to a sign as:

$$Z \pm\ diag(1, 1, 0)W, \quad W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

where W is orthogonal matrix.

(4.21) and (4.22) shows that any essential matrix can be decomposed with SVD, comprising two equal singular values. Reciprocally, any matrix possessing two equal singular values is an essential matrix. Therefore, E can be decomposed as:

$$\begin{aligned} E = [\mathbf{t}]_\times R = U\ diag(1,1,0)\ (WU^T R) \\ [\mathbf{t}]_\times = UZU^T \quad R = UW^T V^T \end{aligned} \tag{4.23}$$

It can be noted that matrix UZU^T has a Frobenius norm of $\sqrt{2}$ [7], which means that the translation vector $\mathbf{t}$ is a unit vector. Assuming that the first view has the projection matrix $P_1 = [I_{3\times3}|\mathbf{O}]$, it is then possible to retrieve the camera matrix of the second view $P_2 = [R|\mathbf{t}]$ using the SVD decomposition. But (4.23) is not the only decomposition possible. Since the singular values are equal and the sign of Z cannot be determined, four different solutions can be obtained:

$$\begin{aligned} P_2 = [UWV^T| + \mathbf{u}_3] \quad P_2 = [UWV^T| - \mathbf{u}_3] \\ P_2 = [UW^TV^T| + \mathbf{u}_3] \quad P_2 = [UW^TV^T| - \mathbf{u}_3] \end{aligned} \tag{4.24}$$

It can be noticed that the difference between the first and second line is the sign of the translation vector. This directly affects the positioning of each camera with respect to the other. The difference between the right and left equations on the other hand is characterised by a change in rotation. It corresponds to a 180° rotation around the baseline. The geometric interpretation of these four solutions is illustrated in Figure 4.2. From the diagram, it is clear that there is only one solution where the 3D point appears in front of both cameras. This fact is used as a criterion to remove the ambiguity and select the only solution physically valid. When the R and $\mathbf{t}$ have been estimated, the features are triangulated and the solution where they all end up in front of both cameras is chosen. In fact, only one point is necessary to determine which one of the four options is the right one.

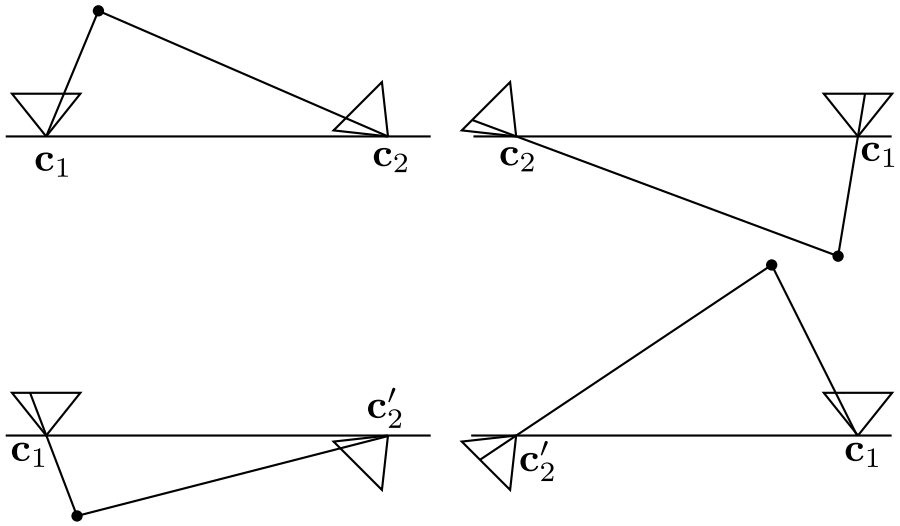


Figure 4.2: The four possible solutions to the essential matrix decomposition problem expressed in (4.24). The top-left diagram corresponds to the true solution, the right diagrams have the baseline inverted ($-\mathbf{u}_3$) and the bottom ones have the camera $\mathbf{c}_2$ rotated by 180°.

4.5 Scale ambiguity

Due to the nature of projective geometry, monocular vision systems suffer from a scale ambiguity. At least two views are necessary to reconstruct the 3D position of a 2D feature, but because depth information is lost during the projection, an overall ambiguity remains. Indeed, the position of 3D points and camera poses can be estimated relative to a common frame (usually the first camera pose). However, as it was explained in the previous section, the essential matrix is missing a DoF and the norm of the translation vector cannot be estimated from 2D-to-2D correspondences only. Thus, the absolute distance to the origin is unknown as shown in Figure 4.3. This global scale factor can be recovered with a prior knowledge of the system (e.g. stereovision) or information about the real world. For instance, GPS measurements could be used during the initial stage, to estimate the distance travelled and thus provide a metric to initialise the system. Alternatively,

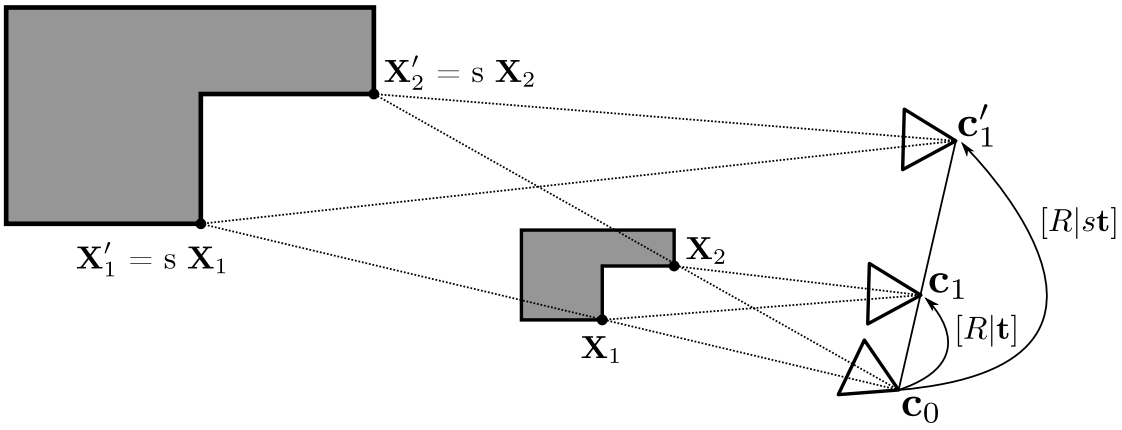


Figure 4.3: Scale ambiguity generated when triangulating 2D points from multiple views.

speedometers could be employed instead of GPS to obtain a more precise velocity measurement. Knowing the size of specific objects or location of certain markers can also be used to determine the global scale [70]. This is especially the case for object pose estimation where a model is available and its dimensions are known [70, 55]. In [71] and [72], the authors take advantage of the fixed position of the camera on board of a car, and especially its height, to recover the depth of ground features.

In recent years, with the raise of small autonomous systems (ground robots and drones), a need for small lightweight sensors has appeared and a whole new research area has emerge: *Visual-Inertial Navigation Systems* (VINS). VINS consists of fusing visual information (monocular or stereo) with *Inertial Measurement Unit* (IMU). By taking advantage of the accuracy of image-based localisation, VO trajectories can be used to correct the drifty nature of inertial sensors. On the other hand, IMUs provide world measurement such as gravity and other accelerations applied to the vehicle which can then be used to recover the full scale. This combination has become really popular and a lot of different ways to fuse data have been tested, from filters[73, 16] to non-linear optimisation [74, 10], tightly

[16, 10] or loosely [75, 76] coupled. VINS for multispectral navigation is studied in more details Chapter 6.

However, when motion is estimated from two views only, it is impossible to estimate the scale factor. For example, in Section 4.4 the relative transformation between two camera poses is computed from the essential matrix and a normalised translation is obtained. But due on the varying speed of the vehicle, the magnitude is likely to be different from 1. Assuming that images are acquired at regular intervals, when the speed remains constant, the distance between each frame is identical and all resulting translation vectors should have the same norm. When the speed changes however, the magnitude of the resulting translation vectors should also vary for a constant time interval. And accumulating normalised transformations obtained from the 2D-to-2D motion estimation cannot take this into consideration (see Section 4.7).

Even if the global scale cannot be recovered, it is still possible to compute relative scale changes. To estimate the relative scale variations between several camera poses, more views are necessary. For instance, in [77] a closed-form method with three views is proposed. It has the advantage of being much faster than iterative optimisation, but it is also less precise. Three views is the minimum required, more could be employed to obtain a better accuracy. To estimate the pose of a new frame relative to previous poses, a wide range of *Perspective N Points* (PnP) algorithms exists, such as P3P [78, 79] or EPnP [80]. These algorithms are based on 3D-to-2D correspondences. First, features are triangulated from already estimated poses and a 3D point cloud is generated. Then, the last camera pose is computed from the matched features (2D) in the image and the 3D corresponding point cloud. This is usually less precise than pure 2D-to-2D motion estimation but allows the local scale to be estimated [3]. Indeed, errors in previous camera pose estimations are propagated to the 3D points through the triangulation process,

which are subsequently accumulated in the last pose estimate [81].

A convenient solution to avoid rapid errors accumulation is to refine the structure (position of the 3D points) and the camera poses at the same time. This process is called *Bundle Adjustment* (BA) and is commonly used for VO and SLAM [13]. BA is computationally heavier than the other solutions proposed in this section, and its complexity increases drastically with the number of points and camera poses. But it also provides a much more accurate motion estimation.

4.6 3D-to-2D Motion Estimation

Even though physical measurements are required to assess the overall scale factor, precision can be improved and local scale can be recovered from a certain set of images with the use of a bundle adjustment framework. Furthermore, a certain distance (translation) between the camera frames is required in order to triangulate the 3D features accurately. However, selecting a constant framerate does not guarantee that this condition will be satisfied when the camera is moving slowly or when it is stationary. In the following sections, a step selection method is proposed and the bundle adjustment problem is presented.

4.6.1 Keyframe selection

When motion is estimated from a set of 2D correspondences like in Section 4.4, certain degenerate configurations can arise. This is the case when all points lie on a critical surface (ruled quadratic surface or plane) or when the camera undergoes a pure rotation (no translation) [7]. The former is called *degenerate structure* and the latter *degenerate motion*. Therefore, to avoid these situations, it is important to carefully select the adequate frames. Instead of processing all the frames available, each one of them is tested and only those providing satisfactory conditions are

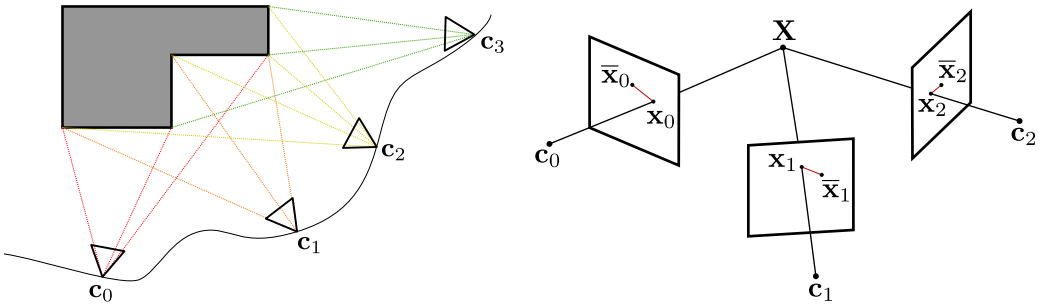
retained for motion estimation. These special images are called *keyframes.* In the event of degenerate structure, there is not much that can be done. If all features are located in the same plane, one can discard the current frame and wait for the next one where features will be spread. Nevertheless, a large shift cannot be tolerated either, because this would create a risk of loosing track of the features and ending up with no correspondences at all. On the other hand, if the camera is stationary or moves particularly slowly (degenerate motion) it is not possible to detect how much the camera moved compared to the last keyframe. Thus, in a similar way to [82], an image is added to the system only if it provides sufficient translation. But keyframes can also be included in case of important rotation in order to keep track of a sufficient number of points in subsequent images.

To evaluate how much translation separates two keyframes, the notion of *parallax* is employed as described in [83]. Parallax is the displacement observed by two parallel cameras at two different viewpoints. If the camera is rotating during the image acquisition, it will produce some shift in the images which needs to be compensated in order to observe the parallax. To do so, when a new image becomes available, its relative orientation with respect to the last keyframe is computed by estimating and decomposing the essential matrix. As illustrated in Figure 4.4, applying the inverse rotation to the tracked features removes the displacement due to the rotation, making the parallax measurable. The new frame is then accepted if the mean of all parallax vectors is larger than a certain threshold or if the initial shift (with the effect of rotation) becomes too significant. For both the visible and the infrared cameras used in this work, a minimum parallax threshold of 10px and a maximum shift of 50px have been empirically selected.

Figure 4.4: Rotation compensation between consecutive frames. Red lines show the correspondences between the frames and green lines show the parallax after removing the effects due the camera rotation.

4.6.2 Windowed Bundle Adjustement

Bundle Adjustment (BA) refers to the optimisation problem which jointly estimates 3D features from the scene and camera poses. From a geometric point of view, it attempts to find the best parameters which "adjust" bundles of rays between the 3D points and the camera centres (see Figure 4.5a). From a mathematical perspective, BA is a minimisation process which consists in finding the set of parameters that minimises a certain cost function, usually the Sum of Squared reprojection Errors (SSE). As illustrated in Figure 4.5b, each error is equal to the Euclidean distance between the 2D features $\mathbf{x}_{ij}$ detected in a certain image and the position where its corresponding 3D point is reprojected in that image ($\overline{\mathbf{x}}_{ij}$). As mentioned previously, BA parameters are made of 3D vectors representing the physical location of the features expressed in a common reference frame and the 3D position and orientation of each camera pose. They form a high dimensional non-linear space where no simple global representation exits. Points can be expressed with either Euclidean or homogeneous coordinates and rotations can be described with any representation presented in Section 2.3. But each parametrisation leads to specific constraints and singularities, as discussed previously. To improve numerical stability, iterative optimisation methods are used and combined with local parametrisations. Local parametrisations are defined and valid around the current state. They are chosen to be as uniform and linear as possible in its vicinity. For example, rotation matrices can be used to represent the global orientation but only small perturbations $\delta\mathbf{R}$, represented as 3D vectors (angle-axis), are considered at each iteration. The small increment is then converted with the exponential map and used to update the state $R' \rightarrow R + \exp(\delta\mathbf{R})$.



(a) Relationship between 3D points and cameras. Rays from different cameras are represented with different colours.

(b) Single point reprojected in several images corresponding to different poses. The geometric error is shown in red.

Figure 4.5: Illustration of the general bundle adjustment problem.

Problem formulation

Having observed a certain number of features in several images, an obvious choice to estimate the different parameters of the model would be the use of non-linear least-squares optimisation. Considering a vector $\mathbf{S}$ containing all the parameters, the 2D position of a feature i in a specific image j can be predicted by the function $\mathbf{x}_{ij} = \pi(\mathbf{S})$:

$$\begin{aligned} &\mathbf{x}_{ij} : \mathbb{R}^{3\times N+6\times M} \mapsto \mathbb{R}^2 \qquad i \in [1, N], j \in [1, M] \\ &\mathbf{x}_{ij} = \pi(\mathbf{S}) = P_j \mathbf{X}_i \end{aligned} \tag{4.25}$$

where $\mathbf{x}_{ij}$ is the predicted point, P_j the projection matrix for the image in which the point is reprojected (depends on the camera pose and calibration parameters). $\mathbf{X}_i$ is the 3D position of the i^{th} feature in the world coordinate frame. The least-squares cost function is defined as the SSE for all points in each camera:

$$f(\mathbf{S}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \Delta \mathbf{x}_{ij}^T \Delta \mathbf{x}_{ij}, \qquad \Delta \mathbf{x}_{ij} = \overline{\mathbf{x}}_{ij} - \mathbf{x}_{ij} \tag{4.26}$$

But not all features are the same. Some are sharp corners that can be localised precisely in the image whereas others are blurrier and less precise. To take this

factor into account, one can use the weighted least-squares cost function:

$$f(\mathbf{S}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\Delta\mathbf{x}_{ij}^{T}W_{ij}\Delta\mathbf{x}_{ij} \tag{4.27}$$

where W_{ij} is a symmetric positive definite matrix. For this least-squares formulation to have a meaningful statistical interpretation, W_{ij} should represent the inverse covariance of the observed feature $\overline{\mathbf{x}}_{ij}$ [13]. It then coincides with the log-likelihood of $\overline{\mathbf{x}}_{ij}$. However, SSE is sensitive to outliers due to the nature of the square function and a single wrong match can have a significant impact on the rest of the estimation. To avoid this, a Maximum-Likelihood estimator function $\rho_i(\mathbf{x})$ is added to the formulation [84]. This function is chosen with a small tail so it reduces the effect of features producing large errors (see Figure 4.6). To do so, ρ_i needs to be an increasing function with $\rho_i(0) = 0$ and $\frac{d\rho_i}{d\mathbf{x}}(0) = 1$. SSE for example satisfies these criteria as it corresponds to $\rho_i(\mathbf{x}) = \mathbf{x}$, but more robust loss functions could be employed instead [84, 85]. The BA formulation becomes:

$$f(\mathbf{S}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\rho_i(\Delta\mathbf{x}_{ij}^{T}W_{ij}\Delta\mathbf{x}_{ij}) \tag{4.28}$$

Obviously, the main drawback of BA techniques is the number of parameters which grows every time a new keyframe is added or when new features are detected. This makes it impossible to use for real-time navigation when the framerate is high (more than 10 Hz) and the amount of points is significant (several hundreds). Hopefully, it does not have to be applied to all frames and points. Instead, it could be used locally with a limited number of parameters to refine the previously estimated structure and camera path [86, 87]. This method is called *windowed bundle adjustment* (WBA) or *local bundle adjustment* and consists in optimising only the latest camera poses, in a certain window, and the corresponding features that have been observed in that interval.

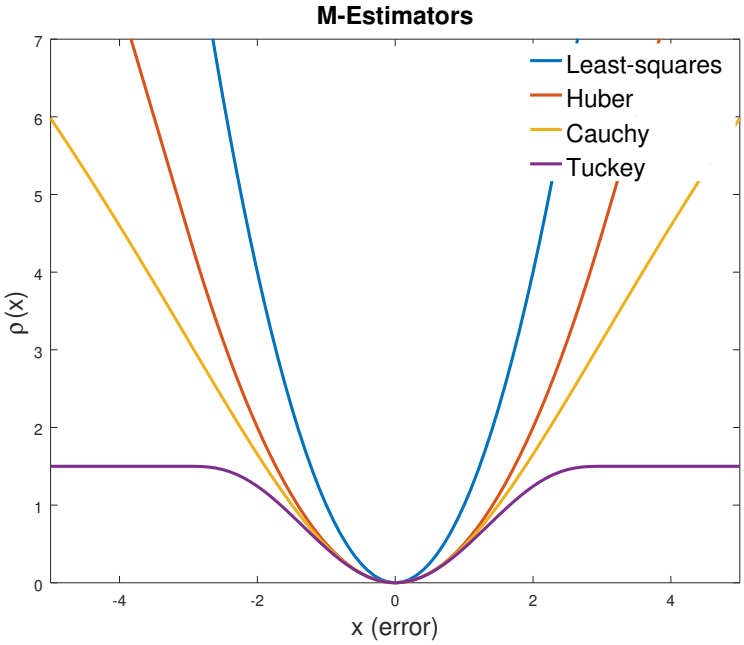


Figure 4.6: Example of loss functions.

4.7 Monocular Visual Odometry

Making use of the step selection process and windowed bundle adjustment described previously, a monocular VO framework for both infrared and visible images is presented (see Figure 4.7). Features are registered and tracked over a certain window with a pyramidal *Lucas-Kanade* algorithm. The choice of feature tracking over feature matching was driven by the speed of the algorithm and its consistency between modalities. Because it is fast, feature tracking can deal with a greater number of features from which robust motion estimation can be performed. It also does not require to tune certain thresholds and parameters for each spectrum, which is usually the case when matching features, because the quality of the descriptors is directly affected by the nature of the images [88]. It is nevertheless affected by the texture and contrast of the images.

When a new image becomes available, a set of new features is detected in the

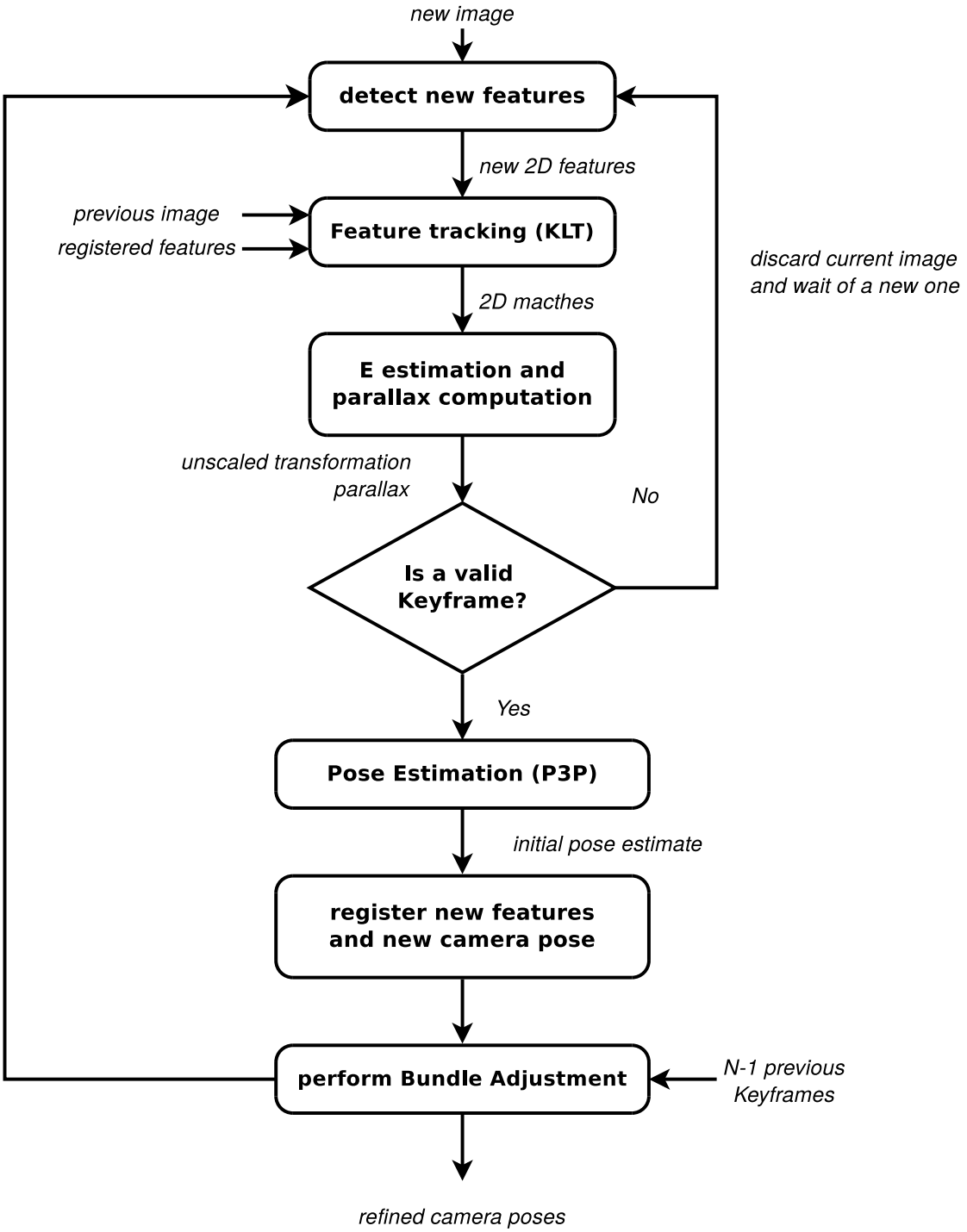


Figure 4.7: Flowchart of the monocular VO algorithm.

Algorithm 1 Monocular VO framework

Input:
P_{tot}: list of estimated camera poses
I_{tot}: list of acquired images
F_{reg}: list of registered features
Output:
P_{tot}: set of refined camera poses

while new image is available **do**
  /** *feature detection and tracking* **/
  $I_{new} \leftarrow$ retrieveLatestImage()
  $F_{new} \leftarrow$ featureDetection(I_{new})
  $M_{reg} \leftarrow$ lucasKanadeTracking(F_{reg},I_{new})
  $M_{new} \leftarrow$ lucasKanadeTracking(F_{new},lastImage(I_{tot}))
  /** *Epipolar geometry and parallax estimation* **/
  $E \leftarrow$ computeEssentialMatrix(M_{reg},M_{new})
  inliers $\leftarrow$ outlierRejection(E,M_{reg},M_{new})
  $p \leftarrow$ computeParallax(inliers)
  /** *Keyframe selection* **/
  if $p >$ MIN_PARALLAX **or** computeMeanShift(M_{reg}) $>$ MAX_SHIFT **then**
    /** *P3P* **/
    $Pt_{3D} \leftarrow$ triangulatefromPoses(P_{tot},F_{reg})
    $P_{new} \leftarrow$ perspective3Points(Pt_{3D},inliers)
    /** *Keyframe update* **/
    $F_{reg} \leftarrow F_{reg}+$ inliers
    $P_{tot} \leftarrow P_{tot} + P_{new}$
    $I_{tot} \leftarrow I_{tot} + I_{new}$
  else
    wait for next image
  end if
  /** *Bundle Adjustment* **/
  F_{win},$P_{win} \leftarrow$ extractSlidingWindow(F_{reg},P_{tot},WINDOW_SIZE)
  runBundleAdjustment(F_{win},P_{win})
  $P_{tot} \leftarrow$ updatePoses(P_{win},P_{tot})
end while

previous image and added to the existing set of registered features. This guarantees that a sufficient number of points are tracked throughout the window, despite some of them being discarded or leaving the image. To ensure a robust tracking, the features that do not satisfy the epipolar constraint are removed. The remaining points are then triangulated from the $N-1$ frames in the window. From the point cloud generated, the new camera pose is estimated with the P3P method [79], employed in a RANSAC framework to robustify the estimation. For more information about the RANSAC outlier rejection scheme, the reader may refer to Appendix B. Finally, BA is applied to the window in order to refine the structure of the scene and the camera pose estimation. The points that have not been triangulated properly (do not appear in front of the camera) or which have been labelled outliers during the RANSAC scheme are not used in the BA optimisation. A summary of the algorithm is presented in Figure 4.7 and its corresponding pseudo-code can be found in Algorithm 1.

4.7.1 Experimental validation

The performance of monocular VO for both thermal and visible modalities is evaluated on several datasets collected with the car setup presented in Section 1.4.1. Camera calibration parameters were obtained from the multispectral calibration algorithm described in Chapter 3. To provide a meaningful analysis, three different methods are compared; a simple 2D-to-2D motion estimation; camera pose estimation using only P3P; and the WBA algorithm presented in the previous section. They are compared to the accurate pose estimate provided by the Xsens MTi-G GNSS which combines inertial data and GPS coordinates. As explained in Section 1.4.1, all errors are computed with respect to these INS/GNSS positions since they are considered ground truth.

Errors are evaluated by computing the Euclidean distance between each es-

timated camera position and the closest GPS measurement in time. Because of the error accumulation, two types of errors are considered. Absolute errors are computed for the current camera pose, which corresponds to the accumulation of all previous estimates, and relative errors which corresponds to the difference in position of the relative transformation between two camera poses. This way, the accuracy of the method can be evaluated at a specific point in time regardless of previous estimations. These results are summarised in Table 4.1.

The initial orientation (roll and pitch angles) is estimated for all datasets from the acceleration measurements and the value of the gravity vector at the location of the experiment [1]. This requires the car to be stationary and located on a flat surface. Hence, it is performed beforehand when the cameras are mounted on the car as this orientation is unlikely to change much once the cameras are fixed. The initial pose can however be refined at the start of each dataset if the car is not moving. However, because the gravity vector is measured on the vertical axis, it is not possible to define the direction of travel and a certain ambiguity concerning the yaw angle remains. This problem is tackled by aligning the trajectory obtained from GPS with the one obtained from VO for the first few frames only. Concerning the initial position, it is set to zero. This means that the first camera position defines the origin of the world coordinate frame. The scale ambiguity is not addressed here because, as explained in Section 4.5, there are several ways to solve it. The focus of this chapter is to show the suitability of feature tracking and bundle adjustment for both visible and infrared images. Therefore, the global scale is estimated over the first 10 frames, by computing the ratio between the 2D translation vector obtained from GPS measurements and the 2D translation vector obtained from BA. This scale is then applied to the whole trajectory estimate.

The trajectories recorded combine mostly straight lines and 90° turns, where the car is slowing down before each turn and accelerating afterwards. The first and

Table 4.1: Absolute and relative errors achieved with each techniques.

	Visible			Infrared		
	2D-2D	**P3P**	**WBA**	**2D-2D**	**P3P**	**WBA**
Distance of sequence 1	133m					
mean absolute error (m)	4.88	19.07	**0.66**	2.90	5.39	1.00
std absolute error (m)	4.08	16.62	**0.39**	2.17	5.27	0.59
mean relative error (m)	0.79	1.28	0.09	0.73	0.49	**0.03**
std relative error (m)	0.62	1.13	0.05	0.63	0.39	**0.02**
Distance of sequence 2	511m					
mean absolute error (m)	12.42	73.30	9.01	13.48	122.35	**6.38**
std absolute error (m)	7.52	40.84	7.42	7.83	83.29	**5.81**
mean relative error (m)	0.38	0.62	**0.10**	0.39	1.24	0.16
std relative error (m)	0.29	0.54	**0.08**	0.25	1.31	0.14
Distance of sequence 3	581m					
mean absolute error (m)	33.72	66.88	15.04	23.73	96.98	**10.13**
std absolute error (m)	32.91	74.03	8.14	13.78	47.43	**7.29**
mean relative error (m)	0.49	0.83	0.21	0.54	0.98	**0.20**
std relative error (m)	0.34	0.78	**0.13**	0.39	0.62	0.18

most simple trajectory is made of only one turn and two straight lines whereas the other trajectories are longer and comprise more turns in several directions, making them more complex and challenging.

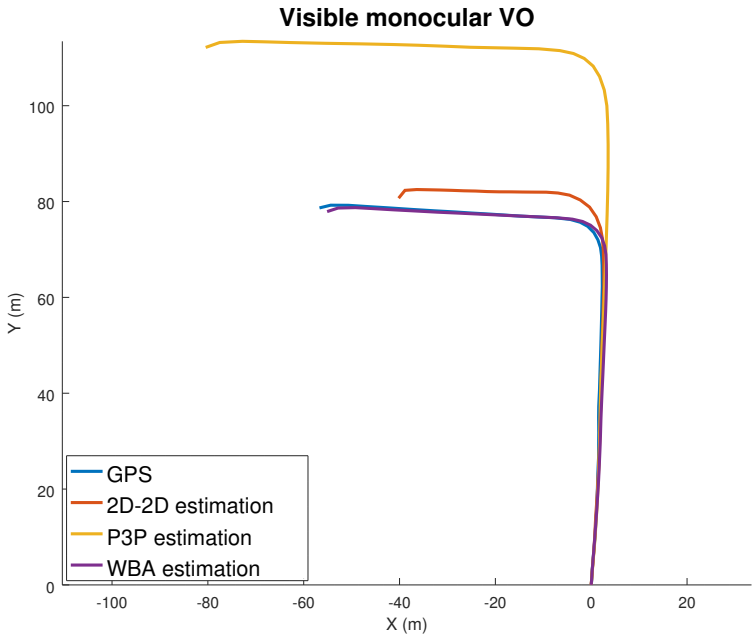
Motion Estimation

Even if the global scale is missing, Figures 4.8 to 4.10 show how local scale can be recovered from multiple frames and how the estimation is improved compared to frame-to-frame motion estimation. Orange trajectories represent 2D-to-2D estimation. As explained in Section 4.3, it is computed by simply decomposing the essential matrix between two consecutive frames and produces unit translation vectors. Even though keyframes are selected so the translation is similar between consecutive frames, it is clear from Figure 4.10 that 2D-to-2D estimation does not reflect changes in speed as the trajectory appear distorted. Moreover, Table 4.1

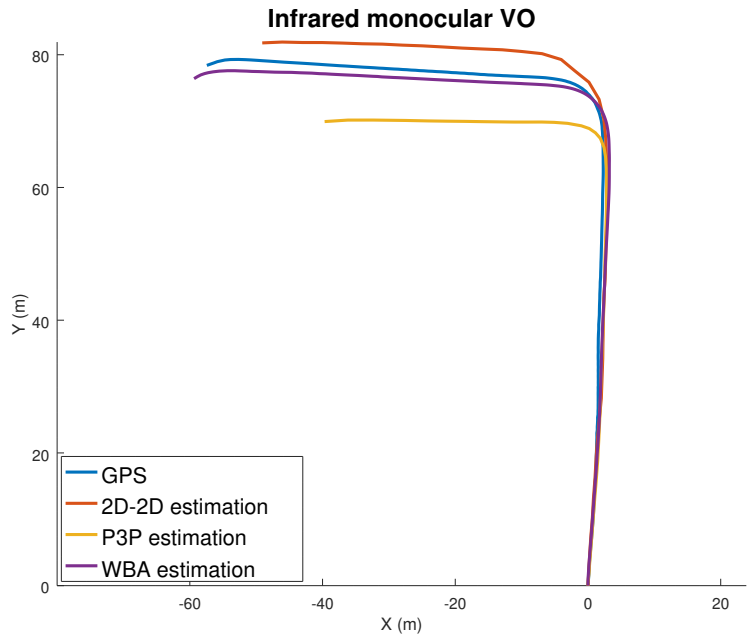
and Figure 4.11 show that even though this technique does not seems to perform a lot worse than WBA, more relative errors are produced. However, when these relative errors do not compensate each other, an important drift is produced as seen in Figure 4.10.

Yellow trajectories on the other hand, were obtained by estimating the new camera pose with P3P, based on the features triangulated in the local window, as described in Section 4.7. When computing the structure over several frames, the local scale can then be estimated. However, the graphs in Figure 4.11 show that the P3P technique drifts rapidly and creates the largest relative errors on all datasets. This leads to increasing errors and inconsistent motion estimation. Thus, the P3P estimate needs to be corrected in order to contain the drift. This is exactly the role of the WBA algorithm which was used to produce the purple trajectories. As it can be seen in all figures, this approach helps refining the camera pose estimate and prevents the errors to accumulate significantly. This is reflected in Figure 4.11 where the errors stay low compared to other techniques and do not produce any peaks. The trajectory then appears much closer to the ground truth in Figure 4.8-4.10. For both P3P and WBA methods, a sliding window containing a total of 7 frames was used. 3D points were obtained by triangulating the 2D image features from the first 6 frames. P3P then provided an estimate of the last camera pose, whereas WBA optimised the last 3 poses, creating a more robust estimation. Concerning the WBA algorithm, the first 4 frames of the window were kept fixed to solved the gauge freedom problem [13] and avoid drifts.

From these results, it can be concluded that decomposing the essential matrix is not sufficient to obtain an accurate motion estimation because of the scale ambiguity. P3P is partially solving the problem as it estimates the relative changes between several frames but it suffers from severe drift. WBA on the other hand, is able to reduce the drift and generates more precise trajectories. However, it

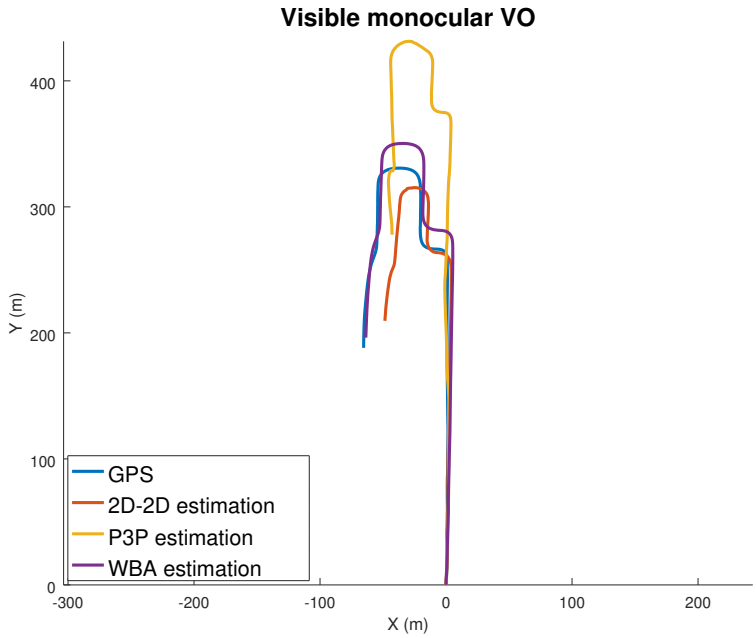


(a) Visible images

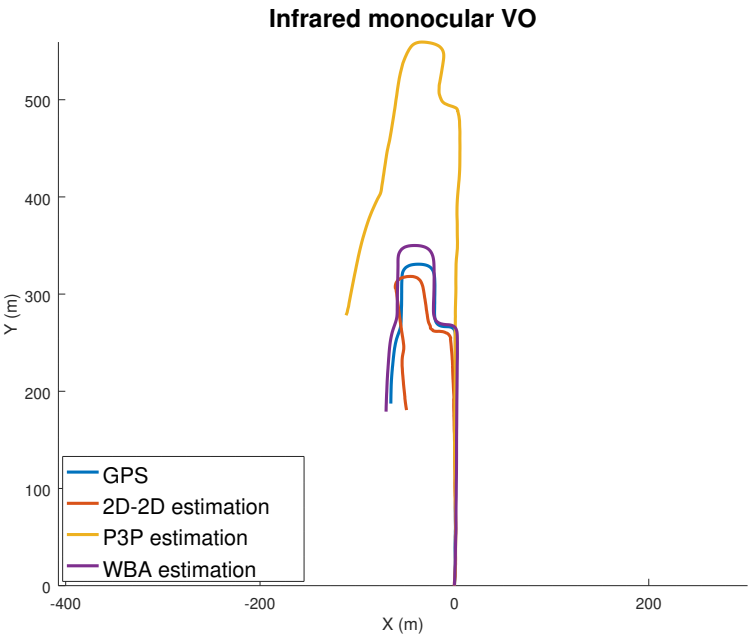


(b) Thermal images

Figure 4.8: Trajectory estimation on Sequence 1.

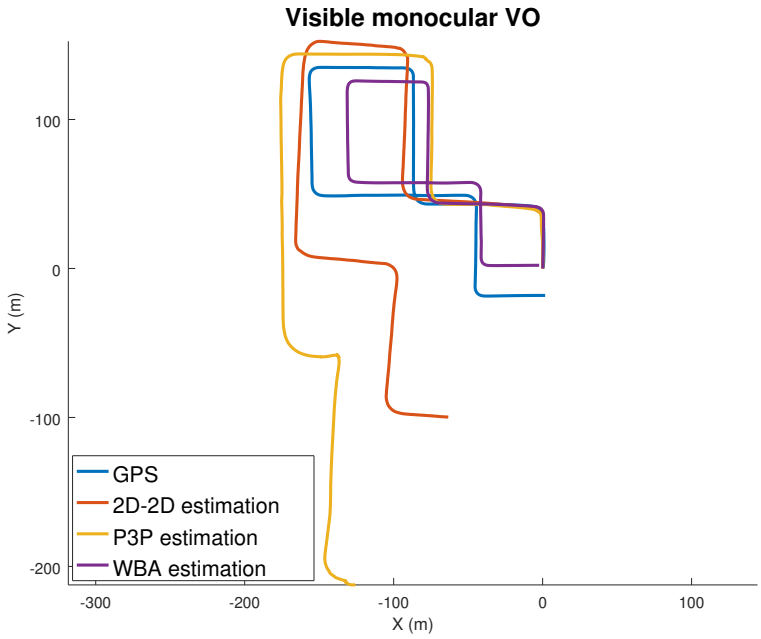


(a) Visible images

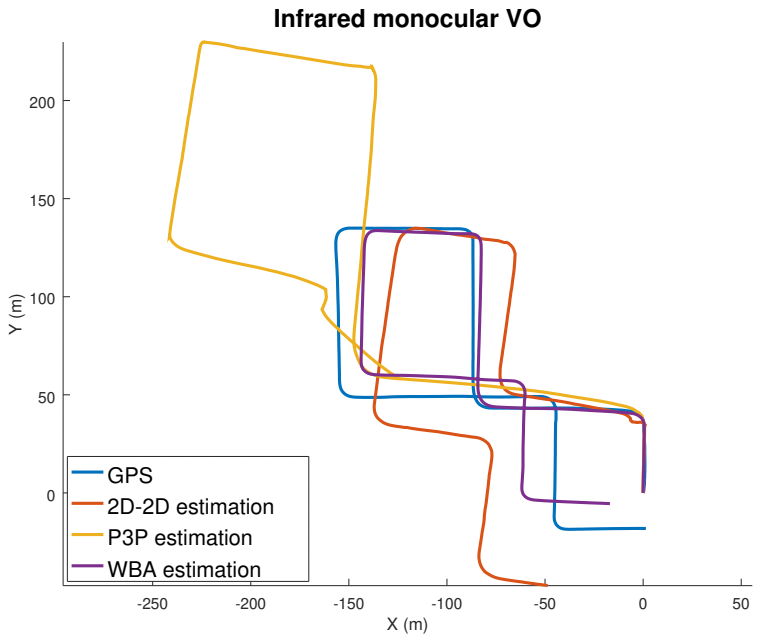


(b) Thermal images

Figure 4.9: Trajectory estimation on Sequence 2.

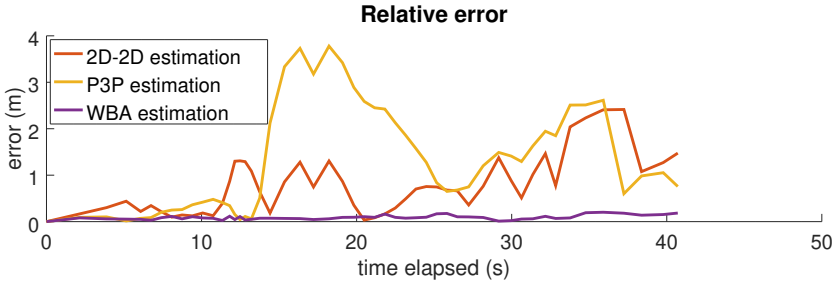


(a) Visible images

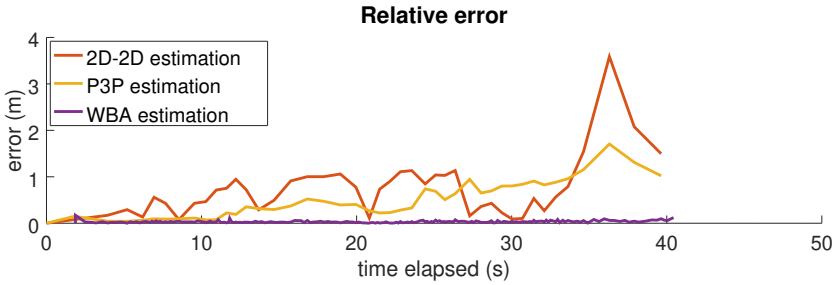


(b) Thermal images

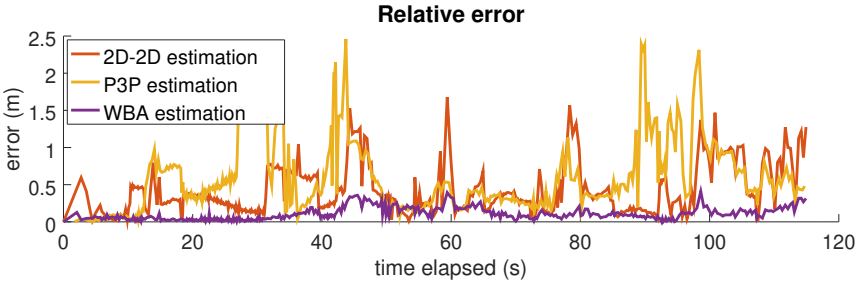
Figure 4.10: Trajectory estimation on Sequence 3.



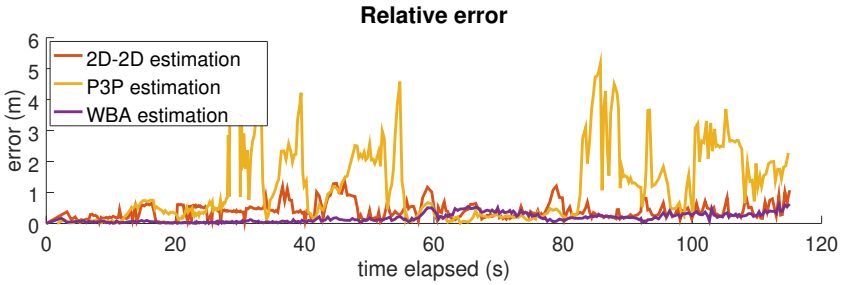
(a) Sequence 1: visible images.



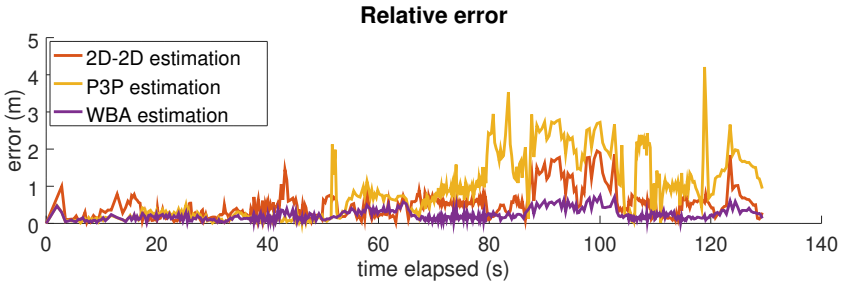
(b) Sequence 1: thermal images.



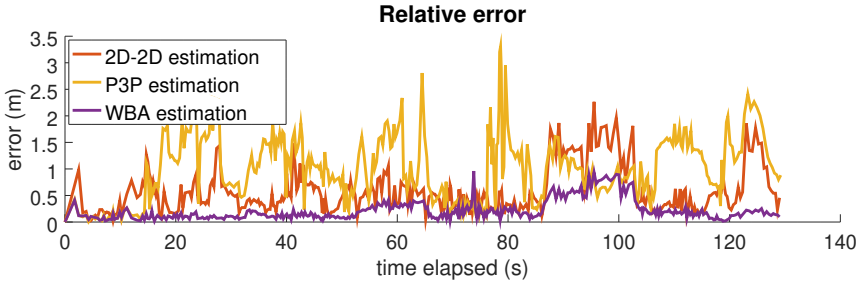
(c) Sequence 2: visible images.



(d) Sequence 2: thermal images.



(e) Sequence 3: visible images.



(f) Sequence 3: thermal images.

Figure 4.11: Relative errors obtained on the three sequences.

can be seen from Figure 4.11 that WBA is nevertheless drifting with time. This can be explained by the fact that local BA, like the P3P technique, is based on previously estimated camera poses. So, when poses are not estimated properly and errors are introduced in the process, it impacts on the rest of the estimation. Since there are no ways of correcting this drift from images only (except for loop closure), errors tend to accumulate rapidly and dramatically. Nevertheless, it is shown from all three sequences that WBA provides a reliable estimation for at least a few hundred meters.

Finally, the overall scale ambiguity remains as it was estimated from the GPS coordinates during the initialisation step. This is a more general problem with monocular setups, as explained in Section 4.5, and it will be investigated further in the following chapters with the use of stereovision and inertial data.

The VO errors obtained on all datasets are summarised in Table 4.1. Overall,

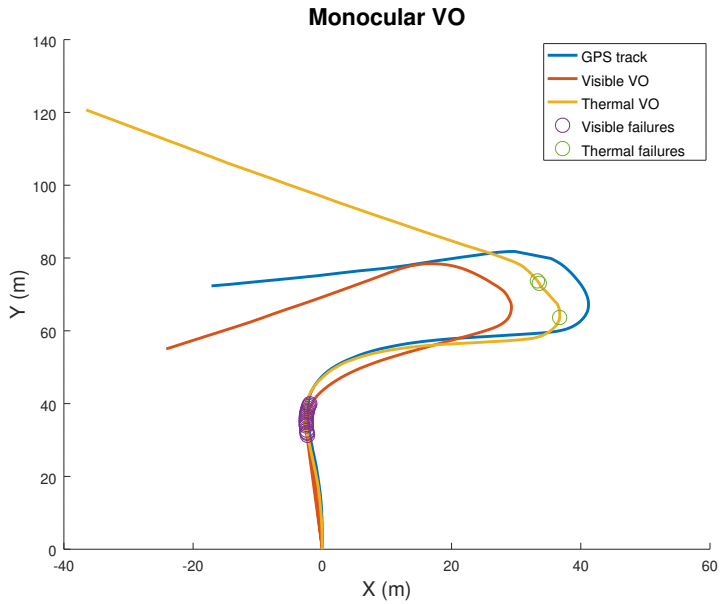
these results are promising as they show that VO can be performed in both visible and infrared modalities with the same technique and the same set of parameters. Therefore, it does not require special tuning for each type of images and can be used in a generic manner. Nevertheless, it can also be noted that the thermal camera generates more accurate trajectories than the visible one, with an overall gain in accuracy of 0.5% the distance travelled.

4.7.2 Failure recovery

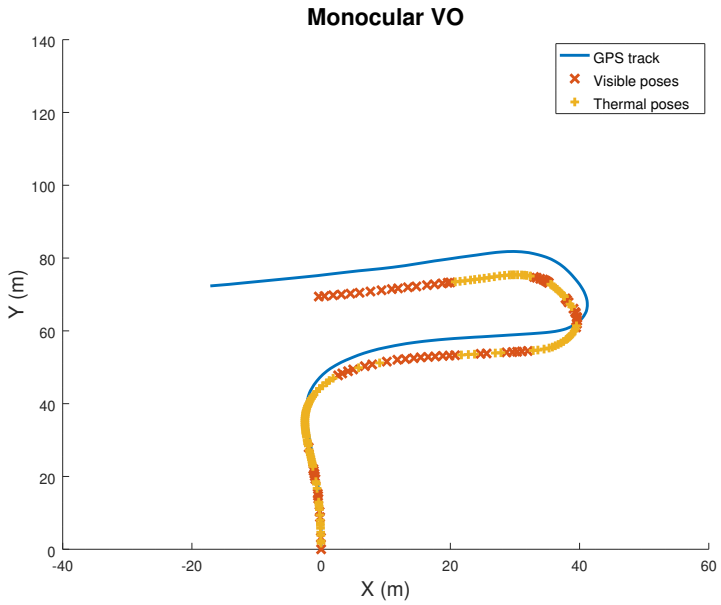
In this section, the advantage of multispectral monocular VO is demonstrated by testing the monocular VO algorithm presented in this chapter on a new dataset with challenging illumination conditions. Feature tracking was performed with both visible and thermal images simultaneously and BA was run independently with the point clouds triangulated from each camera. The trajectories obtained are shown in Figure 4.12a. It can be seen that both modalities fail to estimate motion for a few frames because the quality of the images deteriorates.

An example of images that caused these failures can be seen in Figure 4.13. The visible camera is not able to detect and track features properly as is it facing the sun and the top-right corner of the image becomes excessively bright while the rest of image is completely dark. The thermal camera on the other hand, produces images of particularly low contrast when facing trees. Some features are detected in the right side of the images but they cannot be tracked properly, making motion estimation impossible. Moreover, due to the accumulative nature of VO, these failures are affecting the rest of the trajectory which is then drifting away from the GPS track (see Figure 4.12a).

However, with a calibrated multispectral setup it is possible to continue performing VO when one of the cameras fails to produce decent images, as long as the other one captures enough details for feature tracking. This would be im-

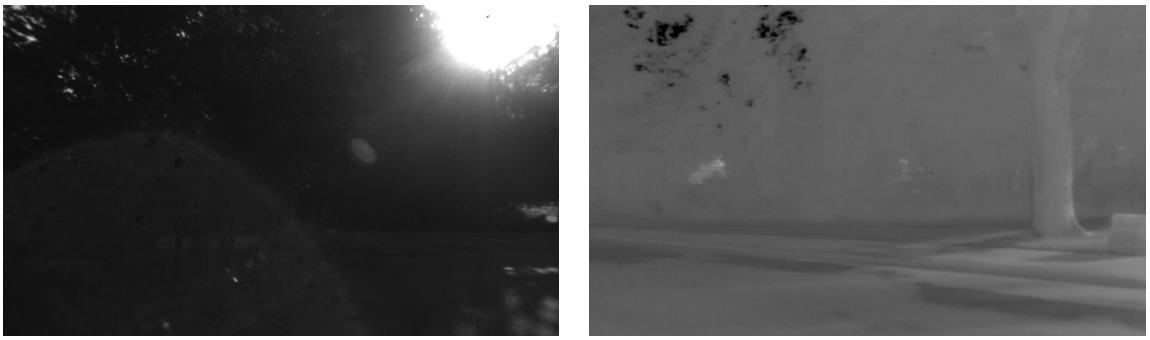


(a) Independent trajectory estimation.



(b) Alternating trajectory estimation.

Figure 4.12: Monocular VO performed independently or alternating between modalities.



(a) Visible fails. (b) Infrared fails.

Figure 4.13: Example of images where feature tracking fails.

possible with a single camera or with stereo cameras of the same modalities as both sensors would be affected in the same way. To fully take advantage of the multispectral setup, a single VO trajectory is computed using each camera alternatively, instead of producing two trajectories corresponding to each modality. Hence, feature detection and tracking are performed with both visible and thermal images simultaneously. However, only one set of 2D/3D points is used to perform BA. To determine which set to use, the camera providing the highest number of inliers after triangulation and P3P estimation is selected. Figure 4.12b shows the trajectory obtained when switching between modalities. Even though a notable drift can be seen, the trajectory is fully recovered, even when one of the cameras fails to produce adequate images. This is confirmed by the analysis of the errors obtained (see Table 4.2), as each modality alone produces larger errors. Indeed, it can be seen that combining the two modalities reduces the mean position error by 66% compared to the visible estimation and by 33% compared to the infrared.

Moreover, these results are consistent with the ones obtained in the previous experiments since the mean error produced with failures is larger than the errors obtained with datasets that do no include failures (WBA technique in Table 4.1). If the rotations induced by the lack of motion estimation during image failures are

Table 4.2: Errors comparison between each modality and alternating.

	Visible	Thermal	Multispectral
distance traveled	165m		
Error on final point (m)	18.68	55.12	**11.33**
Mean error (m)	9.67	6.81	**4.16**
Error on final point (%)	11.32	33.4	**6.87**
Mean error (%)	5.86	4.13	**2.52**

ignored, one can also notice that the shape of both visible and thermal trajectories would match the ground truth. Indeed, the scale seems to be estimated appropriately and the final trajectories do not drift as much as P3P techniques in Figure 4.8-4.10. This shows that even with short failures, the BA method do behave as before.

It must be kept in mind that this solution is only valid when feature tracking is unsuccessful with one of the camera. This would not be possible if both cameras fail to produce satisfying images. Nevertheless, the experiment shown demonstrates the superiority of multispectral setups over monocular and standard stereo setups (same modality) for challenging illumination conditions or low contrast environments.

4.8 Conclusion

In this chapter, several detection and matching techniques were introduced in order to find corresponding points of interest in consecutive images. It was explained how the relative transformation between two views can be computed from 2D features only, but also how more complex 3D-to-2D motion estimation algorithms work. The scale ambiguity problem inherent to monocular navigation was introduced.

A monocular VO system for visible and thermal images was proposed. Based on a keyframe approach, it automatically tracks features in every incoming image

but only selects those presenting the best geometric properties. It then triangulates features over a certain window and refines both the structure and camera poses in a local BA process. The superiority of this method to other simpler motion estimation techniques was demonstrated.

Additionally, it was proven that the alternating selection of multispectral cameras is useful to deal with failure cases, where it is impossible to track or perform accurate estimation in one of the modalities. However, due to the lack of global scale and because errors tend to accumulate over time, this algorithm is not sufficient to run on its own for a long period of time. To tackle this problem, the next chapter will thus focus on solving the scale ambiguity by combining the visible and thermal cameras in a stereo system.

5 | Multispectral stereo visual odometry

In this chapter, the use of multispectral stereo setups for localisation purposes is investigated. The objective of this work is to propose new techniques able to carry out feature matching between heterogeneous modalities and to show that stereo visual odometry can be performed accurately with images from distinct parts of the electromagnetic spectrum. First, a review on stereovision and multispectral state-of-the-art algorithms is given. Two innovative feature-based matching techniques are then presented to find stereo correspondences between images of different modalities. Finally, a multispectral visual odometry process is proposed to validate the concept. Both methods are evaluated on unique car datasets, including real driving conditions. All the algorithms presented in this chapter have been specifically designed to tackle the challenges arisen by the combined use of thermal and visible image data streams.

5.1 Introduction

The main advantage of stereo localisation is the use of prior knowledge regarding the orientation and positioning of two cameras, which makes the depth estimation of a scene straightforward and accurate. This information is obtained beforehand from calibration (see Chapter 3) and since it is assumed the stereo setup is properly fixed and not moving throughout the acquisition, intrinsic and extrinsic parameters are kept constant. Knowing precisely the depth of the scene makes the stereo localisation system more accurate than monocular localisation, because the later requires the scale to be estimated [3]. Depth map computation from stereo pairs of images has been extensively studied in the last decades with the development of computer vision algorithms for navigation purposes [89, 90]. However, multispectral stereovision is relatively new and currently not robust enough to rely completely on images for localisation purposes [24]. The main motivation for this work is therefore to improve state-of-the-art multispectral stereo matching and to create a robust VO algorithm which can deal with two distinct modalities.

Stereovision is limited by the depth resolution of the system because the accuracy of a point depth estimation is directly proportional to its actual depth and the distance between the cameras (baseline), as explained in Section 3.2. This means that the farther the point, the less accurately its depth can be obtained. However, if a feature is too close to the cameras, it will not appear in both images. Therefore, the baseline needs to be selected depending on the proximity of the features and the nature of the environment. This is why stereo navigation systems tend to be less efficient when embedded on vehicles such as UAVs, flying a few hundred meters above the ground [83].

Furthermore, the manipulation of multimodal images raises additional challenges. Since no linear relationship can be established between pixel intensities in

the different spectra, it is not possible to infer the value of a pixel in one spectrum based on its value in the other. This makes multispectral stereo matching challenging and less precise than conventional stereo visible matching [91, 25, 23].

5.1.1 Related work

Stereovision has received a lot of attention in the last decades with the wide spread of cameras in various fields such as 3D reconstruction, surveillance or robotics. The main aspects of stereovision are:

Calibration: initial step to estimate the stereo rig parameters.

Precision: accuracy of the reconstructed scene.

Computation time: amount of time necessary to run the algorithm.

Calibration plays an important role in any stereovision system because the whole reconstruction process depends on the intrinsic and extrinsic parameters of the cameras. Therefore, the quality of the calibration directly impacts on the accuracy of the reconstruction [66].

Accurate 3D reconstruction is the main goal of stereovision algorithms. However, the better the reconstruction, the more complex the technique and the more computation are required to match, sort and triangulate features. Precision and speed are thus closely related and one is rarely achieved without impacting the other. In that sense, Tippetts et al. have performed an exhaustive review of existing dense stereo algorithms [89]. In this review, the focus was made on precision and computation time. Even though numerous methods were tested, the study shows that GPU processing is required to reduce time consumption and obtain real-time performances. Additionally, only a few algorithms are able to run on systems with limited computation resources.

Even though Tippetts showed that real-time dense reconstruction is achievable, the tedious nature of dense algorithms makes it less suitable for embedded systems. That is why sparse features are preferred in robotics and navigation systems, where small on-board computers with limited computational power are used [92, 93]. Indeed, it is not necessary to find the depth of all pixels in the image, which reduces the computation load considerably. Moreover, image features are not only limited to a simple pixel position. Recent works have investigated the additional use of lines to robustify both motion estimation [94] or camera pose estimation [55].

Most stereovision matching techniques have been developed on stereo datasets made of still image pairs, such as the famous Middlebury dataset [95]. However, since these images were not reflecting navigation environments, Geiger has built a robust stereo setup mounted on a car and acquired long datasets to show real driving conditions in urban environment. Several versions are available; a first collection of 22 sequences over a distance of 40 kms was acquired in 2012 [90]; a more recent and accurate version focusing on scene flow and object detection was collected in 2015 [96]; the latest version was acquired in 2017 and was designed to evaluate emerging machine learning algorithms for depth prediction [97]. Because of their high quality, the distance travelled and the amount of data available, these datasets have rapidly become the reference for stereo navigation benchmarks. Algorithms are rated according to several categories depending on the application and the dataset used for testing. Concerning visual localisation, the best algorithm tested on the KITTI dataset is [98]. It is a SLAM technique based on careful feature selection and matching. The odometry is computed by estimating the orientation and translation separately. Noisy matches are detected and discarded in a Random Sample Consensus (RANSAC) framework. From the features selected, the map is then updated and when loop closures are detected, previous estimates are used to

optimise the pose graph and reduce the VO drift.

With the emergence of new visual sensors however, stereo-visible is no longer the only type of combination used for stereovision. New thermal and multispectral image processing techniques have been developed, not only for navigation purposes [21, 19, 20], but also for space applications such as space rendezvous or debris removal [99, 29]. Since pixel intensities are dissimilar between modalities, classic intensity-based techniques have become irrelevant for multispectral images, driving the need for other metrics to compare such images. Multispectral stereovision is a very recent topic and a large amount of work [100, 91, 101, 102, 103, 104] has been influenced by Viola's work on multimodal medical image registration [105]. Viola used *Mutual Information* (MI) as a metric to assess the local similarity between images.

Mutual Information is a statistical technique coming from information theory that gives a measure of mutual dependency between two random variables. MI has become popular and was used later on for wide baseline stereo matching, where images suffer from changing lightning conditions [100]; dense stereo visible-thermal reconstruction [91, 101]; multimodal pedestrian detection and tracking [38, 102]. Torabi showed that local self-similarity performs better when registering human shapes in thermal-visible pairs [103]. But experiments were carried out indoors and with a still camera which does not reflect the outdoor navigation conditions. It is therefore difficult to assess its suitability for non-controlled environments. Campo on the other hand, made use of the gradient information to take into account both magnitude and orientation, and combined it with MI to obtain an improved stereo matching metric [104]. However, the gradient intensity highly depends on the contrast in the image, which can be significantly different between multimodal images. Kovesi claims that *phase congruency* (PC) can tackle this problem as it is computed in the frequency domain rather that in the spatial one, making it more

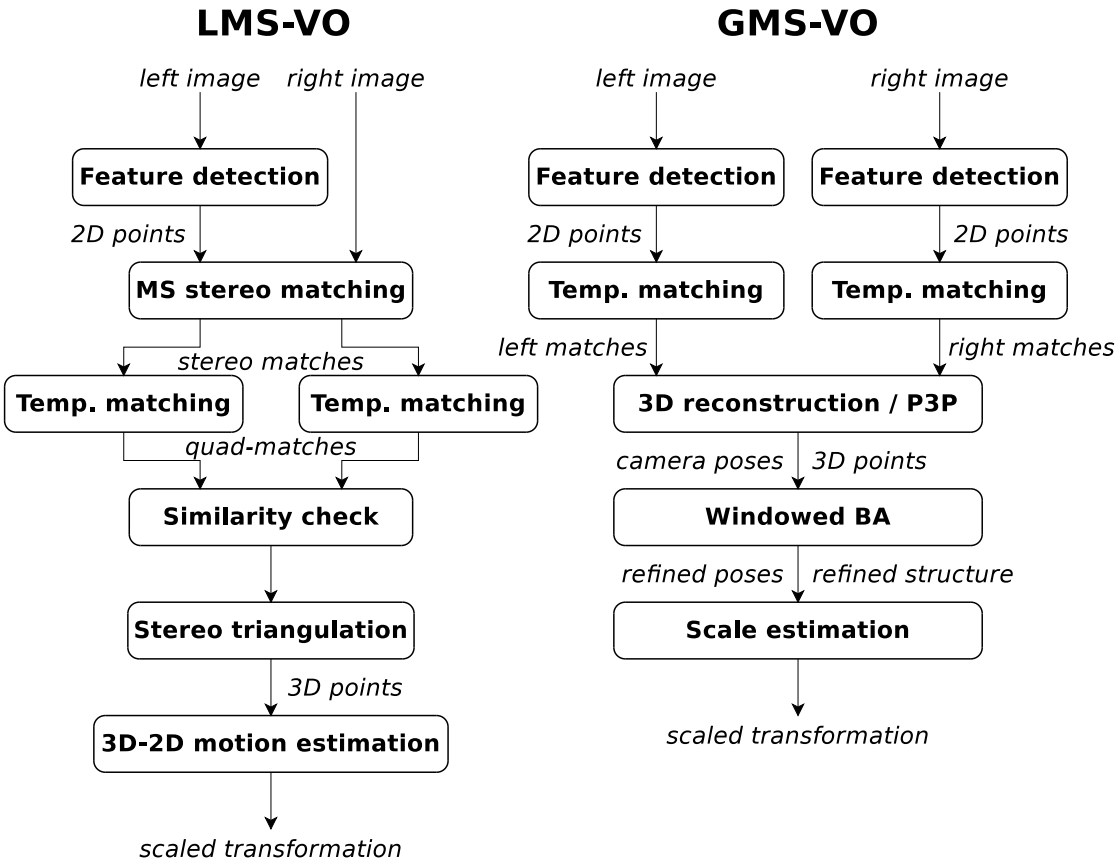


Figure 5.1: Multispectral VO processes. On the left, the local matching technique (LMS-VO). On the right, the global matching technique (GMS-VO).

robust to contrast changes [106]. For these reasons both PC and MI have been investigated in the present work and combined in order to produce a more robust stereo matching process.

5.2 Multispectral stereo localisation

As explained in Section 5.1, performing visual odometry with images of different nature is challenging. Therefore, two novel VO techniques are introduced to tackle the lack of similarity between visible and LWIR images. An overview of each method is presented in Figure 5.1, so they can easily be compared.

The first VO approach is based on local matching. It follows the conventional VO pipeline [3], which consists in detecting features in one image, performing stereo and temporal matching to find the corresponding points in all the other images, and then estimating the 3D-to-2D relative motion. The innovation of this work is coming from the multimodal stereo matching which overcomes the lack of similarity between multispectral images. The process is based on both *mutual information* and *phase congruency* concepts.

The second approach was later developed to reduce the computational load and improve the performance of the algorithm. Unlike the first technique, features are matched altogether, in a global manner. The novelty in this approach is coming from the fact that it does not require to perform the challenging feature-to-feature stereo matching. A matching process is still necessary, but it is done all at once to make it more robust. Unlike other VO techniques, temporal matching is performed first, on each modality separately. Triangulation and BA are then performed to reconstruct the scene up to a scale. Finally, the scale is recovered in an optimisation process which reprojects the scene 3D points into the stereo pairs. This technique is based on *mutual information* only, which avoids computing the time-consuming *phase congruency* images. This improves greatly the speed of the algorithm.

PC and MI are first introduced in the upcoming sections as they are the main image processing concepts adopted. The two VO methods are then explained in details. Finally, they are tested on different experimental datasets and their performances are evaluated and compared.

5.2.1 Phase Congruency

Phase congruency is a low level invariant property of an image that is defined based on the local energy model developed by Morrone and Owen [107]. The model specifies that important features in an image (corners and lines) can be

detected in the frequency domain where its Fourier components are maximally in phase. Kovesi reused and extended this concept for robust feature extraction in 2D images [108]. He showed that detecting corners in the frequency domain produces more stable features than extracting them based on gradient information (eg. Harris[56], Good Features To Track [57] or SIFT [59]). PC was originally developed to tackle changes in illumination conditions but was later employed in [88] to extract repeatable features in both infrared and visible domains.

Owen and Morrone defined the PC of a signal as:

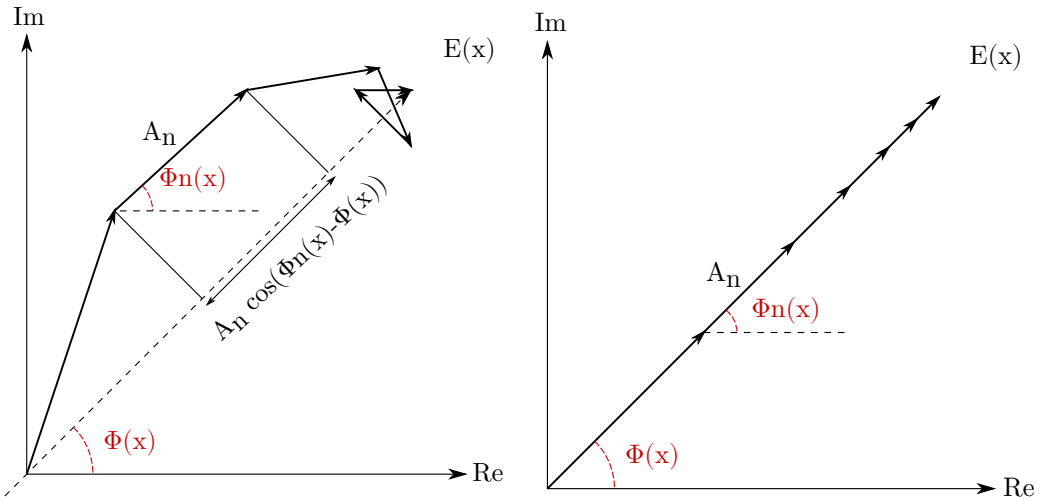
$$PC(x) = \frac{\sum_n A_n \cos(\phi_n(x) - \bar{\phi}(x))}{\sum_n A_n} \tag{5.1}$$

where A_n corresponds to the amplitude of the n^{th} Fourier component and $\phi_n(x)$ its local phase. The sum $E(x) = \sum_n A_n \cos(\phi_n(x) - \bar{\phi}(x))$ represents the local energy of the signal.

Figure 5.2 illustrates how Fourier components are related to PC. It can be noticed that $E(x)$ corresponds to the magnitude of the vector between the origin and the end point when adding all local vectors together. Hence, PC can be seen as the ratio between the local energy of the signal and the sum of local magnitude of each Fourier components. It is therefore a dimensionless quantity with maximal value 1 when all Fourier components are in phase (Figure 5.2b) or $[0, 1[$ otherwise (Figure 5.2a). This property is useful because it means that PC does not depend on the magnitude of the signal, which makes it invariant to contrast or illumination conditions for 2D image processing.

The main issue with the formulation in (5.1) is its noise sensitivity, especially when the magnitude of the signal is small. Moreover, feature localisation is not so accurate. That is why Kovesi proposed a new measure of phase congruency addressing these issues [106]:

$$PC_2(x) = \frac{\sum_n W(x) \lfloor A_n \Delta\phi_n(x) - T \rfloor}{\sum_n A_n + \varepsilon} \tag{5.2}$$



(a) Out of phase. (b) In phase.

Figure 5.2: Relationship between the phase congurency and the Fourier components of the signal when plotted head to tail.

where $W(x)$ is a weighting function to weight the different frequencies. $\Delta\phi_n(x)$ is the phase deviation : $\Delta\phi_n(x) = \cos(\phi_n(x) - \bar{\phi}(x)) - |\sin(\phi_n(x) - \bar{\phi}(x))|$. $\lfloor . \rfloor$ represents the operator that returns the value it contains if positive or 0 otherwise. T corresponds to a threshold that removes the influence of the estimated noise and ε is a small constant to avoid PC to be ill-conditioned when A_n is small.

While Morrone used only the cosine of the phase difference, Kovesi added the sine to make the calculation of phase deviation more sensitive, resulting in a sharper PC.

In practice, banks of Log-Gabor filters with different frequencies and orientations are used to compute PC. Log-Gabor filters are preferred over classic Gabor filters because they have Gaussian transfer functions in both linear and frequency scales. This way, they offer a large bandwidth while their DC component stays null in the even-symmetric filters [109]. 2D Log-Gabor filters are defined as follows:

$$G(\omega) = \exp\left(\frac{\log(\frac{\omega}{\omega_0})^2}{2(\log(\frac{k}{\omega_0})^2)}\right) \tag{5.3}$$

$$G(\theta) = \exp\left(\frac{-(\theta - \theta_0)^2}{2T(\Delta\theta)^2}\right) \tag{5.4}$$

$G(\omega)$ is the radial component where ω_0 corresponds to the centre of frequency and k to the bandwidth. $G(\theta)$ is the angular component and θ_0 is the filter orientation. $\Delta\theta$ corresponds to the spacing between orientations.

5.2.2 Mutual Information

Mutual Information is an estimate of the dependence between two random variables and is closely related to the concept of entropy. In information theory, the entropy reflects the uncertainties or the quantity of information provided by a random variable. For a random variable X, the entropy is defined as:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{N} P(x_i)\log(P(x_i)) \\ &= \mathbf{E}[-\log(P(X))] \end{aligned} \tag{5.5}$$

H(X) corresponds to the expected value of the random variable $-\log(P(X))$, also called information content, which relates the amount of information gained for each instance of X. Hence, the entropy can be seen as the average rate at which information can be learnt from an instance of X.

As illustrated in Figure 5.3, MI corresponds to the intersection between the two marginal entropies $H(X)$ and $H(Y)$. Therefore, it depends on the joint and conditional entropies of X and Y. The joint entropy corresponds to the entropy of the joint distribution P(X,Y), while the conditional entropies $P(X|Y)$ or $P(Y|X)$ reflect the uncertainties remaining about a certain variable after observing the other. As a result, MI can be expressed in different ways:

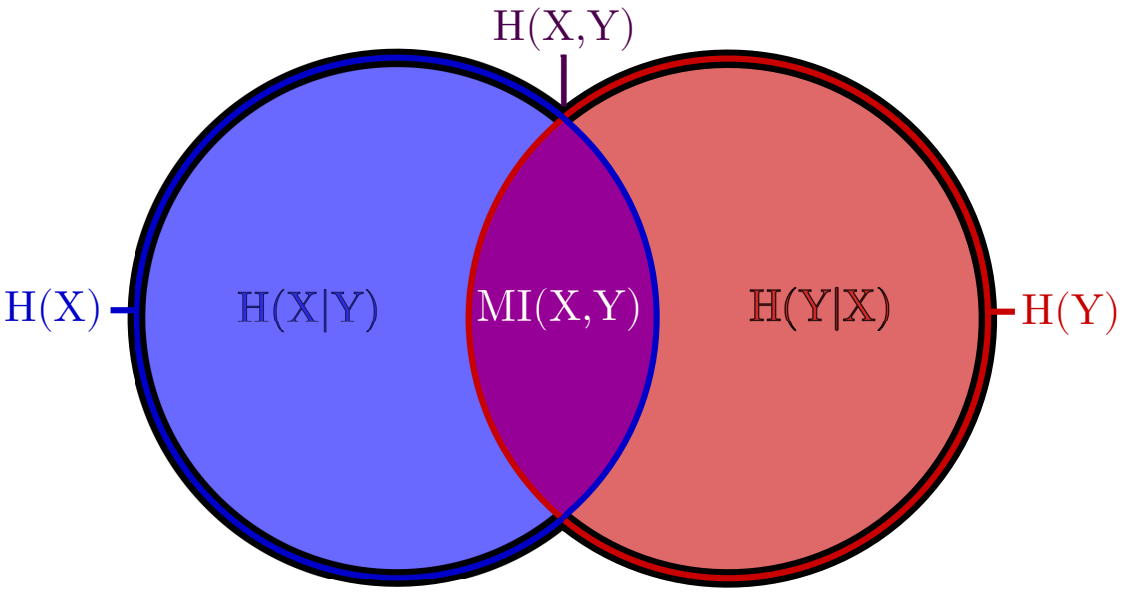


Figure 5.3: Mutual Information Venn diagram. H(X) corresponds to the blue circle, included the purple part. H(Y) corresponds to the red circle, included the purple part. Finally, H(X,Y) comprises the red, blue and purple sections.

$$\begin{aligned} MI(X,Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X,Y) - H(X|Y) - H(Y|X) \end{aligned} \tag{5.6}$$

Mutual Information can be seen as the amount of information about one variable gained by observing the other, or the reduction in uncertainties about one variable from the observation of the other. The MI between two discrete random variables X and Y can be expressed in terms of probabilities as follows:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \ln \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{5.7}$$

where $p(x,y)$ is the joint probability function of X and Y. $p(x)$ and $p(y)$ corresponds to the marginal probability functions of X and Y.

It can be noticed that unlike covariance and correlation, the notion of entropy and MI do not depend on the values of the random variables but on their probabilities instead. This is why MI is preferred for noisy or multimodal images where pixel values cannot be compared directly.

In practice, MI is employed to assess the similarity between image patches. Variables X and Y represent the pixel intensity values in each patch where x and y are specific instances of X and Y. These values belong to the interval $[0, 255]$ for 8-bit images but it is usually preferred to quantise the image into a lower number of pixel values [100]. The minimum MI value that can be obtained is 0. It means that the two random variables are independent and therefore the images tested do not share any similarity. On the contrary, if $MI(X, Y) > 0$, the higher the value the more similar the content of the images is.

5.2.3 Local MultiSpectral-Visual Odometry (LMS-VO)

Feature detection

In [108], Kovesi showed that phase congruency could be used as a mean of extracting points of interest in an image. He also showed that PC corner detection outperforms the Harris operator [56] when extracting strong corners. This approach was reused later on, for detecting robust features across modalities [25]. In the same way image moments comprise the invariant properties of an image, phase congruency moments can be computed to extract information about the PC variations, regardless of its scale and orientation. To do so, PC needs to be estimated at different orientations, which is achieved with a bank of Log-Gabor filters described in Section 5.2.1. A set of response images is obtained, from which the following quantities are defined:

$$a = \sum_{i=0}^{N} (PC(\theta_i)\cos(\theta_i))^2 \tag{5.8}$$

$$b = 2\sum_{i=0}^{N} (PC(\theta_i)\cos(\theta_i))(PC(\theta_i)\sin(\theta_i)) \tag{5.9}$$

$$c = \sum_{i=0}^{N} (PC(\theta_i)\sin(\theta_i))^2 \tag{5.10}$$

where $PC(\theta_i)$ is the phase congruency image computed for orientation θ_i.

The principal axis, which corresponds to the major eigenvector of the PC covariance matrix, defines the orientation of the image. For a local neighbourhood, it characterises the orientation of a feature. This axis can be obtained from the PC orientations by computing the angle:

$$\phi = \frac{1}{2}\text{atan2}\left(\frac{b}{\sqrt{b^2+(a-c)^2}}, \frac{a-c}{\sqrt{b^2+(a-c)^2}}\right) \tag{5.11}$$

From a, b and c the PC maximum (M) and minimum moments (m) can also be derived:

$$M = \frac{1}{2}(a+c+\sqrt{b^2+(a-c)^2}) \tag{5.12}$$

$$m = \frac{1}{2}(a+c-\sqrt{b^2+(a-c)^2}) \tag{5.13}$$

These moments correspond to the eigenvalues of the PC covariance matrix. Thus, the following properties can be inferred:

- The magnitude of M shows the importance of a feature in at least one direction. It highlights edges in the image.
- The magnitude of m shows the significance of a feature in all directions. It

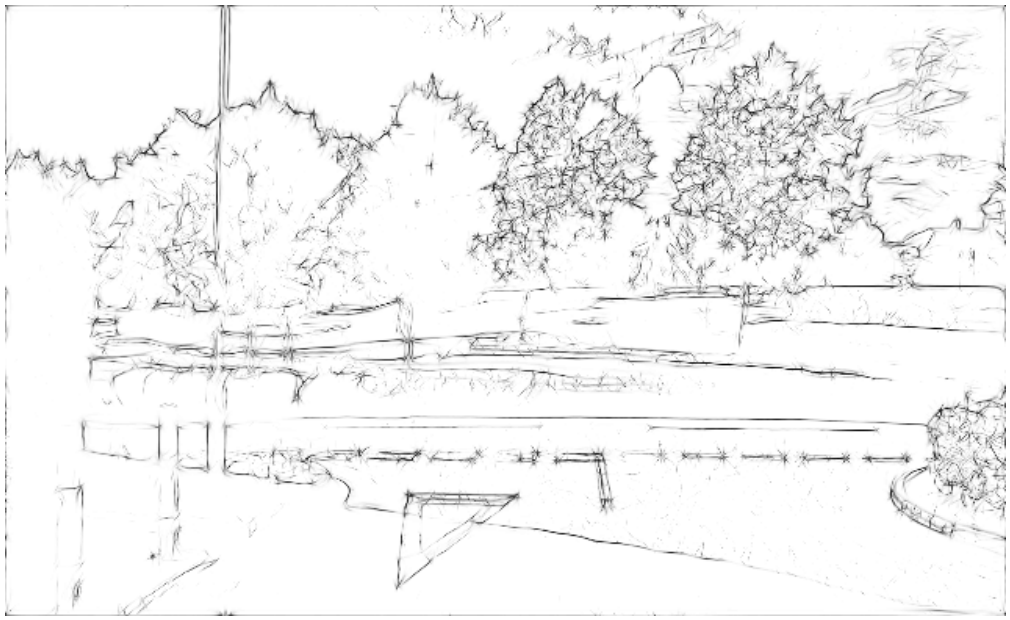
reveals the presence of a corner.

An example of PC maximum and minimum moments is given in Figure 5.4. Once the minimum moment image has been computed, corners are extracted by selecting the features that return the highest scores. Additionally, to prevent ambiguities during the matching process, a non-maxima suppression scheme is applied to keep only the strongest points in their neigbourhood (typically 3-5 pixels). Furthermore, features are not extracted in the whole image directly but in separate regions instead. This technique is called *bucketing*. It helps spreading features around the image and prevents from detecting all the corners in a tiny portion of the image. This choice was motivated by the fact that VO performs better when features are spread all over the image. Finally, the total number of points detected is limited in order to prevent having too many features being processed and to keep the VO program running at an acceptable speed. This means that a maximum number of points per bucket is enforced.

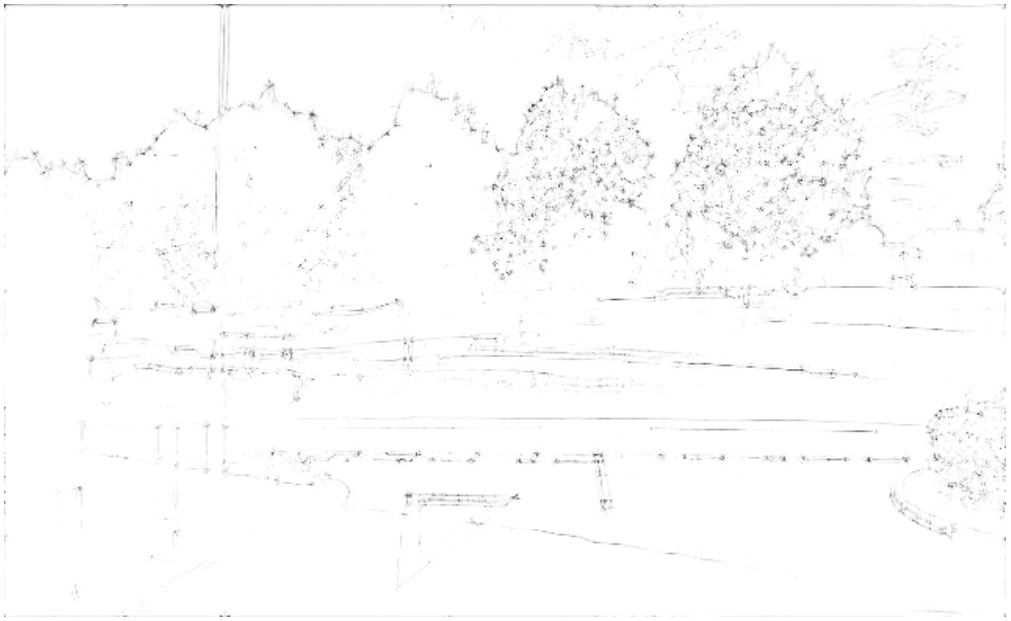
MS stereo matching

In order to compute the relative transformation between camera poses, it is necessary to find correspondences in consecutive images, and because stereo images are acquired by pairs, a total of 4 pictures needs to be matched. This process is called *quad-matching*. As illustrated in Figure 5.5, it can be divided in two distinct steps: stereo and temporal matching. As their name implies, stereo matching processes images from distinct cameras but acquired at the same time (stereo pairs), whereas temporal matching is performed on images from the same camera but acquired at different points in time (consecutive images).

As explained in Section 3.2, the use of rectified images simplifies the triangulation process and guarantees that the epipolar lines between stereo pairs are horizontal. This means that for a feature detected in one image, its stereo cor-



(a) Maximum moments.



(b) Minimum moments.

Figure 5.4: Phase congruency moments. At the top, the maximum moments. At the bottom, the minimum moments used for feature detection.

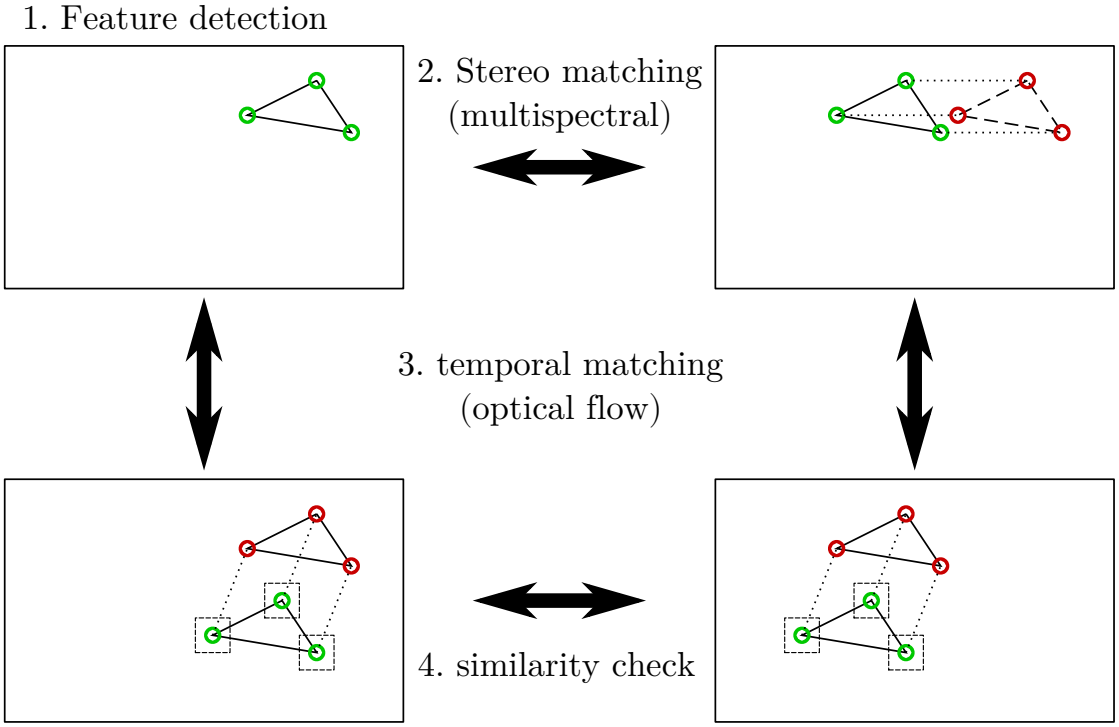


Figure 5.5: The quad-matching process. Top-left: features are detected in the left image. Top-right: red features are matched with green features located on the same row. Bottom images: features are tracked then compared using self-centred windows.

respondent should be located on the same row, in the other image. The most common approach to perform stereo matching consists in detecting a large set of features in both stereo images and then computing similarity scores based on their descriptors (see Section 4.2). However, this method is not suited for multispectral images because features are not likely to be detected at the exact same location in both spectra. So, if keypoints are not detected on the same row, they cannot be matched or if they do (by extending the search to others rows around), some noise is added to the process due to the difference in position. This method was nevertheless tested in [19], but the number of correctly matched features obtained at the end of the process was limited.

To avoid this kind of inaccuracies, the proposed method detects features in only one image, usually the left one. Stereo matching is then performed by testing every pixel in the right image that belongs to the epipolar line (same row as the original feature). This process is greedier than the descriptor approach, but it is also more precise as it guarantees that matches will be located on the epipolar line. However, it is not necessary to go through the whole epipolar line to find stereo corresponding features. As shown in Figure 5.6, the projection $\mathbf{x}_r = [u_r, v_r]^T$ of a point $\mathbf{X}$ in the right image will always be located farther on the left than its stereo correspondent in the left image ($\mathbf{x}_l = [u_l, v_l]^T$). Moreover, for car navigation, it is highly improbable that the front of a moving vehicle will be obstructed by an object closer than few meters. A minimum distance can then be set and converted into a maximum disparity value (in pixels) where it is not worth looking beyond. Of course this distance can be adapted depending on the type of platform used. For this work, a minimum distance of 2 meters was set, which corresponds to the distance between the cameras, located on the roof, and the front of the vehicle. The maximum disparity can be expressed as:

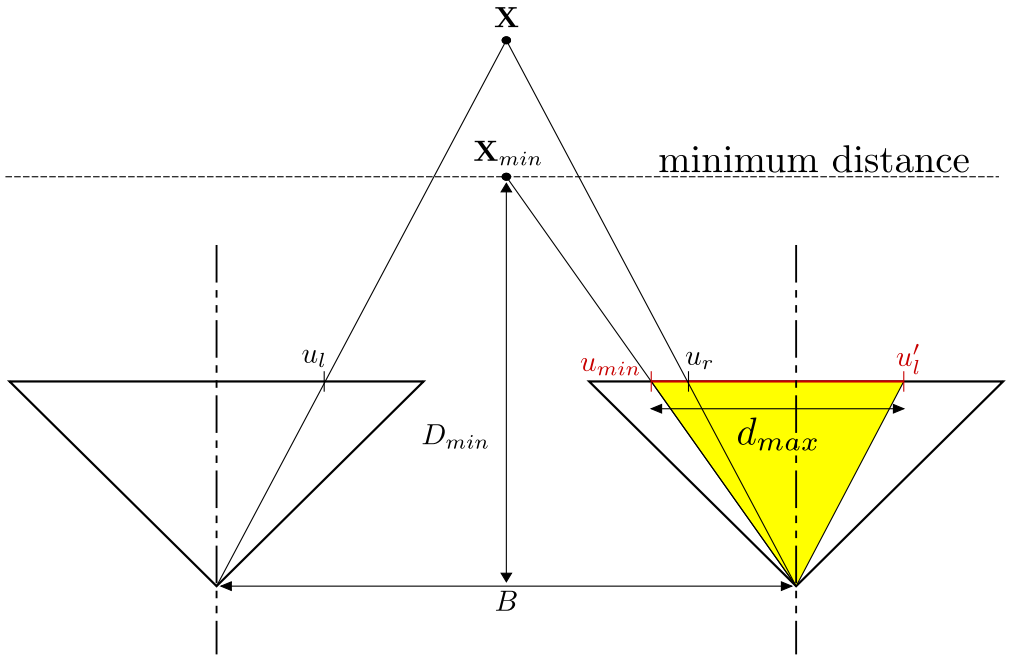


Figure 5.6: Limiting the search area based on a minimum distance. The projection of point $\mathbf{X}$ in the right image is located in the interval $[u_{min}, u'_l]$ (yellow area).

$$d_{max} = \frac{f \times B}{D_{min}} \tag{5.14}$$

with D_{min} the minimum distance, f the focal length and B the baseline.

Given a feature $f_l = I_l(u_l, v_l)$ in the left image I_l, the search area (Ω) in the right image I_l is defined as :

$$\begin{aligned} \Omega &= \bigcup_i I_r(i, v'_l), \quad \forall i \in [u_{min}, u'_l] \\ u_{min} &= \begin{cases} u'_l - d_{max} & \text{if } (u'_l - d_{max}) > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.15}$$

where $x'_l = [u'_l, v'_l]^T$ corresponds to the position of x_l in the right image.

To reduce the processing load even more, the search area is further reduced by testing only pixels with high PC responses. Each pixel belonging to the reduced search area is then compared to the original feature in order to find the best

match. Comparison is done between a window centred around the pixel to test in the right image and a window centred around the original feature in the left image. The window size has been experimentally set to 80 $px \times 80$ px. This way, the window is large enough to capture relevant information about the feature, but small enough to describe it locally. A similarity score is computed based on the MI between the PC of the two windows. PC provides information about the content of the window by highlighting its contours, whereas MI reflects the dependence between the windows in a probabilistic manner. MI is preferred over simpler distance metrics because it is not possible to compare pixel or PC values directly due to the different nature of the multispectral images. The similarity score can be expressed as:

$$Z_i = MI(PC(\Delta I_l(f_l)), PC(\Delta I_r(f_i))) \quad i \in \Omega \tag{5.16}$$

where $\Delta I_l(f_l)$ refers to the window centred on the original feature in I_l and $\Delta I_r(f_i)$ represents the window centred on the pixel being tested in I_r. $MI(.)$ and $PC(.)$ corresponds respectively to the mutual information and phase congruency operators. The pixel in the search area returning the highest score is then selected as the best match.

Temporal matching and similarity check

Once features have been matched in a stereo pair at time t, temporal matching needs to be performed with the next stereo pair at time $t + 1$ to complete the quad-matching process. Because temporal tracking is done independently, with images of the same modality, several standard and robust matching techniques can be employed. As it has been shown in Chapter 4, optical flow methods achieve satisfying results in both infrared and visible domains without the need for special parameter tuning. Therefore, the same approach, based on the pyramidal KLT

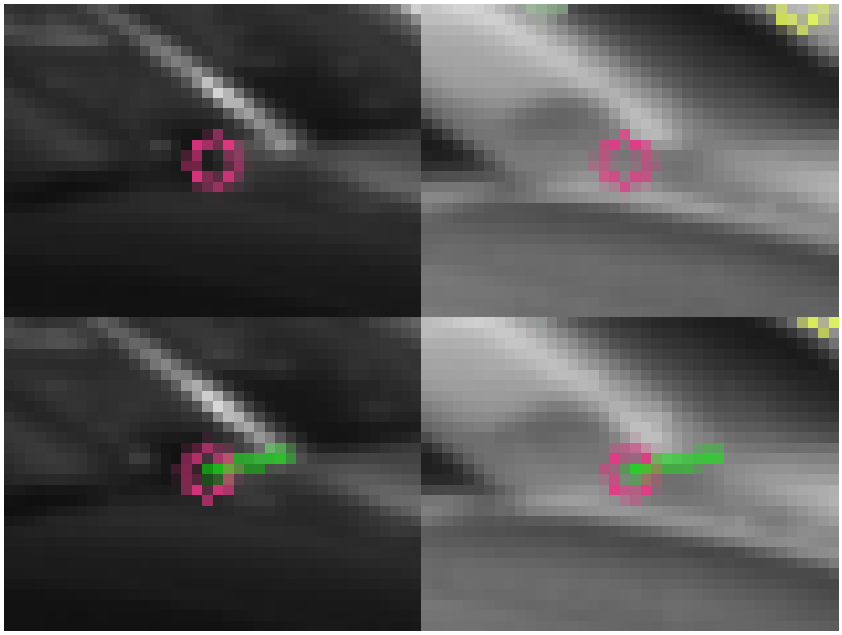
algorithm [65] is used here to track features from one stereo pair to the other. The main advantage of tracking features over time rather than matching features on different stereo pairs is that tracking is more robust to abrupt changes (e.g turn, speed bump, etc...)[19]. Moreover, employing the same matching approach as before would be too heavy since it was designed for a multispectral stereo pairs and the search cannot be restricted to a single row this time. Indeed, unlike stereo images the transformation between two consecutive camera poses is unknown and the epipolar lines cannot be determined prior to feature matching.

Stereo and temporal matching can induce some errors due to few wrong feature associations or tracking. This cannot always be prevented and these outliers can have a dramatic effect on the trajectory if they are kept during the motion estimation step. To detect wrong matches, one more step is added to the algorithm. It consists in checking that the features tracked are indeed the same and located on the same row. Thus, similarity between these features is evaluated to guarantee that the image content matches. As in the previous section, MI between the two PC windows centred around each tracked feature (in the left and right images) is computed. If the score is lower than a certain threshold, it will not be considered similar enough and the quad-match will be rejected. An example of quad-matches is displayed in Figure 5.7 to show the features obtained after the matching process.

Motion estimation

Once features have been matched in all four images, the relative transformation between the two stereo pairs can be estimated as shown in Figure 5.8. The first pair is used to triangulate the features and compute the structure of the scene. As explained in Section 3.2, triangulation is straightforward with stereo rectified images and a 3D point cloud is obtained. These points are then reprojected into the second stereo pair to assess the errors produced by the current motion estimate

(a) All quad-matches between consecutive stereo pairs.



(b) Close up view of a quad-match.

Figure 5.7: Quad-matches. Coloured circles show the position of each feature in all the images while green lines in the bottom images show the 2D shift between the two stereo pair.

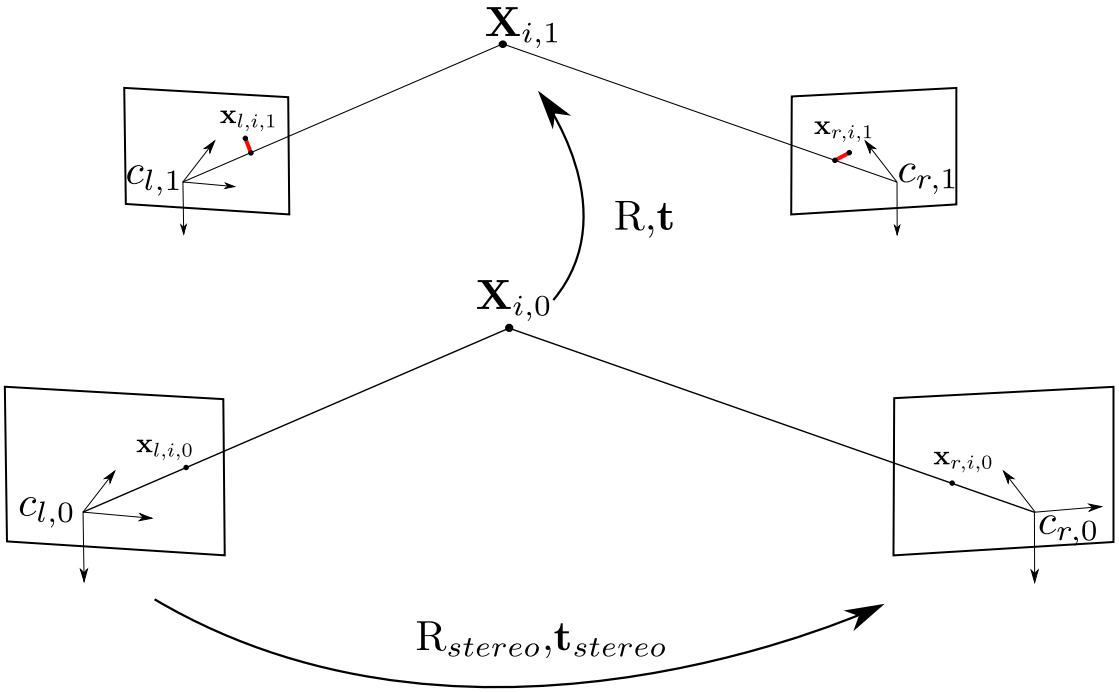


Figure 5.8: Coordinate systems and camera poses of a moving stereo rig.

(shown in red in Figure 5.8). The relative transformation $[R|\mathbf{t}]$ is then updated in an optimisation scheme which minimises these reprojection errors for all the features, until a good estimate producing a small overall error is reached. The least-squares cost function of this optimisation problem is defined as:

$$\arg\min_{R,\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{x}_i - (\pi(R\mathbf{X}_i + \mathbf{t}))\|_2 \tag{5.17}$$

where R and $\mathbf{t}$ correspond to the relative rotation and translation being estimated, respectively. $\mathbf{X}_i$ is a 3D feature which has been triangulated from the observed feature $\mathbf{x}_i$. π is the projecting function.

This optimisation problem can also be viewed as the estimation of the pose of the next stereo pair with respect to the first left camera pose (reference frame). The camera poses corresponding to each view are then expressed as:

$$
\begin{aligned}
P_{l,0} &= [I_0|\mathbf{t}_0] \\
P_{r,0} &= [R_{stereo}|\mathbf{t}_{stereo}], \text{ with } R_{stereo} = I_0 \text{ and } \mathbf{t}_{stereo} = [B,0,0]^T \\
P_{l,1} &= [R|\mathbf{t}] \\
P_{r,1} &= [RR_{stereo}|R\mathbf{t}_{stereo} + \mathbf{t}]
\end{aligned}
\tag{5.18}
$$

Due to the rotation involved, the pose estimation corresponds to a non-linear least-squares problem. Therefore, it cannot be solved easily and the solution is approximated iteratively, by assessing the cost function locally and computing an update which reduces the cost. Moreover, due to the non-convexity and non-linearity of the problem, the algorithm is likely to stop when it reaches a local minimum. This scenario is not satisfactory because it does not necessarily corresponds to the optimal solution and can end up far from the global minimum. The initial parameters are then critical as they guide the optimisation process and ultimately affect the solution. Generally, the closer to the truth, the faster the algorithm converges and the better it behaves. By default, the initial motion is set to $R = I_{3\times3}$ and $\mathbf{t} = [0,0,0]^T$, which means that the cameras did not move. This is not a particularly good initial estimate but because the rotations and translations involved between consecutive frames are not significant, it is a sufficient guess for the algorithm to converge. Better initial values are utilised when previous motion has been estimated. Indeed, the relative transformation is changing at a slow rate because a vehicle does not change speed abruptly, it accelerates or decelerates first. Hence, the transformation that has been estimated at a certain step can be reused as initial guess for the next iteration. In terms of optimisation techniques, both *Gauss-Newton* and *Levenberg-Marquardt* methods were tested. Even though they produced similar results, *Levenberg-Marquardt* seemed more robust when the initial values were less accurate. It was thus used in the experimental section. More details concerning non-linear least-squares optimisation can be found in Appendix

A.

As shown in Figure 5.7, even if most of the features are matched properly, several wrong matches can still be spotted. This is another problem because the presence of a few outliers deteriorates the trajectory estimation, and it is nearly impossible to obtain a perfect matching when dealing with multispectral images. But rather than solving this tremendously challenging task, the motion estimation process is made more stable by utilising a RANSAC scheme. RANSAC is a framework which allows robust estimation from noisy data, as long as the number of inliers is sufficiently bigger than the number of outliers [7]. Motion is first computed from a set of 3 points selected randomly. From this estimate, every point is reprojected in the different images and labelled based on its reprojection error. If the error is small they are considered inliers, otherwise outliers. The operation is then repeated several times with a different set of random features each time, and the set which generates the most inliers is considered the most representative of the sample. The inliers from this set are considered the most robust features, and only these are used in the final motion optimisation. A pseudo-code version of the framework is presented in Appendix B.

5.2.4 Global MultiSpectral-Visual Odometry (GMS-VO)

As seen in the previous section, local neighbourhoods do not always provide sufficient information to perform accurate multispectral matching and a lot of features can produce wrong matches. This is especially the case in low contrast and texture-less regions. To prevent these matches from deteriorating the motion estimation, solutions such as RANSAC or similarity checks were employed. However, it is not always enough and some outliers can still remain. In order to produce a stronger matching process, a different approach is proposed. This method consists in matching the features from different modalities together, in a global manner.

A monocular approach is used to detect, track and triangulate features in each modality, but also to compute a first unscaled estimation of the new camera pose. A global matching process is then performed to recover the unknown scale. As it can be seen in Figure 5.1, the structure of the algorithm is quite similar to the monocular process presented in Section 4.7. In fact, the main difference resides in the scale estimation step which takes advantage of the stereo setup to remove the scale ambiguity.

Feature detection and tracking

As feature extraction and tracking were performed reliably with both thermal and visible images in Chapter 4, the same algorithm is employed here. Therefore, features are detected with the Good Feature To Track (GFTT) algorithm [57]. Their new positions in the subsequent images are then estimated with a pyramidal Lucas-Kanade method [65]. To obtain an accurate feature triangulation, it is important that the camera undergoes sufficient translation between time t and $t+1$, especially when moving along its optical centre. Thus, the keyframe approach described in Section 4.6.1 is adopted to ensure sufficient translation between consecutive frames. At each iteration, outliers are detected and rejected during the estimation of the essential matrix to make sure that all features satisfy the epipolar constraints. The estimation is performed with the 5-point algorithm, used inside a RANSAC scheme. Feature detection and tracking are performed with images from each camera separately. This means that two sets of features are obtained simultaneously.

3D reconstruction

From the essential matrix, the relative transformation between two camera poses is obtained by singular value decomposition. Because cameras are treated indepen-

dently, each camera produces its own motion estimation and generates a different point cloud. Additionally, they have different coordinate systems. However, they relate to the same physical transformation as they are mounted on the same vehicle. Since the extrinsic parameters of the stereo setup have been estimated during the calibration, it is possible to express one camera in the coordinate frame of the other. Therefore, a single point cloud can be generated by expressing all points in a common coordinate system. It is more complex to obtain a single transformation from both motion estimates but because they should not be so different, the average rotation and translation can be selected as the best approximation of the two transformations.

When rectified, left and right images are related by a simple translation along the baseline. To facilitate calculations, the origin of the system is set to match the left camera coordinate frame. Points triangulated from this camera are already expressed in the appropriate coordinate system, but points triangulated from the right camera are not and need to be converted. This is done by simply adding the value of the baseline to their coordinate. The same process is applied to the camera positions. The camera orientations are not affected as both cameras are already aligned.

Finally, once all points and camera poses have been expressed in the same frame, they are refined altogether in a bundle adjustment framework.

Scale recovery

After bundle adjustment, the 3D points are reprojected in all the images. However, BA was performed in the left coordinate frame with unscaled coordinates, regardless of the position of the right camera. This means that when 3D points are reprojected into the right images, they will very unlikely be located at the correct position. Their position along the y-axis is the same as in the corresponding

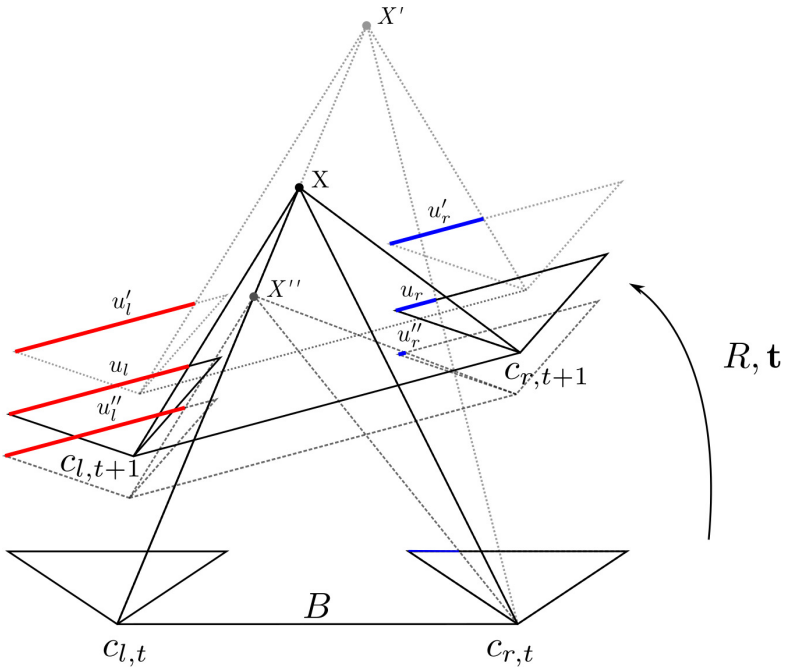


Figure 5.9: Feature reprojection for different scales.

stereo image, but there is an ambiguity concerning their position along the x-axis. Since the geometry between camera poses is known, the only information missing is the scaling factor (see Figure 5.9). Therefore, the global matching technique presented here aims at finding the scale that maximises the global similarity between the reprojected features. This approach is similar to direct VO where the relative transformation between camera poses is estimated directly by minimising the difference in pixel intensities [51]. However, the parameter estimated here is the scale rather than the relative pose transformations and MI is employed as a measure of likelihood rather than pixel intensities.

The optimal scale can be expressed for a stereo pair as:

$$\hat{\lambda} = \arg\max_{\lambda} \; MI(\Delta I_l(P_l\mathbf{X}), \Delta I_r(\lambda P_r\mathbf{X})) \tag{5.19}$$

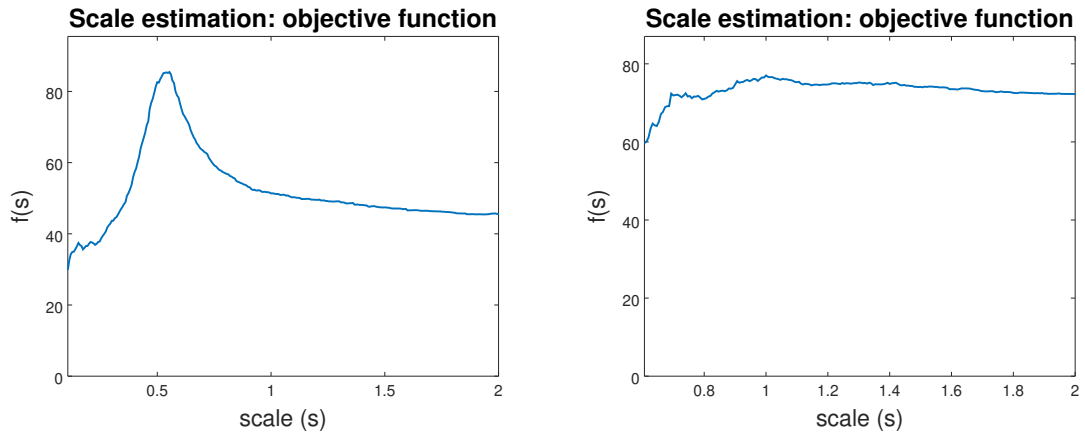
where $\mathbf{X}$ is the 3D location of the feature considered and $\Delta I()$ corresponds to the operator that extract a window around a certain position in the image. P is the

projection matrix that projects a 3D point into a 2D pixel location.

A minimum of one feature is necessary to estimate the scale. However, optimising with only one feature might not be accurate and could easily end up in a local maxima. This is why the scale is estimated in a least-squares approach, by maximising MI for all the features triangulated. The square sum of all MI then gives an estimate of the global similarity between the stereo pairs. It forms an objective function to the following optimisation problem:

$$\begin{aligned}\hat{\lambda} &= \arg\max_{\lambda} \sum_{i=1}^{N}\sum_{j=1}^{M} MI(\Delta I_{l,ij}, \Delta I_{r,ij})^2 \\ &= \arg\max_{\lambda} \sum_{i=1}^{N}\sum_{j=1}^{M} MI(\Delta I_{l}(P_{l,j}\mathbf{X}_i), \Delta I_{r}(\lambda P_{r,j}\mathbf{X}_i))^2\end{aligned} \tag{5.20}$$

When the images have an important contrast and the scene provides textured regions, the objective function produces a big and narrow peak centred at the optimal scale (see Figure 5.10a). This is convenient and can be optimised easily to reach the maximum. However, when contrast is reduced or edges are not so clear (eg. motion blur), changes in scale do not affect the objective function as much because of the reduced similarity in the overall images. As a result, the peak is flattened (see Figure 5.10b) and it becomes harder to find the optimal scale. Yet, this is a general problem for multispectral images. When features are less salient it becomes more challenging to find similarities. For local multispectral matching it results in more outliers and biased motion estimation. However, bundle adjustment computes motion based on previously estimated camera poses. This means that when scaled camera poses have been properly estimated at the previous iteration, the new camera poses and 3D points obtained from BA are also scaled. As a result, for each stereo pair, 3D points can directly be reprojected at the right location in both images. The optimal scale for this iteration where motion is already scaled



(a) Objective function with a clear peak.

(b) Flat objective function.

Figure 5.10: Examples of objective functions.

should then be 1. Therefore, this value is used as an initial guess for the scale estimation process. This way, if the objective function is flat, the value of the scale should not vary much during the optimisation, due to the small gradient in the objective function. The final scale should then be approximately the same as its initial value, which is an acceptable scale estimation if the motion has been scaled properly at previous iterations. In practice, the scale is never exactly 1 because of the noise included in the images and the precision of the stereo VO algorithm. However, when these errors accumulate and produce significant under or over-estimations, the scale optimisation process takes advantage of the stereo setup and compensates for this drift. It does not remove the drift completely and cannot correct errors made at previous iterations, but the scale is adjusted for the next steps unlike monocular VO which keeps drifting until the end.

Several adjustments were made to improve the performance of the scale recovery algorithm. First, as for the LMS-VO technique, mutual information was computed on phase congruency images to evaluate the similarity between local regions from the same stereo pair. However it was noticed that, unlike the LMS-

VO method where each pixel was tested independently, the use of PC in GMS-VO was leading to non-smooth objective functions during scale estimation, which were harder to optimise. Moreover, using only MI removes the need to perform time-consuming PC computations. Therefore, the new technique relies entirely on Mutual Information (computed from the original images) and runs faster. Secondly, it has been noticed that close features are the ones contributing the most to the objective function because they are shifting more as the scale changes, and most of these features are located in the most recent keyframe. The older the frame the less close features it contains. This is due to the fact that these features are the first to get out of the image scope or to be rejected as feature tracking is less precise along the image border. To improve further the computation speed of the scale estimation process, it was decided to only consider the last stereo pair of the BA window and the features observed in these images. However, the full window is still utilised to perform accurate triangulation and bundle adjustment.

Pose covariance estimation

The accuracy of the solution returned by an optimisation process can be assessed for all parameters by computing the covariance matrix. For bundle adjustment, the set of all parameters corresponds to the 6 DoF poses of each keyframe in the sliding window and 3 DoF position of the points. Thus, the covariance matrix corresponds to a $6 \times M + 3 \times N$ symmetric matrix, where M is the number of keyframes in the window and N the number of points being considered. For the scale estimation, the only parameter being optimised is the scale, instead of computing a covariance matrix, a scalar is then obtained which corresponds to the variance of the scale. With least-squares optimisation processes, the covariance matrix can be obtained by computing the inverse of the Hessian matrix, evaluated at the optimal state (state resulting from the optimisation). Let us consider the following least-squares

optimisation problem:

$$x^* = \arg\min_x \|f(x)\|^2 \tag{5.21}$$

The covariance of x, evaluated at x^*, is defined as:

$$P(x^*) = H(x^*)^{-1} \simeq (J(x^*)J(x^*))^{-1} \tag{5.22}$$

As explained in Appendix A, when solving the least-squares problem with iterative methods such as *Gauss-Newton* or *Levenberg-Maquardt*, the Hessian is approximated by the square of the Jacobian matrix. However, to compute $P(x^*)$, $J(x^*)J(x^*)$ needs to be invertible, which means it must be full rank. To avoid cases where $J(x^*)J(x^*)$ is rank deficient, feature are carefully selected by rejecting outliers after triangulation and during the PnP estimation, but also by enforcing a minimum distance between every 2 points in each image in order to avoid duplicates.

Because each optimisation is independent from each other, pose covariances are computed at each iteration in the local window. As explained in Section 5.2.4, BA is run first and provides first pose estimates and their corresponding covariances. The scale is then computed and used to scale the BA poses. This can be represented by a scaling transformation:

$$f(\mathbf{X}, \lambda) : \mathbb{R}^{6\times M+1} \rightarrow \mathbb{R}^{6\times M} \tag{5.23}$$

$$f(\mathbf{X}, \lambda) = F\mathbf{X} \tag{5.24}$$

$$F = \begin{bmatrix} \lambda I_{3\times3} & 0_{3\times3} & \dots & \dots & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \lambda I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \dots & \dots & 0_{3\times3} & I_{3\times3} \end{bmatrix} \mathbf{X} \tag{5.25}$$

However, pose covariances cannot simply be multiplied by the scale variance. Instead, an augmented P is created from both :

$$P = \begin{bmatrix} Cov(\mathbf{X}) & \mathbf{0}_{6M\times 1} \\ \mathbf{0}_{1\times 6M} & Var(\lambda) \end{bmatrix} \tag{5.26}$$

The covariance of the scale poses is then obtained by applying the linear transform to P:

$$P' = FPF^T \tag{5.27}$$

Once scaled, camera poses and their covariances can then be accumulated at each iteration to estimate the current camera pose and its corresponding uncertainties. Keeping the same sliding window, each keyframe can be represented as a local transformation $T_i^L \in \mathbf{X}$, relative to the first frame of the window:

$$\begin{aligned} T_i^L &= [\boldsymbol{\theta}_i^L, \mathbf{t}_i^L]^T \; or \; [R_i^L|\mathbf{t}_i^L], i \in [1, M] \\ P_i^L &= \mathrm{Cov}(T_i^L) \end{aligned} \tag{5.28}$$

where P_i^L is the covariance of T_i^L, obtained from (5.27).

In the same way P was propagated into P' following the scaling transformation, local pose covariances P_i^L are converted into a global pose covariance P_i^G. To do so, the relative transformation T_i^L is added to the global pose of the first window frame:

$$T_i^G = T_1^G \otimes T_i^L \tag{5.29}$$

$$T_i^G = \begin{bmatrix} R_1^G \mathbf{t}_i^L + \mathbf{t}_1^G \\ R_1^G R_i^L \end{bmatrix} \tag{5.30}$$

The covariance is then updated with the jacobian of the pose multiplication:

$$P_i^G = J_i \begin{bmatrix} P_1^G & 0_{6\times6} \\ 0_{6\times6} & P_i^L \end{bmatrix} J_i^T \tag{5.31}$$

$$J_i = \begin{bmatrix} \frac{\partial \mathbf{t}_i^G}{\partial \mathbf{t}_1^G} & \frac{\partial \mathbf{t}_i^G}{\partial R_1^G} & \frac{\partial \mathbf{t}_i^G}{\partial \mathbf{t}_i^L} & 0_{3\times3} \\ 0_{3\times3} & \frac{\partial R_i^G}{\partial R_1^G} & 0_{3\times3} & \frac{\partial R_i^G}{\partial R_i^L} \end{bmatrix} \tag{5.32}$$

The current global camera pose then becomes T_M^G, with associated covariance P_M^G.

5.2.5 Experimental Validation

The validity of the solutions proposed and their performances are evaluated on various image sequences acquired with the car stereo rig presented in Section 1.4.1. Both LMS-VO and GMS-VO techniques are tested on all the datasets collected, because one estimation is generally not representative of all the scenarios possible and thus not sufficient to show the suitability of the algorithms proposed. A total of 3 datasets are presented, ranging from 100 m to 600 m, and comprising straight lines, turns and various changes in speed. For comparison purposes, both the local and global matching techniques are displayed along with the GPS track in Figure 5.12. The Xsens MTi-G GNSS device was utilised to record fused GPS/INS coordinates which will be considered as a ground truth in the following experiments. The precision and frequency of GPS receivers can be questioned when used as a mean of comparison. However, the filtered trajectory provides data at a high rate (rate of inertial sensor) and reduces the risk of measuring erroneous data. More information about this sensor can be found in Section 1.4.1.

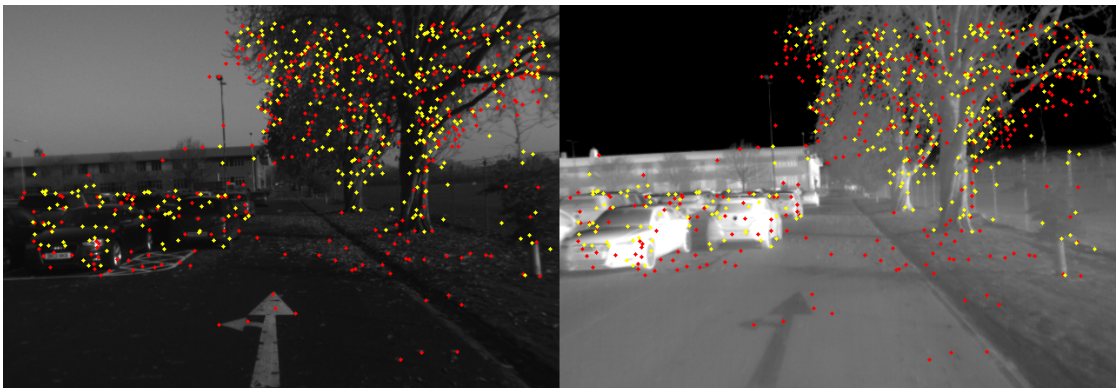
Scale estimation (GMS)

The first parameter to be tested is the accuracy at which the scale is recovered. Figure 5.11 shows the reprojection of the 3D scene into the stereo images before and after the scale optimisation step. On the top image pair, features are reprojected using unscaled camera poses. It can be clearly seen that the stereo points obtained do not match the image content, especially at the bottom of the images. In the bottom pair however, the points were reprojected with the scale obtained after optimisation and it can be seen that the features now correspond. Red points represent features which have been detected and tracked in visible images (from the left camera) and yellow points correspond to features detected and tracked in the thermal domain (from the right camera).

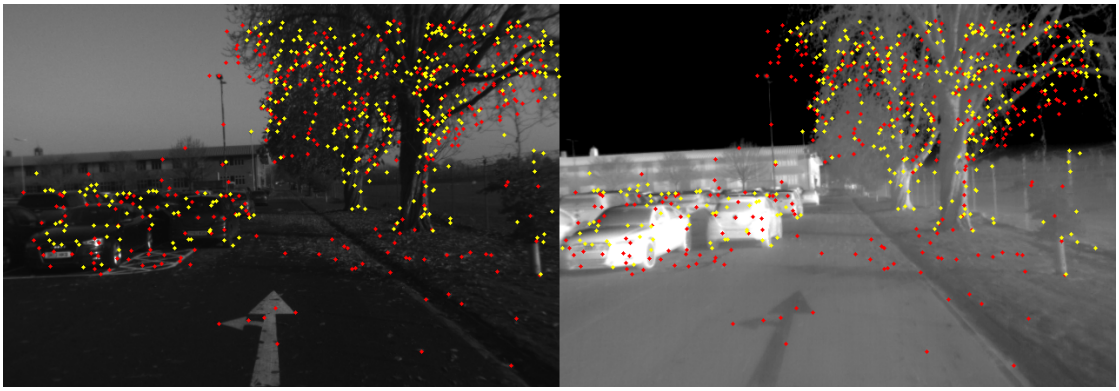
For this example the pose estimation and 3D reconstruction were performed on a window of 7 image pairs. An initial scale value of **1.0** (unscaled poses) was selected which returned a similarity score of **51.34** and an optimal scale factor of **0.5315** was obtained, which returned a similarity score of **85.22**. This indeed corresponds to the peak of the objective function shown in Figure 5.10a. Furthermore, a standard deviation of **0.029** was computed from the inverse Hessian, as explained in the previous section. This means that the optimal scale can be recovered accurately with this method.

Trajectory estimation

The performance of the multispectral stereo VO process is evaluated by comparing the trajectories obtained with the ground truth. Figure 5.12a to Figure 5.12c show that the global matching produces a better estimation even though the scale is not always recovered accurately, leading to some drift. Nevertheless, the overall shape of the trajectory is similar to the GPS track (ground truth) unlike the one obtained with the local matching technique, which tends to drift more on the side. This is



(a) Wrong scale estimated.



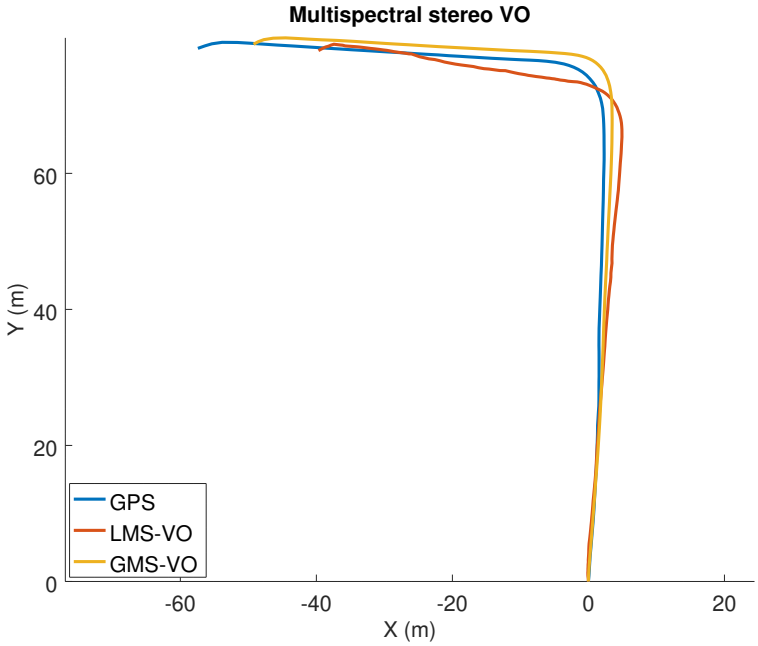
(b) Right scale estimated.

Figure 5.11: Features reprojection in both stereo images based on the scale estimated.

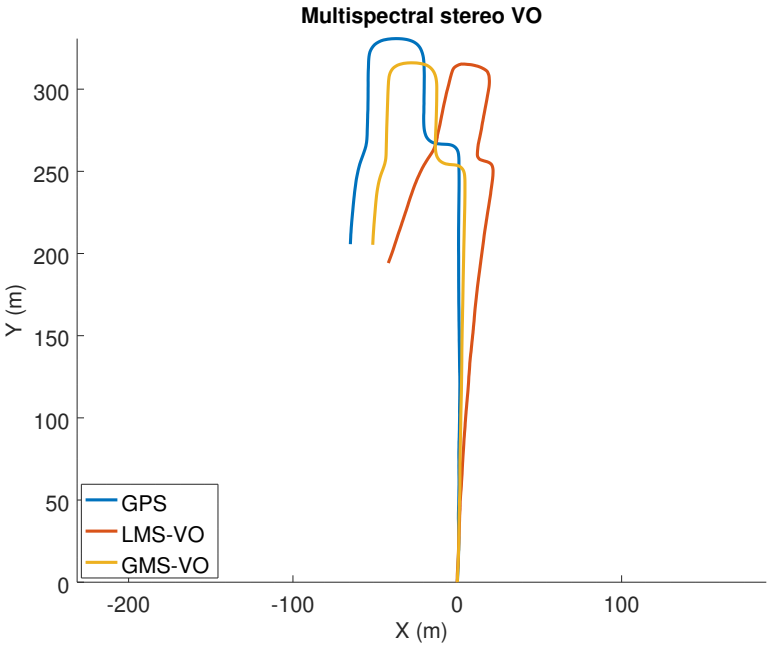
due to the fact that GMS-VO estimates motion from 2D features only and across several frames. Relative orientations and unscaled translations are then computed more accurately. The main source of errors resides in the scale estimation. On the contrary, LMS-VO relies on 3D-to-2D motion estimation which depends greatly on the quality of the matching and the triangulation of the features. As it has been seen throughout this chapter, finding correspondences between multispectral images is a challenging process which is not always accurate. Having said that, both local and global techniques provides satisfying results considering the multispectral challenge that is faced.

In Figure 5.13, the relative error between consecutive keyframes was estimated for each camera pose by computing the Euclidean distance between the translation vector obtained from the GPS track and the ones estimated by the proposed algorithms. It can be noticed that both techniques produce a few peaks which correspond to a poor estimation for the step at which it occurs. Nevertheless, most of the camera pose estimates produce less than **0.5m** errors and the number of poses generating large errors is limited considering that several hundred keyframes were processed for each dataset. Moreover, an increase in relative error can be noticed during turns. Both local and global techniques are affected. As mentioned earlier, this could be coming from motion blur appearing in the images, but also from the reduced number of features tracked due to large optical flow.

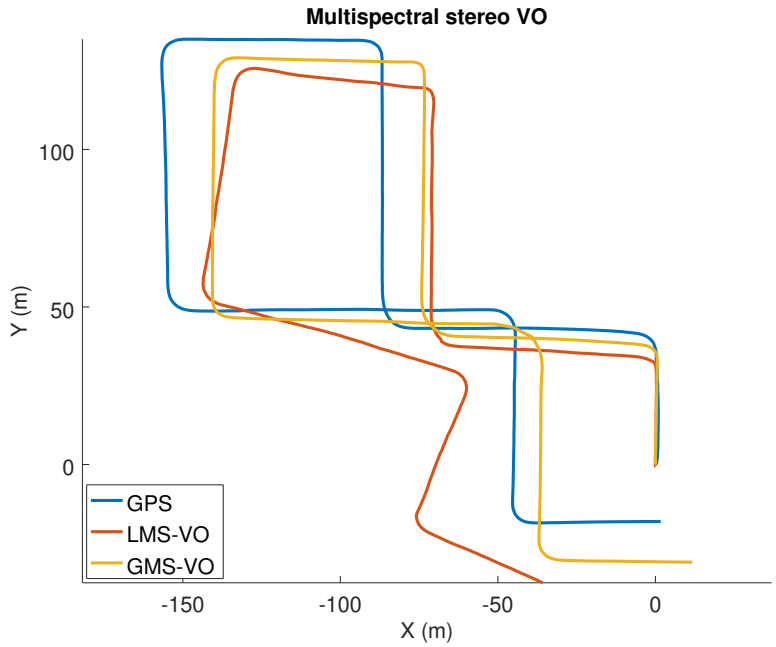
As seen in Table 5.1, LMS-VO tend to produce smaller relative errors and in Figure 5.13 higher peaks can be noticed from the GMS-VO technique. This means that LMS-VO is more accurate locally. However, the absolute error increases more rapidly with the local matching. This confirms the results seen from the full trajectories and shows that even if GMS-VO sometimes produces high relative errors when the scale is not estimated properly, the estimation is still consistent over time. As mentioned before, running bundle adjustment produces a more



(a) Sequence 1.



(b) Sequence 2.



(c) Sequence 3.

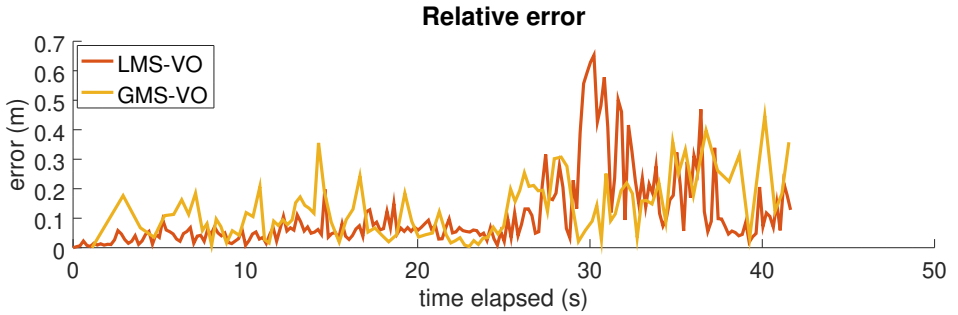
Figure 5.12: Multispectral stereo VO full trajectories compared to GPS.

accurate motion estimation and most of the errors are produced while estimating the scale. On the contrary, LMS-VO produces errors which are accumulating more rapidly over time because both errors in the magnitude and direction of the translation vector are generated.

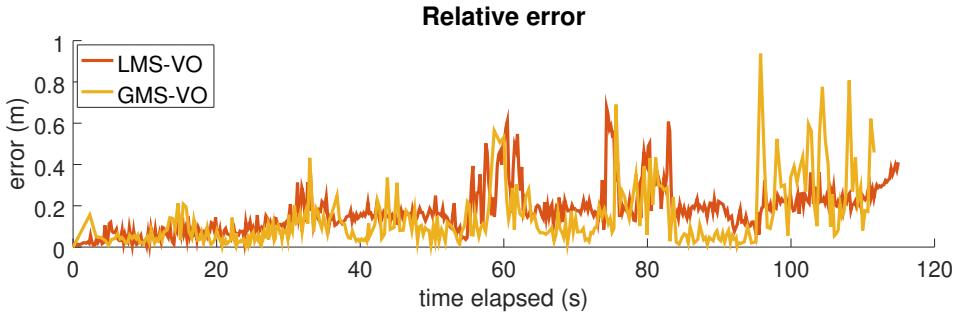
Table 5.2 compiles the final and mean errors obtained on all datasets. Overall, both methods provide final errors under or close to 10% of the distance travelled and mean errors under 5%.

Covariance estimation

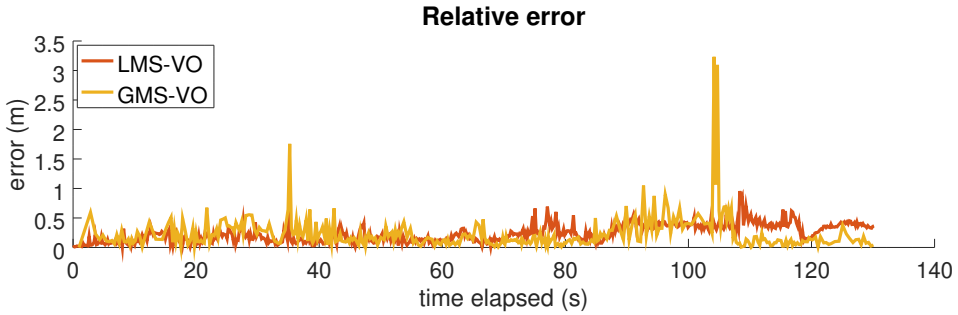
In order to check that the covariance is estimated properly, the uncertainties of the current camera pose are displayed in Figure 5.14. More specifically, the 2D ellipses corresponding to the uncertainties in position along the x and y axes are shown. It can be noted that the covariance ellipse is narrow at the start, when the



(a) Sequence 1.



(b) Sequence 2.



(c) Sequence 3.

Figure 5.13: Multispectral stereo VO relative errors.

Table 5.1: MS-VO relative and absolute errors comparison. Bold values represent the best performance obtained for each dataset.

	LMS-VO		GMS-VO	
	Rel. error	Abs. error	Rel. error	Abs. error
Sequence 1	133m			
Mean (m)	**0.11**	5.55	0.13	**2.77**
Std (m)	0.13	5.99	**0.10**	**2.57**
Sequence 2	511m			
Mean (m)	0.18	24.48	**0.12**	**9.31**
Std (m)	**0.11**	17.39	0.14	**6.02**
Sequence 3	581m			
Mean (m)	**0.24**	17.17	0.26	**12.25**
Std (m)	**0.15**	10.04	0.29	**5.31**

Table 5.2: MS-VO final and mean distance errors. Bold values represent the best performance obtained for each dataset.

	LMS-VO	GMS-VO
Distance of Sequence 1	133m	
Final error (m)	17.99	**8.20**
Final error (%)	13.53	**6.17**
Mean error (m)	5.55	**2.77**
Mean error (%)	4.14	**2.08**
Distance of Sequence 2	511m	
Final error (m)	24.25	**13.62**
Final error (%)	4.74	**2.67**
Mean error (m)	24.48	**9.31**
Mean error (%)	4.79	**1.82**
Distance of Sequence 3	581m	
Final error (m)	42.66	**16.33**
Final error (%)	7.34	**2.81**
Mean error (m)	17.17	**12.25**
Mean error (%)	2.96	**2.11**

car is travelling in a straight line. Its principal axis corresponds to the direction of travel. As the vehicle is turning, the covariance is also increasing in the y-axis and the ellipse becomes more round. Nevertheless, more uncertainties are generated in the x-axis over the whole dataset as the car is mostly moving in this direction.

Similarly, Figure 5.15 shows the diagonal elements of the covariance matrix for position. As mentioned previously, covariance is increasing in x-axis throughout the dataset. However, it can be noted that covariance is only increasing significantly in y-axis when moving along this axis (around keyframes 15 and 35), corresponding to positions (-10m,60m) and (-40m,120m) respectively. Finally, covariance along the z-axis stays low all the time as the altitude of the car is not changing. This confirms that the covariance matrix is estimated properly, not only in position but also in orientation as the position of the current pose corresponds to the accumulation of relative transformations and therefore depends on the orientation of previous poses.

These however, only represent one realisation of each trajectory and several parameters have been set empirically such as the number of keyframes in the sliding window, the minimum parallax threshold and the maximum feature shift required to automatically accept a new keyframe. The selection of these parameters could then be optimised in order to find the set of parameters which produces the best results.

Computation time comparison

Now, the performance of both algorithms are evaluated in terms of computation time. They are tested on a computer equipped with a i7-3770 CPU running at 3.4 GHz. Time is measured at each step for every new keyframe and is summarised in the Table 5.3.

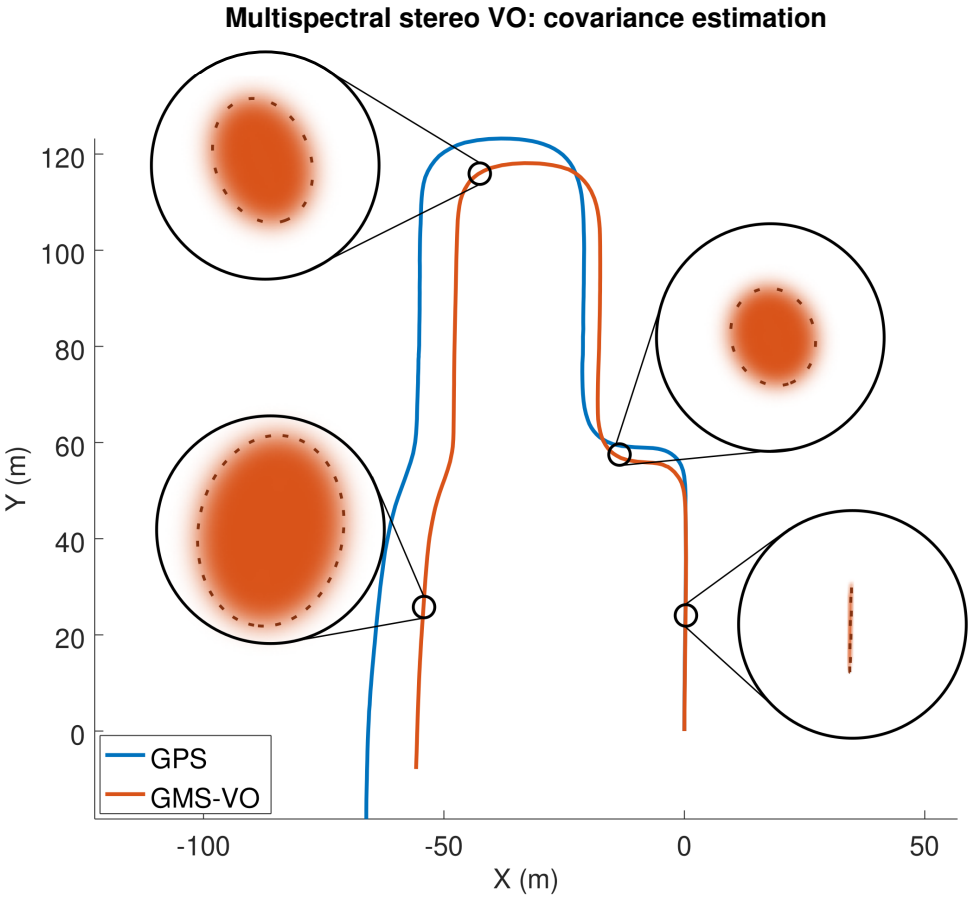


Figure 5.14: 2D covariance ellipses for positioning.

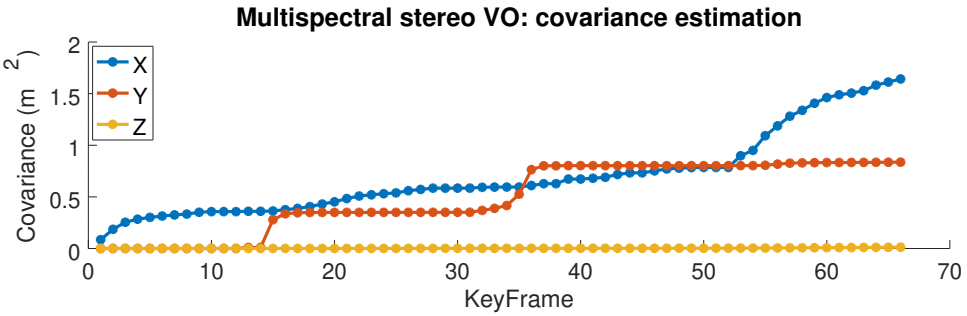


Figure 5.15: Evolution of the position covariance as the vehicle moves.

Table 5.3: Speed comparison between LMS-VO and GMS-VO.

Seq 1: LMS-VO	PC computation	feat. detection	feat. matching	Motion Estimation	Total (ms)
max	1633.00	5.00	1943.00	**47.00**	3543.00
min	984.00	2.00	402.00	**25.00**	1454.00
mean	1010.16	2.76	953.79	**38.52**	2006.65
Seq 1: GMS-VO	**PC computation**	**Feature tracking**		**Motion Estimation**	**Total (ms)**
max	**0.00**	**128.62**		559.55	**666.29**
min	**0.00**	**98.55**		175.81	**282.23**
mean	**0.00**	**107.32**		306.74	**414.06**
Seq 2: LMS-VO	**PC computation**	**feat. detection**	**feat. matching**	**Motion Estimation**	**Total (ms)**
max	1649.00	5.00	1720.00	**53.00**	3403.00
min	987.00	2.00	554.00	**29.00**	1600.00
mean	1018.11	3.01	901.48	**38.72**	1962.66
Seq 2: GMS-VO	**PC computation**	**Feature tracking**		**Motion Estimation**	**Total (ms)**
max	**0.00**	**126.09**		825.42	**930.74**
min	**0.00**	**98.70**		182.91	**290.62**
mean	**0.00**	**108.24**		305.82	**414.07**
Seq 3: LMS-VO	**PC computation**	**feat. detection**	**feat. matching**	**Motion Estimation**	**Total (ms)**
max	1783.00	5.00	1755.00	**53.00**	3582.00
min	1001.00	2.00	346.00	**32.00**	1430.00
mean	1082.20	2.78	788.17	**39.91**	1914.51
Seq 3: GMS-VO	**PC computation**	**Feature tracking**		**Motion Estimation**	**Total (ms)**
max	**0.00**	**123.90**		811.56	**915.79**
min	**0.00**	**97.09**		174.82	**275.25**
mean	**0.00**	**107.04**		320.39	**427.43**

Looking at the total amount of time it takes to process a new pair of images, it is clear that the global matching technique is a lot faster than the local one, about 5 times faster on average. This is partially due to the costly computation of phase congruency images which takes about one second per pair of images and that is not necessary for GMS-VO. Feature detection is fast but the matching part is heavy and remains a bottleneck for LMS-VO. The slow speed is also due to the number of features required in order to get sufficient matches to apply RANSAC and obtain decent motion estimation. On the other hand, feature tracking is much faster and produces less outliers. However, once the local matching is performed, motion estimation is straightforward because the 3D features are directly triangulated from the matches and not optimised in the motion estimation process. Whereas global matching requires more time to apply the sparse bundle adjustment algorithm and the additional optimisation process to recover the scale. Moreover, the global matching technique also requires a certain number of features spread in the image to estimate the scale accurately, and the more features, the longer it takes to evaluate the scale objective function. Hence, mutual information might be fast enough to compute for a few hundred features, but it nevertheless remains a costly similarity measure and can be problematic when considering larger sets of features.

5.3 Conclusion

Two new stereo multispectral VO techniques were presented to tackle the challenging problem of detecting and matching stereo pairs with different modalities. The first one (LMS-VO) matches features locally by searching for stereo correspondents along the epipolar line and assessing their similarities with phase congruency and mutual information. The other one (GMS-VO) tracks features and reconstructs

the 3D scene through a windowed bundle adjustment framework on separate spectra before estimating the missing scale by reprojecting all the features and check the similarity between the images in a global manner.

After testing both algorithms on several car datasets, it has been proven that GMS-VO is much faster than LMS-VO, 5 times faster on average. This is mostly due to the fact that GMS-VO does not require to compute PC images, which is a time consuming step. The speed of the stereo matching process also makes the local matching technique slower. Even though the LMS-VO technique is able to directly compute matches from a single stereo pair, it remains computationally heavy.

In terms of precision, GMS-VO also outperforms LMS-VO. Both methods produce an average error of **5cm** at each iteration but the global approach is more consistent and generates less drift. Overall, maximum final errors of **16m** and **46m** were obtained after 581m with GMS-VO and LMS-VO, respectively. These correspond to **3%** and **7%** of the distance travelled. Finally, under **2%** and **5%** mean errors were obtained for all the image sequences with the same algorithms, respectively, reaching as low as **1.82%** mean error for GMS-VO on Sequence 2.

This confirms that visible-thermal stereo images can be exploited to perform accurate stereo visual odometry. In the next chapter, the use of multispectral imaging for localisation purposes is investigated further by fusing the global VO method presented here with additional data.

6 | Multispectral visual-inertial localisation

In this chapter, a loosely coupled visual-inertial localisation solution is proposed. It fuses inertial data with camera pose measurements obtained from the multispectral stereo odometry algorithm introduced in the previous chapter. The kinematics of the system are introduced first and the fusion strategies are explained afterwards. Making use of an error-state Kalman filter, a state estimate is predicted from high rate inertial data and corrected with lower rate camera pose measurements. Two methods are proposed. The first one uses absolute pose measurements to correct the latest state prediction whereas the second one utilises relative transformations between camera poses to correct the state.

6.1 Introduction

With the development of *Micro Electro-Mechanical Sensors* (MEMS), *Inertial Measurement Units* (IMUs) have become popular sensors in robotics and are available in almost every mobile platform. Indeed, IMUs provide reliable measurements concerning the acceleration, angular velocity and magnetic field applied to the system. MEMS are particularly interesting because they are small, light-weight and supply high-rate information while requiring a very low power input. However, due to the amount of noise included in the measurements, these IMU data are only valid for a short period of time. As a consequence, the pose estimate obtained from the strapdown integration drifts over time and becomes rapidly unreliable if not corrected properly. This is especially the case for position estimation which necessitates to integrate acceleration measurements twice, including noises generated by the sensor. This results in significant errors accumulating over time and large uncertainties in the prediction process.

While IMUs sense a propriocetive input to the system, different kind of information can be extracted from external sources present in the vehicle environment. For example, visual and laser sensors can perceive the structure of the scene and provide a more accurate localisation in the long run [110, 11]. With images, more complex algorithms can be utilised (PnP, BA, etc...) as presented in the previous chapters. Even though VO and V-SLAM are also dead reckoning processes, their accuracy is greater than inertial data alone. However, the more information is being processed, the more time-consuming the technique becomes and the less frequent the camera poses are being generated. Nowadays, image processing algorithms can be optimised drastically and simple visual odometry programs can be run in real-time, between 15 and 30 fps [111, 112, 98]. Yet, this is not sufficient to control autonomous systems precisely. Such systems usually require a much higher

processing rate. Moreover, with the improvement of visual sensors, the increase in image resolution also adds up to the amount of information being processed. Hence, more time is required to process each image, especially on embedded systems with limited hardware. This is why inertial sensors are appealing for control and navigation of mobile platforms [1], due to their simple and fast processing they can provide measurements up to several hundred Hertz.

Therefore, the combination of inertial data and visual odometry has become a popular approach in the recent years [73]. Because inertial data generate accurate short-term localisation estimates at high frequency, while cameras provide corrective measurements at a lower rate, the drift of inertial data can be reduced to ensure more robust pose estimates [113]. Obviously, as VO is also accumulating errors, the estimation is not corrected completely. Nevertheless, VO is more reliable than IMUs in the long run. To reduce the drift further, one can additionally use global pose estimates, provided by GNSS or maps for example. For the same reasons, GNNS+INS is another popular localisation approach [114, 115, 116].

Another advantage of visual-inertial navigation systems is the fact that IMU measurements are always available, unlike VO which can fail for several reasons. Therefore, inertial data can still be integrated during the period of time where VO poses are not available. Thus, a pose estimate can be computed all the time, without interruption.

6.2 Related Work

In the literature, two main visual-inertial odometry (VIO) approaches can be distinguished. The first one includes loosely-coupled solutions [73, 75, 83, 71]. Such algorithms separate completely the visual odometry process from the integration of inertial data. Because VO and inertial data are treated independently, there is

no need to modify the VO algorithm and the resulting pose estimate can be fused directly with IMU measurements. Even though any VO or V-SLAM algorithm could be employed, most loosely-coupled techniques rely on the fast and robust PTAM algorithm [82], which simultaneously tracks features and builds a map of the environment in real-time [73, 76, 83]. Another advantage of decoupling the VO part from the fusion scheme is that both algorithms can be run in parallel, improving the overall speed performance of the solution. Furthermore, the VO/V-SLAM method can be changed depending on the application and other sensors can easily be added to the fusion scheme [76]. For these reasons, loosely-coupled methods are easier to implement, more modular and usually run faster than tightly-coupled methods.

On the other hand, tightly-coupled solutions jointly optimise camera poses, feature positions and IMU parameters while including some constraints generated by the integration of inertial data. The location of the image features can either be expressed in 3D [14, 117] in a global reference frame, or in 2D in the image plane [10, 74]. Because VO and IMU parameters are estimated together, the correlation between data from different sources is increased, leading to a more robust and precise estimation. Coupling IMU and VO can be performed in different ways. In [14, 117], filter-based methods are selected, whereas the optimisation-based approach is preferred in [11, 118].

Different types of setup have been investigated to perform VIO with different platforms. The monocular approach is preferred for embedded systems with limited space and sensors such as UAVs [118, 83, 75, 119] or hand held cameras and smartphones [10, 120, 12]. For monocular navigation, IMUs can provide measurements about the external world like accelerations, and especially the gravity vector. With these data, it is possible to solve the scale ambiguity problem by augmenting the state of the system with a scale factor which is corrected along

with the other state parameters. Stereo visual sensors could also be used to provide more accurate visual measurements and to avoid estimating the scale. Such systems are usually seen on bigger platforms like cars [14, 11, 16] even tough they were tested on smaller and light-weight platforms as well [117].

Initialisation is a critical part of any VINS algorithm because a slight errors in IMU biases or initial velocity will propagate considerably and leading to dramatic repercussions, like a filter or optimisation which diverges. This is especially the case for monocular systems where the scale parameter is missing. Several works have addressed this issue. In [8], a closed-form solution is proposed to compute visual-inertial Structure from Motion (SfM) and estimate the IMU parameters in the same time. It is also shown that certain types of motion can lead to an infinite number of solutions, making the initialisation impossible. Using a bundle adjustment approach, [118] performs the initialisation in an optimisation process which minimises a photometric error rather than the geometric error employed by [8]. In both cases, the IMU measurements recorded between the first visual frames are preintegrated to provide some motion constraints. Finally, [83, 121] use the same IMU preintegration approach, but gyroscope biases are estimated independently from the velocity, and the scale. The gravity vector is also estimated in the process. However, it is not explained how the accelerometer biases are initialised.

6.3 IMU noise modelling and kinematics in continuous time

Inertial sensors, and especially low-cost IMUs, provide noisy acceleration and angular velocity measurements which can be modelled with different sources of noise. First, IMU readings are disturbed by a rapidly changing noise represented by an

additive white noise and denoted by n. It is modelled by a continuous-time white Gaussian distribution whose mean is zero and variance is equivalent to σ^2:

$$\begin{aligned} n_a(t) &\sim \mathcal{N}(0, \sigma_a^2) \\ n_\omega(t) &\sim \mathcal{N}(0, \sigma_\omega^2) \end{aligned} \tag{6.1}$$

This white noise is also commonly called *noise density*. The second source of perturbation is the sensor bias, also named *bias stability*. It corresponds to an offset in the measurement that needs to be accounted for in order to obtain accurate pose estimation. The IMU bias varies slowly compared to the noise density and can even be considered constant for short-term applications as it is not fluctuating much. It eventually varies due to internal factors such as changes in the sensor temperature. Therefore, these variations need to be estimated if the IMU is used for an extended period of time. The noise generated by this sensor bias is modelled by a *Wiener process* which corresponds to the integration of a white Gaussian noise of variance σ_b^2:

$$\begin{aligned} \dot{b}_a(t) &= n_{b_a}(t), \quad n_{b_a} \sim \mathcal{N}(0, \sigma_{b_a}^2) \\ \dot{b}_\omega(t) &= n_{b_\omega}(t), \quad n_{b_\omega} \sim \mathcal{N}(0, \sigma_{b_\omega}^2) \end{aligned} \tag{6.2}$$

Accelerations and angular velocities are measured in the local coordinate frame of the sensor. Considering an IMU moving in a global inertial frame, these measurements can be expressed at an instant t as:

$$\boldsymbol{\omega}_m(t) = \boldsymbol{\omega}(t) + \mathbf{b}_\omega(t) + \mathbf{n}_\omega(t) \tag{6.3}$$

$$\mathbf{a}_m(t) = \mathbf{q}^* \, (\mathbf{a}(t) - \mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a(t) \tag{6.4}$$

where $\mathbf{a}(t)$ and $\boldsymbol{\omega}(t)$ correspond respectively to the true linear accelerations and angular rates, $\mathbf{b}_a(t)$ and $\mathbf{b}_\omega(t)$ to the accelerometer and gyroscope biases, $\mathbf{n}_\omega(t)$

and $\mathbf{n}_a(t)$ to white Gaussian noises. It is important to notice that $\boldsymbol{\omega}(t)$ is expressed in the local IMU frame whereas $\mathbf{a}(t)$ and $\mathbf{g}$ are expressed in the global frame. $\mathbf{q}$ corresponds to the rotation between the IMU coordinate frame and the world coordinate frame. From (6.3), the true acceleration and angular rate can be derived:

$$\boldsymbol{\omega}(t) = \boldsymbol{\omega}_m(t) - \mathbf{b}_\omega(t) - \mathbf{n}_\omega(t) \tag{6.5}$$
$$\mathbf{a}(t) = \mathbf{q}\left(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)\right) + \mathbf{g} \tag{6.6}$$

Because angular rates are expressed in the local frame, quaternion time derivative are expressed as follows:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \tag{6.7}$$

The kinematics of the system in continuous time can then be defined as:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{6.8}$$
$$\dot{\mathbf{v}} = \mathbf{a} \tag{6.9}$$
$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \tag{6.10}$$
$$\dot{\mathbf{b}_a} = \mathbf{n}_{b_a} \tag{6.11}$$
$$\dot{\mathbf{b}_\omega} = \mathbf{n}_{b_\omega} \tag{6.12}$$

where $\mathbf{p}$, $\mathbf{v}$ and $\mathbf{q}$ are the position, velocity and orientation of the IMU in the world frame, respectively.

6.4 Loosely-coupled visual-inertial fusion

Having addressed the multispectral challenge first, the use of a loosely-coupled approach came as an obvious choice. Indeed, the multispectral matching techniques and thus multispectral VO processes presented in the previous chapter are already computationally heavy. Hence, implementing a loosely-couple fusion scheme allows to keep the core of the VO algorithm unchanged and does not add excessive processing load to the existing algorithm. Moreover, the main advantage of multispectral setups being its robustness to extreme cases, handling inertial data integration and VO separately makes it easier to manage failure cases, as explained in Section 6.5.3.

Therefore, a filter-based visual-inertial localisation solution, inspired by the work in [76, 75] is presented in this section. An introduction to Kalman filtering is given first, before detailing the process model as well as absolute and relative measurement models.

6.4.1 Extended Kalman filter

The standard Kalman filter is a recursive filter which aims at estimating the state of a linear dynamic system from several measurements, including their uncertainties (noise). It produces state estimates more accurate than the ones obtained from the measurements alone. The filter was originally developed by Rudolf E. Kalman in 1960 to provide a solution to the discrete linear filtering problem [122]. It has become popular in fields such as aeronautics, aero-spatial and more recently robotics [123]. The Kalman filter is an optimal filter for processes including Gaussian noises, linear transition functions and measurements which can be expressed as linear functions of the state. Therefore, the original version of the filter does not guarantee the optimality of the estimator for non-linear dynamic systems, and

thus it is not suited for the estimation of 3D poses or the integration of inertial data due to the non-linearities generated by the rotations. To tackle such problems, other versions of the filter were developed, such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [124]. In this chapter, the EKF version was selected. It consists in linearising the process and measurement models around the current state at each iteration.

6.4.2 The extended Kalman filter framework

Because of its recursive nature, the filter solely estimates the *a posteriori* state $\hat{\mathbf{x}}_{k|k}$ and its covariance matrix $P_{k|k}$ at iteration k from the estimate at the previous iteration $\hat{\mathbf{x}}_{k-1|k-1}$ and observations $\mathbf{z}_k$ at the current iteration. It represents a Markov process, which means that all information about the previous states are included in the current one. The KF framework consists of two main steps:

- a prediction step, where the previously estimated state is propagated according to a transition model, called process model:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{6.13}$$

with f the transition function, $\mathbf{u}_k$ an input vector usually coming from the control command, and $\mathbf{w}_k$ the process noise represented as a zero mean multivariable Gaussian noise with covariance Q_k.

- a correction step, where a function of the state is observed according to the observation model and used to correct the predicted state:

$$\mathbf{z}_k = h(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{v}_k \tag{6.14}$$

with h and $\mathbf{v}_k$ the observation function and noise, respectively. $\mathbf{v}_k$ is represented as a zero mean multivariable Gaussian noise with covariance R_k.

6.4.3 Visual-inertial filtering

For linear systems, f and h and can be directly expressed in matrix form with matrices F_k and H_k respectively. However, this does not apply to the inertial system kinematics and the measurements provided by VO. Therefore, the propagation and observation models need to be linearised. As illustrated in Figure 6.1, the state and its uncertainties are propagated as follows:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \tag{6.15}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{6.16}$$

where $\mathbf{u}_k = [\mathbf{a}_k^T, \boldsymbol{\omega}_k^T]^T$ corresponds to the inertial vector. Matrix Q_k represents the uncertainties involved in the process model at step k. The state is propagated each time a new IMU measurement is received and uncertainties in the state estimate are accumulated in $P_{k|k-1}$.

When a new camera pose $\mathbf{z}_k = [\mathbf{p}_k^T, \mathbf{q}_k^T]^T$ becomes available, it is used to correct the predicted state. A residual term, also called innovation, and its covariance are computed:

$$\mathbf{y}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{6.17}$$
$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{6.18}$$

Then, the Kalman gain is calculated. Based on the covariance of the *a priori* state estimate $\hat{\mathbf{x}}_{k|k-1}$ and the covariance of the measurement $\mathbf{z}_k$, it corresponds to a weight which is applied to the corrective term, so the state follows the behaviour of the most reliable source of information. It is obtained with the following formula:

$$K = P_{k|k-1} H_k^T S_k^{-1} \tag{6.19}$$

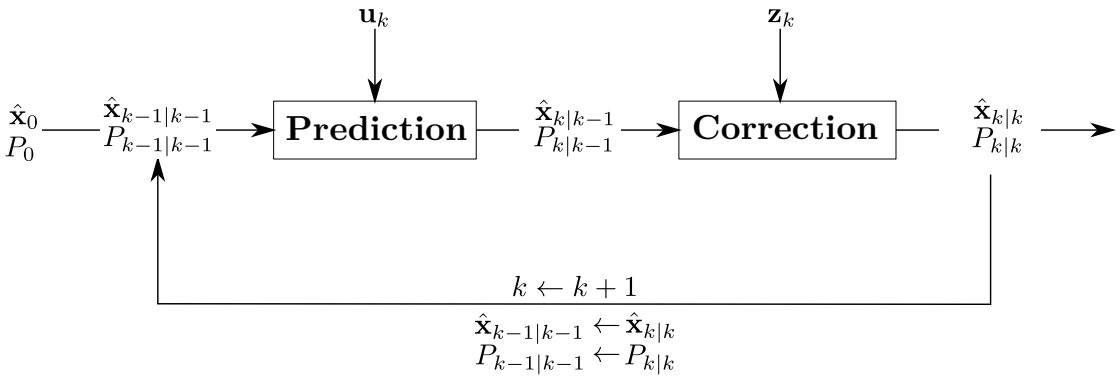


Figure 6.1: The general Kalman filter framework.

The *a posteriori* state estimate $\hat{\mathbf{x}}_{k|k}$ can then be computed as follows:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \mathbf{y}_k \tag{6.20}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T \tag{6.21}$$

Both F_k and H_k are obtained by linearising f and h, respectively, at the current state estimate, using the first order Taylor series expansion:

$$F_k = \frac{\partial f}{\partial x}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}} \qquad H_k = \frac{\partial h}{\partial x}\Big|_{\hat{\mathbf{x}}_{k|k-1}} \tag{6.22}$$

6.4.4 State modelling

For VINS estimation, the state includes the vehicle pose, velocity and IMU biases. It is propagated by integrating the inertial data as they are received. When a new pose is generated by the VO process, the uncertainties of both predicted and actual measurements are weighted optimally by the Kalman filter in order to compute a weighted innovation term. This term is then used to correct the state estimate.

The state vector is defined as:

$$\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{q}^T, \mathbf{b}_a^T, \mathbf{b}_\omega^T]^T \tag{6.23}$$

where $\mathbf{q}$ is the quaternion corresponding to the vehicle orientation whereas $\mathbf{p}, \mathbf{v}, \mathbf{b}_a$ and $\mathbf{b}_\omega$ are 3D vectors corresponding to the vehicle position, velocity, accelerometer bias and gyroscope bias, respectively.

As stated in the previous section, the linear KF is rarely used directly for pose estimation. In many works, an indirect formulation is preferred [73, 76, 83, 14, 125]. This approach is known as *indirect Kalman filter* or *error-state Kalman filter*. The error-state representation consists in propagating an *estimated-state* (or *nominal-state*) which does not account for the system noises and other model inaccuracies. Instead of estimating the state directly, noises and perturbations are estimated in an *error-state* whose magnitude is much smaller than the estimated state. Concerning VIO, inertial data are simply integrated to predict the state of the system and errors are propagated in the error-state. Therefore, the estimated uncertainties grow with time. When a measurement becomes available (camera pose), it is then used to correct the error-state. The new error-state mean can then be utilised to correct the estimated state. It is reset to zero afterwards.

The full-state, estimated-state and error-state are related by the following relationship:

$$\mathbf{x} = \hat{\mathbf{x}} \oplus \delta\mathbf{x} \tag{6.24}$$

$$\begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \\ \mathbf{b}_a \\ \mathbf{b}_\omega \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{p}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{q}} \\ \hat{\mathbf{b}}_a \\ \hat{\mathbf{b}}_\omega \end{bmatrix} \oplus \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{v} \\ \delta\boldsymbol{\theta} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \end{bmatrix} \tag{6.25}$$

It is important to notice that the state vectors cannot be added directly due to

the presence of quaternions. Thus, the operator $\oplus$ is used to represent a generic composition between two states.

The error-state filter offers several advantages compared to the standard EKF. First, all the system dynamics are included in the estimated state, attenuating the errors due to large signals. Secondly, the error-state is always small and varies slowly. As a consequence, linearisation errors are reduced and Jacobians are faster to compute as higher order terms can be ignored. Finally, being close to the origin, the orientation errors can be parametrised with a minimal form, avoiding singularities and constraints. In (6.24) for instance, the 3D rotation vector $\delta\boldsymbol{\theta}$, containing 3 components, is employed to represent errors in orientation. These errors are expressed as a quaternion difference:

$$\begin{aligned} \delta\mathbf{q} &= \mathbf{q} \ominus \hat{\mathbf{q}} \\ \delta\mathbf{q} &= \hat{\mathbf{q}}^* \otimes \mathbf{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} \end{aligned} \tag{6.26}$$

The kinematics of the estimated-state can then be derived from (6.8) by removing the noise terms:

$$\dot{\hat{\mathbf{p}}} = \mathbf{v} \tag{6.27}$$

$$\dot{\hat{\mathbf{v}}} = R(\mathbf{a}_m - \mathbf{a}_b) - \mathbf{g} \tag{6.28}$$

$$\dot{\hat{\mathbf{q}}} = \frac{1}{2}\mathbf{q} \otimes \Omega(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b) \tag{6.29}$$

$$\dot{\hat{\mathbf{b}}}_a = \mathbf{0} \tag{6.30}$$

$$\dot{\hat{\mathbf{b}}}_\omega = \mathbf{0} \tag{6.31}$$

The kinematics of the error-state are also derived from (6.8) as follows:

$$\dot{\delta\mathbf{p}} = \delta\mathbf{v} \tag{6.32}$$
$$\dot{\delta\mathbf{v}} = -R[\mathbf{a}_m - \mathbf{a}]_\times \delta\boldsymbol{\theta} - R\delta\mathbf{a}_b - R\mathbf{n}_a \tag{6.33}$$
$$\dot{\delta\boldsymbol{\theta}} = -[\boldsymbol{\omega}_a - \boldsymbol{\omega}_b]_\times \delta\boldsymbol{\omega} - \delta\boldsymbol{\omega}_b - \mathbf{n}_\omega \tag{6.34}$$
$$\dot{\delta\mathbf{b}}_a = \mathbf{n}_{b_a} \tag{6.35}$$
$$\dot{\delta\mathbf{b}}_\omega = \mathbf{n}_{b_\omega} \tag{6.36}$$

where R is the rotation matrix corresponding to the attitude of the vehicle.

6.4.5 Process model and error propagation

(6.32) can be expressed in a linear form such as:

$$\dot{\delta\mathbf{x}} = F_c\ \delta\mathbf{x} + G_c\ \mathbf{n} \tag{6.37}$$

where $\mathbf{n} = [n_a^T, n_{b_a}^T, n_\omega^T, n_{b_\omega}^T]^T$

To obtain the discrete-time update described in (6.15), (6.32) is integrated as follows:

$$F_d = e^{F_c \Delta t} = I + F_c\Delta t + \frac{1}{2}F_c^2\Delta t^2 + \frac{1}{3!}F_c^3\Delta t^3 + \cdots = \sum_{k=0}^{\infty}\frac{1}{k!}F_c^k\Delta t^k \tag{6.38}$$
$$Q_d = \int_{\Delta t} F_d(\tau)G_cQ_cG_c^T F_d(\tau)^T d\tau \tag{6.39}$$

In the equations above, F_c and G_c are assumed to be constant in the time interval Δt. This assumption is valid since the IMU provides measurements at a high rate and thus Δt is small. This way, F_d and Q_d can be computed quickly in a closed-form [75, 125].

To sum up, the state propagation step is performed by:

1. integrating (6.32) to update the estimated-state

2. computing F_d and G_d from (6.38)

3. propagating the error-state covariance according to (6.40)

Which can be summarised with the following equations:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &\leftarrow f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \\ \delta\mathbf{x}_{k|k-1} &\leftarrow F_d \; \delta\mathbf{x}_{k-1|k-1} \\ P_{k|k-1} &\leftarrow F_d \; P_{k-1|k-1} \; F_d^T + Q_d \end{aligned} \tag{6.40}$$

6.4.6 Measurement model

The VO process provides a partial observation of the full-state, namely the position and attitude of the vehicle. However, this observation also contains uncertainties, represented by a white Gaussian noise v. Thus, the measurement model can be defined as:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_p \\ \mathbf{z}_q \end{bmatrix} = h(\mathbf{x}_{k|k-1}) + v_k, \quad v_k \sim \mathcal{N}(0, R_k) \tag{6.41}$$

Usually R is selected to match the noise properties of the measurement. However, $\mathbf{z}_k$ is estimated in a BA framework and therefore its covariance can be calculated in the optimisation process.

As stated in (6.17), the innovation is computed by taking the difference between the observed vector and a predicted vector, function of the predicted state. However, this equation is not valid when dealing with quaternions because both vectors cannot be subtracted directly. Instead, the inverse composition needs to be used:

$$\mathbf{y}_k = \mathbf{z}_k \ominus h(\mathbf{x}_k) \neq \mathbf{z}_k - h(\mathbf{x}_k) \tag{6.42}$$

From (6.41), an error observation can be defined as follow:

$$\mathbf{z} = h(\mathbf{x}) = h(\hat{\mathbf{x}} \oplus \delta\mathbf{x}) \tag{6.43}$$

$$\mathbf{z} = h(\hat{\mathbf{x}}) \oplus \frac{\partial h}{\partial \delta \mathbf{x}} \delta\mathbf{x} \tag{6.44}$$

$$\delta\mathbf{z} = \mathbf{z} \ominus h(\hat{\mathbf{x}}) = H\delta\mathbf{x} \tag{6.45}$$

This means that $\delta\mathbf{z}$ can be expressed as a linear combination of the error-state.

Absolute camera poses

The first type of measurement used to correct the state corresponds to VO absolute camera poses. These are obtained from the VO program, independently from the filtering process. As explained in Section 5.2.4, local poses estimated by windowed bundle adjustment and their corresponding covariances are propagated throughout the estimation to evaluate the current absolute pose. Using such measurements, the observation model becomes:

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \tag{6.46}$$

$$\delta\mathbf{z} = \begin{bmatrix} \mathbf{p} - \hat{\mathbf{p}} \\ \hat{\mathbf{q}}^* \otimes \mathbf{q} \end{bmatrix} = \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{q} \end{bmatrix} \approxeq \begin{bmatrix} \delta\mathbf{p} \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} \tag{6.47}$$

As for the error-state, the error-quaternion in $\delta\mathbf{z}$ can be expressed and used in its minimal form when representing a small angle rotation (see (6.26)). Then, H becomes:

$$H = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{6.48}$$

Relative camera poses

Because VO is a relative motion estimation process, it makes more sense to use the incremental updates between camera poses instead. However, VO poses and their covariances are estimated in a sliding window. To consistently use these measurements, the current state is only corrected with the last pose estimate in the window, at step k, when the last pose of the window at step $k-n$ has become the first pose of the window at step k. This ensures that the VO measurements used where obtained without overlap.

Furthermore, z now represents the current VO pose w.r.t. the first pose of the window in which it has been estimated. Hence, it cannot be directly compared with the absolute state estimate. Instead, a duplicate of the corrected state $\mathbf{x}_{k-1|k-1}$ is created before IMU integration:

$$\mathbf{x}'_k = \mathbf{x}_{k-1|k-1} \tag{6.49}$$
$$P'_k = P_{k-1|k-1} \tag{6.50}$$

The relative transformation $\mathbf{z}$ is then applied to $\mathbf{x}'_k$:

$$\mathbf{z}' = \begin{bmatrix} \mathbf{q}'_k \mathbf{z}_p + \mathbf{p}'_k \\ \mathbf{q}'_k \otimes \mathbf{z}_q \end{bmatrix} \tag{6.51}$$

Instead of propagating VO poses together, the corrected state is now incremented with relative measurements. To stay consistent, the new measurement covariance R' needs to be propagated accordingly:

$$R'_k = J_k \check{R}_k J_k^T \tag{6.52}$$

where

$$\check{R}_k = \begin{bmatrix} P_{k-1|k-1} & 0 \\ 0 & R_k \end{bmatrix} \tag{6.53}$$

$$J_k = \begin{bmatrix} \frac{\partial \mathbf{z}'_p}{\partial \mathbf{p}'_k} & \frac{\partial \mathbf{z}'_p}{\partial \mathbf{q}'_k} & \frac{\partial \mathbf{z}'_p}{\partial \mathbf{z}_p} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{z}'_q}{\partial \mathbf{q}'_k} & \mathbf{0} & \frac{\partial \mathbf{z}'_q}{\partial \mathbf{z}_q} \end{bmatrix} \tag{6.54}$$

J_k represents the Jacobian of the incremental transformation.

$\mathbf{z}'$ being an incremented absolute measurement, it can now be used to correct $\mathbf{x}_k$ with the same measurement model as before:

$$H = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{6.55}$$

6.4.7 Correction and reset

Once $\delta\mathbf{z}$ has been computed, the innovation term and its covariance can be determined. Since the error-state has not been observed yet it can be expressed as:

$$\mathbf{y} = \mathbf{z} - H\delta\mathbf{x} = \delta\mathbf{z} \tag{6.56}$$

$$S = HP_{k|k-1}H^T + R \tag{6.57}$$

which can then be used to correct the error-state using the Kalman filter theory:

$$\begin{aligned} K &= P_{k|k-1}H^T S^{-1} \\ \delta\mathbf{x}_{k|k} &\leftarrow \delta\mathbf{x}_{k|k-1} + K\mathbf{y} \\ P_{k|k} &\leftarrow (I - KH)\ P_{k|k-1}\ (I - KH)^T + KRK^T \end{aligned} \tag{6.58}$$

Concerning relative measurement updates, $\mathbf{y}$ and S are computed with the incremented state **z'** described in the previous section:

$$\mathbf{y} = \mathbf{z}' - H\delta\mathbf{x} \tag{6.59}$$
$$S = HP_{k|k-1}H^T + R' \tag{6.60}$$

When the error-state has been observed and corrected, it is injected into the predicted state and reset to zero:

$$\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k \oplus \delta\mathbf{x}_{k|k} \tag{6.61}$$
$$\delta\mathbf{x}_{k|k} \leftarrow \mathbf{0} \tag{6.62}$$

6.5 Experimental validation

In this section, an experimental evaluation of the error-state filter is performed. The tests are run on the same datasets as in Chapter 4 and Chapter 5. Because it has been proven that the global stereo matching technique is performing better than the local one, the former VO algorithm was retained for this chapter. Therefore, the output of the GMS-VO method is used to correct the state estimate predicted by the filter. The resulting fused trajectory is then compared to the IMU trajectory obtained from strapdown integration and to the GPS trajectory considered as the ground truth. For the first tests, absolute poses are exploited. They are obtained by accumulating the relative transformations generated by VO. Their covariance matrix is propagated accordingly. Then, a second batch of experiments investigates the use of relative transformations as state increments to create corrective measurements.

While measurement covariances are obtained from bundle adjustment (see Sec-

Table 6.1: Xsens MTi-G 710 noise characteristics.

	accelerometer	gyroscope
Noise density ($\sigma_a^2/\sigma_\omega^2$)	$5.89e^{-4}\ m/s^2/\sqrt{Hz}$	$1.75e^{-4}\ rad/s/\sqrt{Hz}$
Bias stability ($\sigma_{b_a}^2/\sigma_{b_\omega}^2$)	$1.47e^{-3}\ m/s^2$	$4.85e^{-4}\ rad/s$

tion 5.2.4), IMU noises also need to be known precisely in order to propagate the state covariance properly and fuse data consistently. In that sense, the IMU white noise and bias noise are extracted directly from its datasheet [126] and used to construct the process noise matrix Q. Their values can be found in Table 6.1.

6.5.1 Position estimation

Absolute measurements

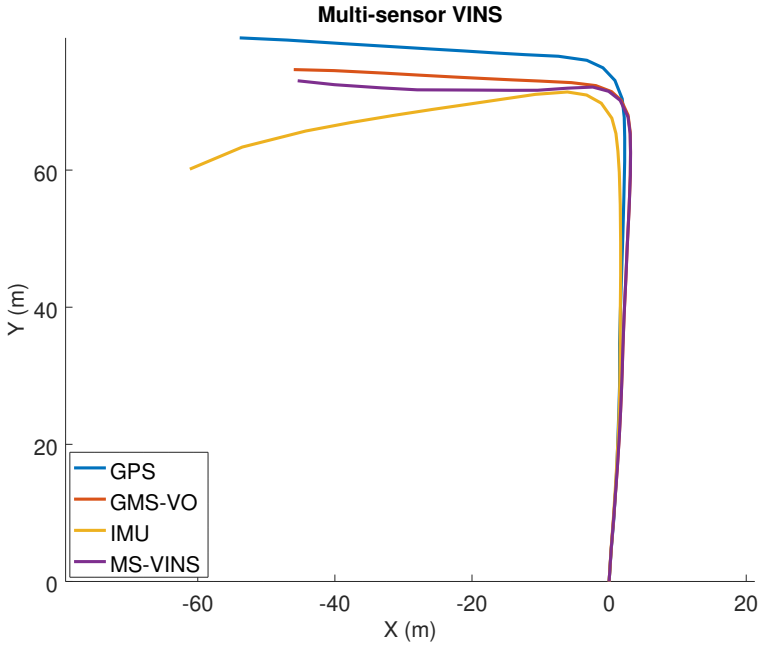
Figure 6.2 shows the fused trajectory obtained through the filter, using the absolute camera pose provided by GMS-VO during the correction step. It can be seen that the predicted state is updated properly in all sequences because the fused trajectory is significantly closer to the ground truth than the one obtained from pure IMU integration. The filtered output clearly follows the VO trajectory as it is a reliable measurement, but it can also be noted that inertial data still have an impact on the final trajectory. This is expected as the filter weights both types of data to produce a fused output. Therefore, the solution is an optimal trade-off between the IMU and VO measurements, but it tends to be closer to the VO trajectory since these pose estimates are more reliable. This result is confirmed by the analysis of the errors, displayed in Table 6.2, where it can be seen that the mean error of MS-VINS is similar to the one obtained with GMS-VO, while IMU integration produces a much larger mean error. Furthermore, It can be noted that the longer and the more complex the trajectory, the more correction is applied. For example, Sequence 3 is the longest sequence and comprises various turns. Yet, it produces

the smallest mean error regarding the distance traveled (**2.06%**). On the contrary, Sequence 1 possesses the simplest and shortest trajectory but produces the largest error (**2.37%**).

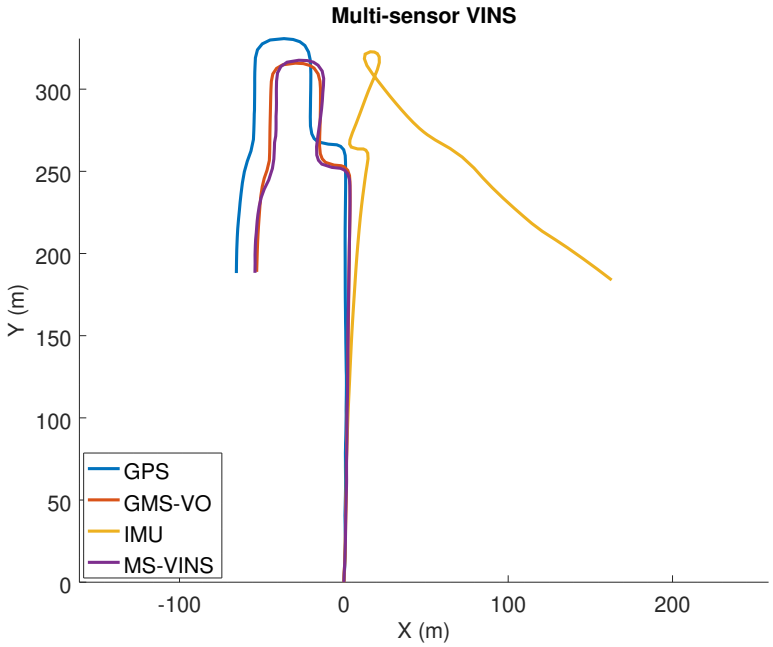
This behaviour can be explained by the fact that if inertial data are never corrected, the integration start drifting significantly after a while and the error in positioning is always increasing. It is particularly clear in Figure 6.3b and Figure 6.3c where it can be seen that relative errors are slowly increasing as time goes by. These errors are much more important at the end of the dataset than at the start. However, they do not grow as much with GMS-VO and MS-VINS. It can even be noted that these errors are even decreased in Figure 6.3c between 70 s and 90 s.

Relative measurements

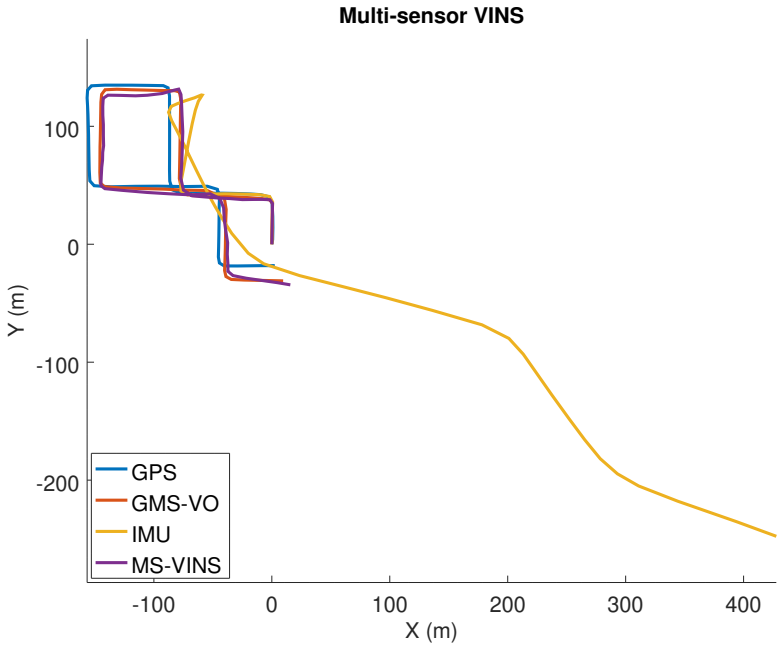
When compared to absolute measurements, it can be seen from Figure 6.4 that the use of relative transformations and their local covariance matrices improves the consistency of the data fusion. Indeed, the shape of the fused trajectory matches the one from VO, which was not always the case with absolute measurements (see Figure 6.2). This is clear in Figure 6.2a, where MS-VINS seems to be drifting on the side, right after the turn. However, this is not the case in Figure 6.4a where the main difference between MS-VINS and GMS-VO appeared to be the distance estimation which varies throughout the dataset. The lateral drift that can be seen with absolute measurements has been completely removed. As it will be explained further in Section 6.5.2, the position covariance of the absolute VO poses encapsulates all previous uncertainties and thus, becomes *round* after a few turns as the uncertainties spread in all directions. This leads to some lateral drift when fused with noisy inertial data. On the other hand, relative transformations obtained from GMS-VO have an extremely small variance in the lateral direction and are propagated to the corrected state. As detailed in the following section,



(a) Sequence 1.

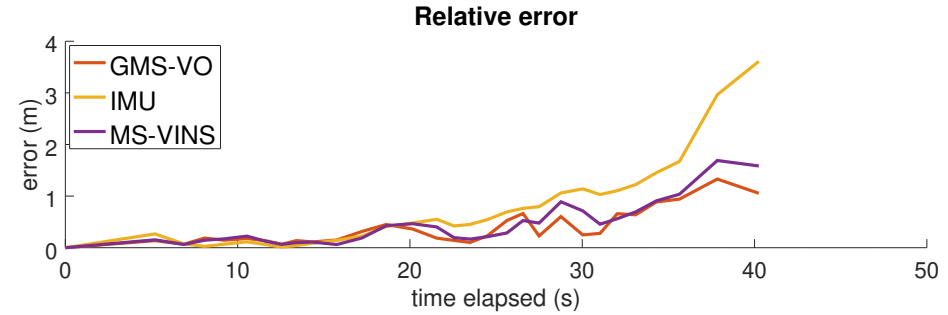


(b) Sequence 2.

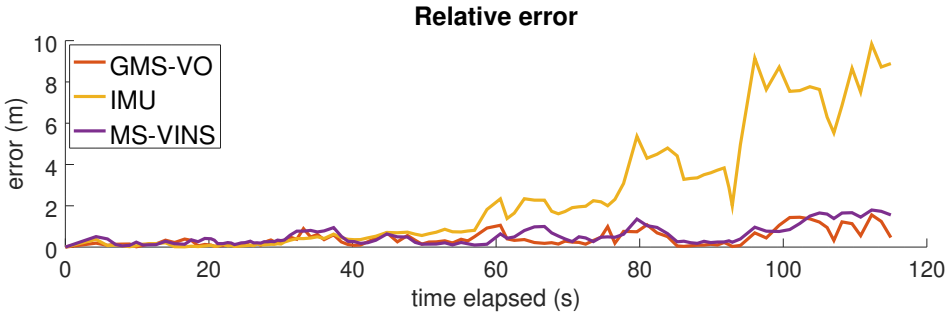


(c) Sequence 3.

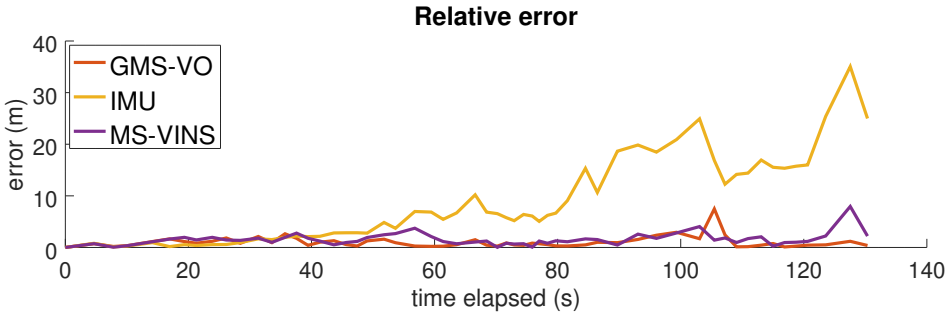
Figure 6.2: Fused trajectories obtained from the visual-inertial error-state filter corrected with absolute measurements.



(a) Sequence 1.



(b) Sequence 2.



(c) Sequence 3.

Figure 6.3: Absolute multispectral VINS relative errors.

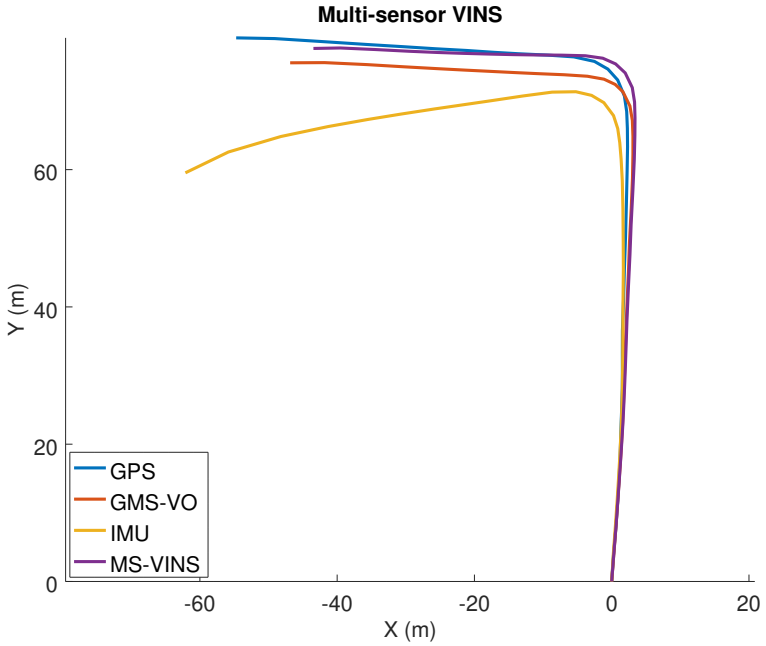
Table 6.2: Error comparison between localisation solutions using both absolute and relative measurements.

	IMU	absolute MS-VINS	relative MS-VINS	GMS-VO
Sequence 1	133m			
mean (m)	4.50	3.15	**2.89**	2.77
std. dev. (m)	5.66	3.09	**2.57**	2.57
mean (%)	3.38	2.37	**2.17**	2.08
Sequence 2	511m			
mean (m)	43.34	**10.63**	11.87	9.31
std. dev. (m)	60.43	**6.97**	7.73	6.02
mean (%)	8.48	**2.08**	2.32	1.82
Sequence 3	581m			
mean (m)	127.69	11.95	**11.18**	12.25
std. dev. (m)	142.42	**4.32**	7.74	5.31
mean (%)	21.98	2.06	**1.92**	2.11

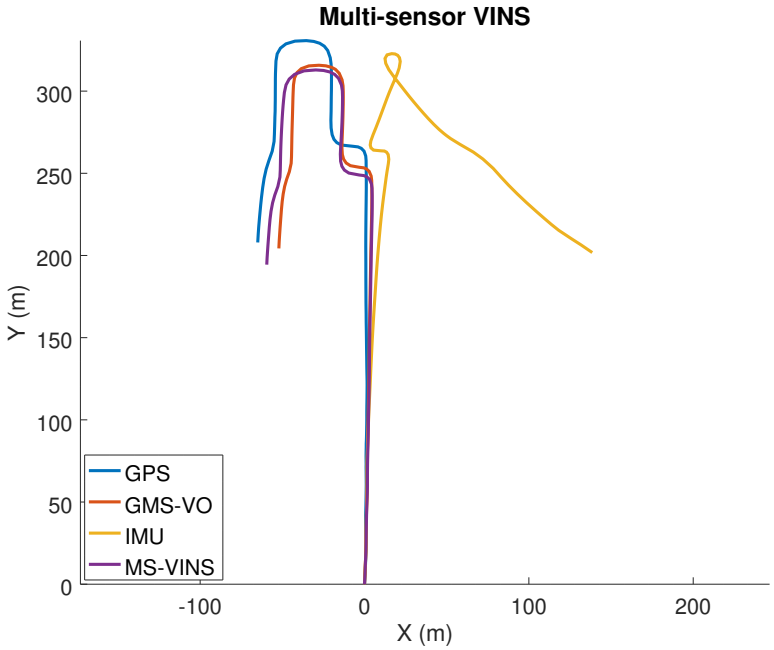
the covariance of the state position varies a lot more and is always much larger in the direction of travel. As a consequence, the lateral drift is considerably reduced.

Table 6.2 also shows that on average MS-VINS performs better when corrected with relative measurements. However, this is not always the case because a smaller mean error is achieved with absolute measurements on the second sequence.

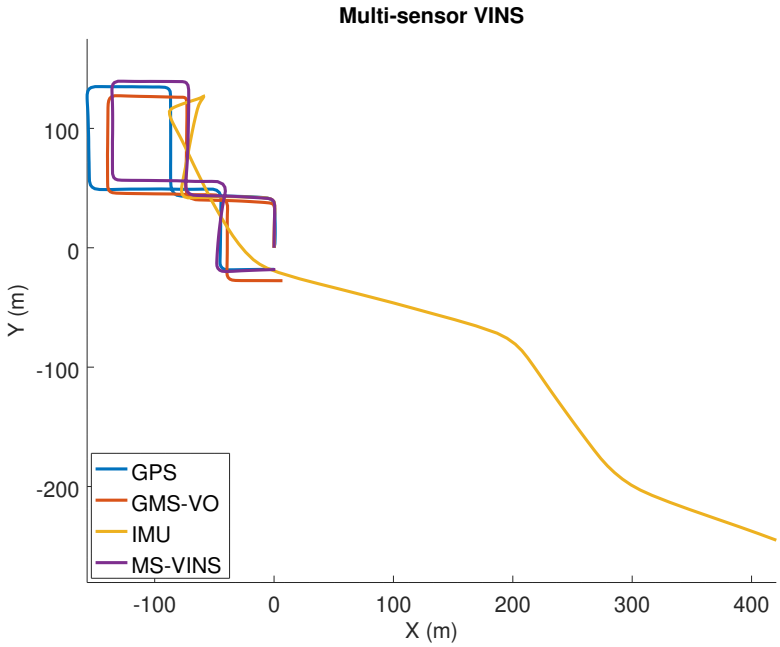
Overall, these results show that the fusion of inertial data and multispectral stereo poses can be performed with both absolute and relative poses. However, each trajectory presented above corresponds to a single trial of a Monte Carlo experiment and further work could be carried out to demonstrate the robustness of the filter. Indeed, the stability of the filter has not been studied (Riccati analysis) and all results are based on experimental data. More data could then be generated to investigate how noises are affecting the filter. Nevertheless, the following section aims at analysing the covariance estimated by the filter in order to understand how uncertainties are taken into account.



(a) Sequence 1.

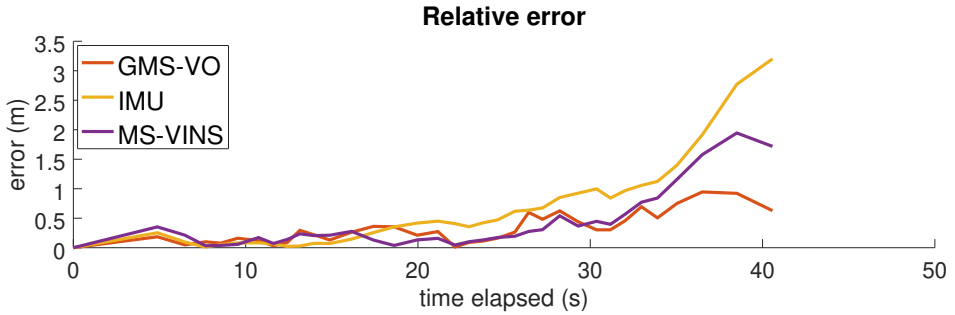


(b) Sequence 2.

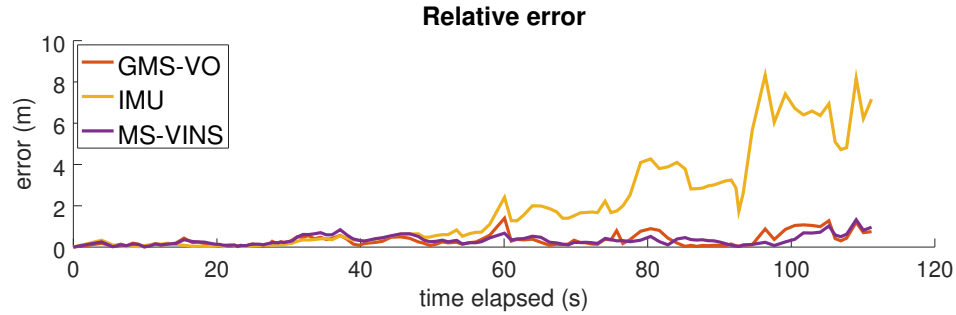


(c) Sequence 3.

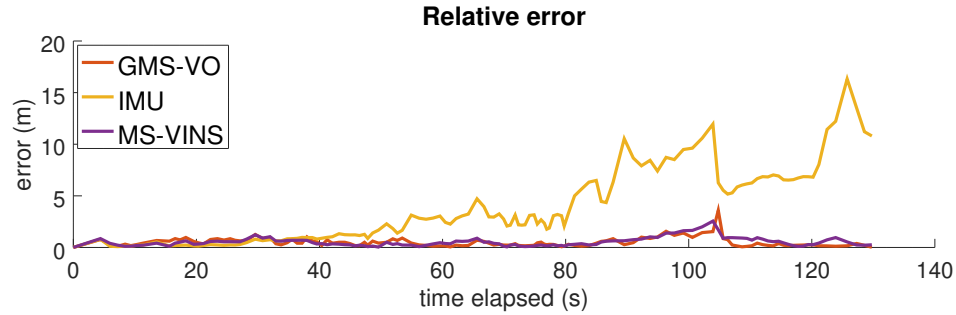
Figure 6.4: Fused trajectories obtained from the visual-inertial error-state filter corrected with relative measurements.



(a) Sequence 1.



(b) Sequence 2.



(c) Sequence 3.

Figure 6.5: Relative multispectral VINS relative errors.

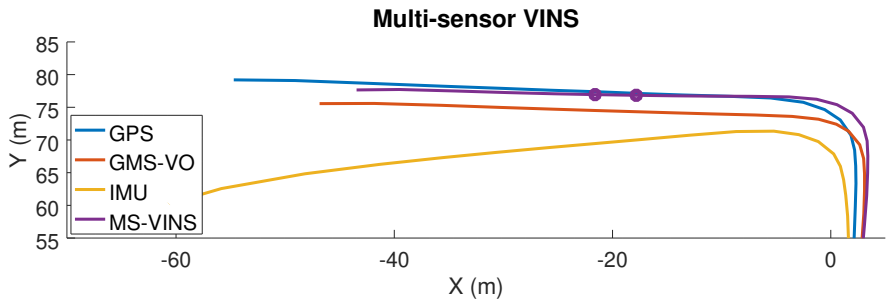
6.5.2 Covariance estimation

In order to assess the consistency of the filter, the state covariance is analysed at the different steps of the Kalman filter. First, an example extracted from sequence 1 is given. It represents the filter update between VO keyframes 203 and 210, as shown in Figure 6.6a. The different position covariance obtained in x and y axes are displayed in Figure 6.6. As expected, it can be seen that the ellipse grows as inertial data are integrated and uncertainties from the process model are included. Then, the covariance is clearly decreased as the VO measurement corrects the state estimate. It can also be noted the difference in shape of the 2D covariance ellipses between absolute and relative measurements and how they affect the current state covariance. Indeed, in Figure 6.6b the major axis remains in the x direction, reflecting uncertainties generated during the first part of the dataset. In Figure 6.6c however, R represents the uncertainties locally, which allows the filter to reduce the covariance in x and to increase it in y, reflecting the new direction of travel.

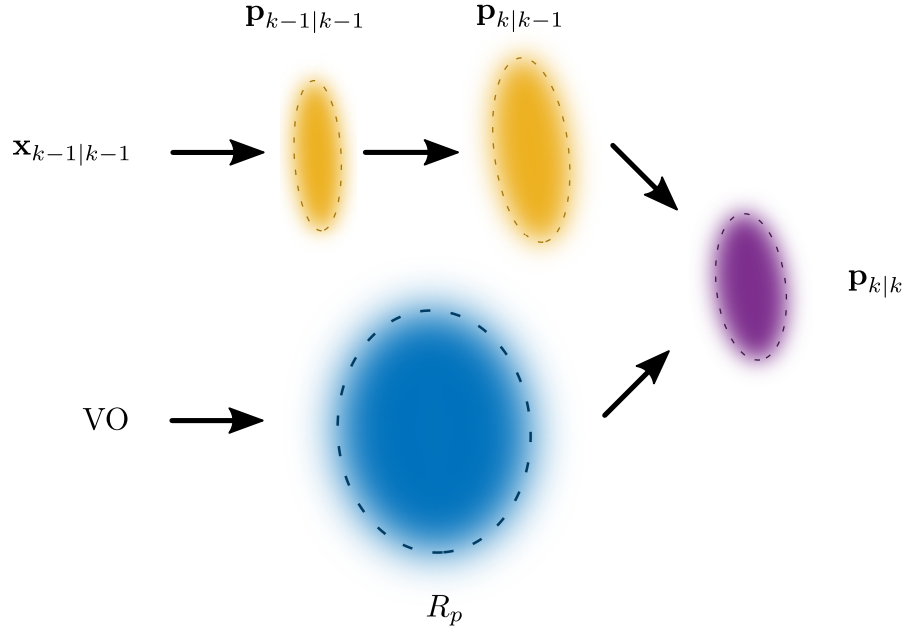
Additionally, Figure 6.7 shows the covariance in position for x and y axes, on sequence 3. For clarity, the z-axis has been omitted as the altitude is not changing and its covariance magnitude remains low. It can be seen that globally covariance grows with time but for each point, the covariance of the corrected state $P_{k|k}$ is always lower than the covariance of the propagated state $P_{k|k-1}$. This confirms that the state covariance is corrected properly for the whole duration of the sequence and confirms the results shown in the previous example.

6.5.3 Failure recovery

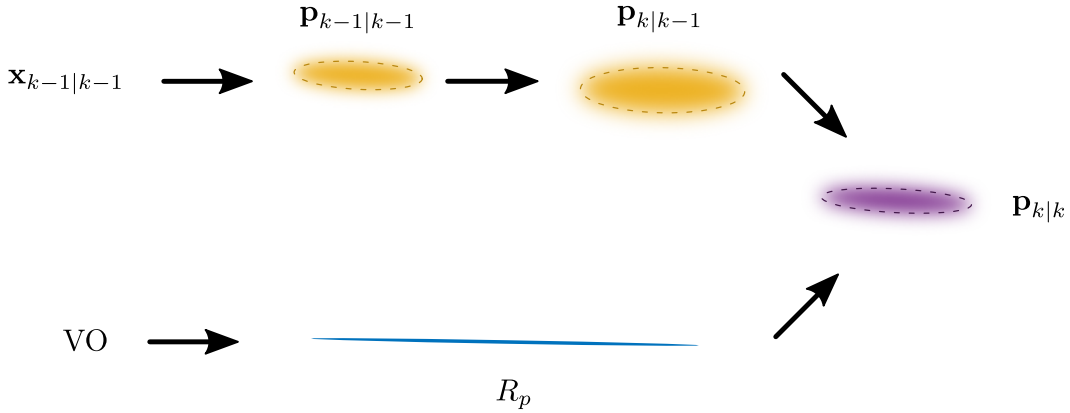
In Section 4.7.2, it has been shown that a multispectral setup recording different parts of the electromagnetic spectrum allows one camera to compensate for the



(a) Location of VO keyframes 203 and 210 on the trajectory of Sequence 1.

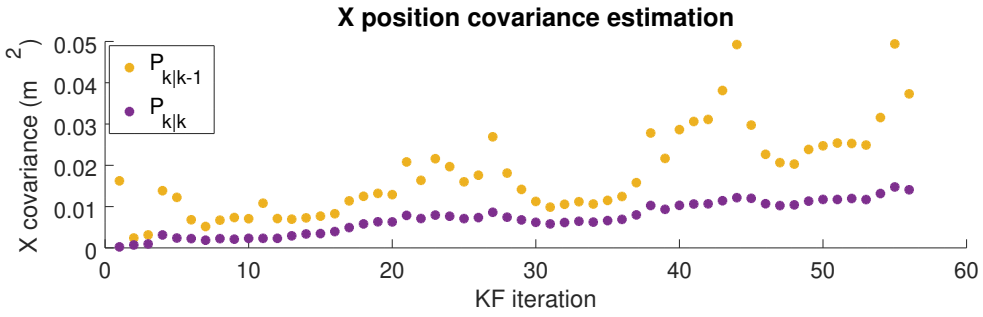


(b) State covariance corrected with absolute measurements.

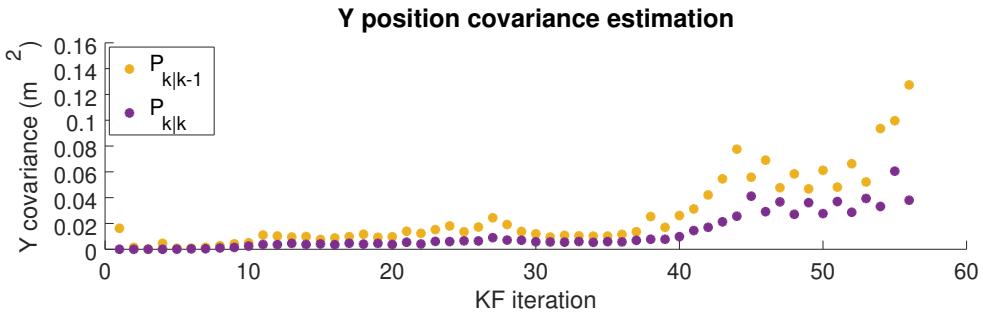


(c) State covariance corrected with relative measurements.

Figure 6.6: Evolution of the state position covariance in x and y.



(a) Covariance in X.



(b) Covariance in Y.

Figure 6.7: Position covariance before and after correction.

failures of the other. However, it is not possible to recover any motion estimate if tracking fails for both modalities at the same time. In that case, the use of an additional sensor is required. As explained earlier in this chapter, when the current state of the system is known precisely, integrating inertial data can then provide an accurate pose estimate for a short period of time. One of the drawbacks of windowed bundle adjustment is the fact that when the VO program stops due to a lack of matches, the whole window needs to be reinitialised. This step necessitates to acquire and select as many keyframes as the window size. Hence, VO measurements are not available for a few frames, which can have a dramatic effect on the VO trajectory estimation. Fortunately, during that period where VO measurements are not provided, the state can still be updated using the propagation model of the Kalman filter and inertial data. Therefore, the robustness of the fusion scheme is tested on specific scenarios where multispectral VO is not able to cope with the images recorded and needs to be reinitialised.

In this chapter, two fusion methods are presented. However, in such cases where VO fails, using absolute measurements would be pointless because errors generated by the failures would be incorporated in the absolute poses and the state would be corrected with erroneous measurements. Relative transformations on the other hand, only contains information relative to the last VO measurement. Thus, this approach is preferred to deal with failures because they can correct the current state regardless of previous failures.

As long as no measurement is received, the state is propagated into $\mathbf{x}_{k|k-1}, P_{k|k-1}$, as shown in Figure 6.8. However, the incremental method should not be used with $\mathbf{x}_{k-1|k-1}, P_{k-1|k-1}$ as it represents the last corrected state before failure (see (6.51)). Instead, the state used to generate the incremented measurement $\mathbf{z}'$ should be reinitialised when the first VO keyframe is selected, at step i. As explained in Figure 6.8, the propagated state $\mathbf{x}_{i|k-1}$ simply becomes $\mathbf{x}_{i|i}$ because no corrective

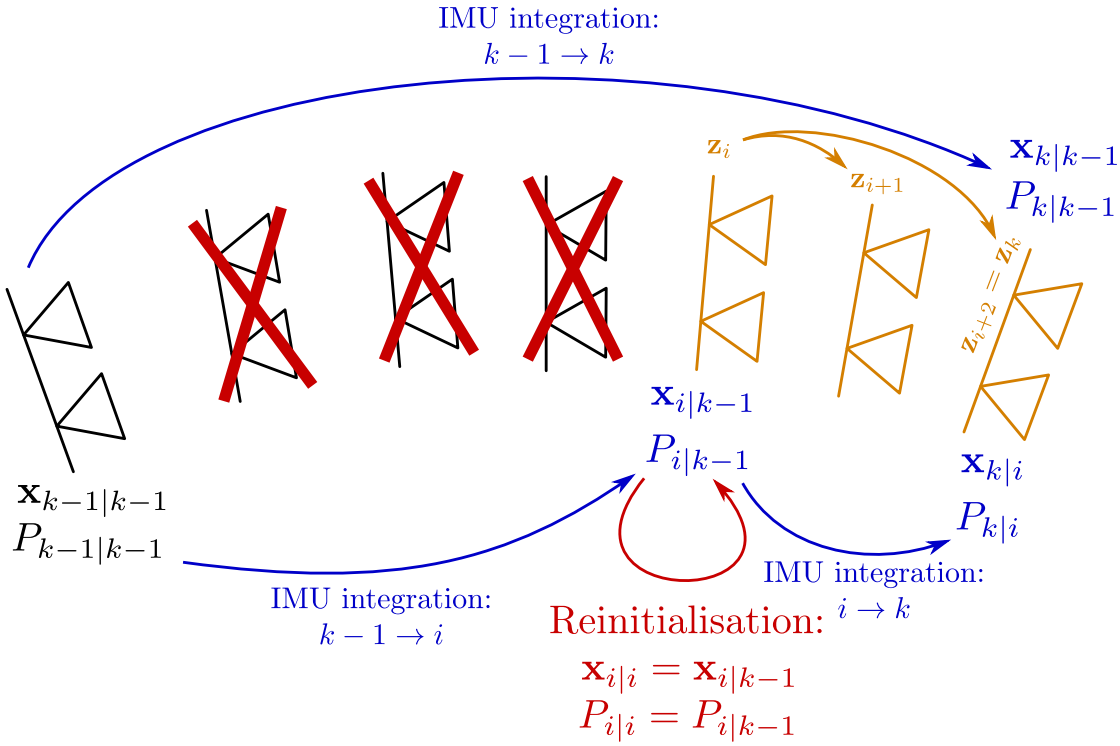


Figure 6.8: State reinitialisation with a window of 3 frames. If the former state $x_{k-1|k-1}$ is not reset with the first VO pose, future corrections will be applied from $k-1$.

measurement was provided between $k-1$ and i. $\mathbf{z}'$ will then be computed based on $\mathbf{x}_{i|i}$ and not $\mathbf{x}_{k-1|k-1}$. This means that the motion estimation between $k-1$ and i will not be corrected, which is the goal as no measurement was available to correct it.

Two datasets where feature tracking is lost simultaneously in both modalities are presented. With the first one, the car performed a fast and sharp turn and the images on both cameras have been affected by motion blur. This deteriorates the matching process of an already limited number of features, as an important part of the images represents the road. On the second dataset, VO fails when turning in front of a building where the repetitive geometry of the wall makes feature tracking more challenging and prone to errors. An example of the corresponding

(a) Visible image. (b) Thermal image.

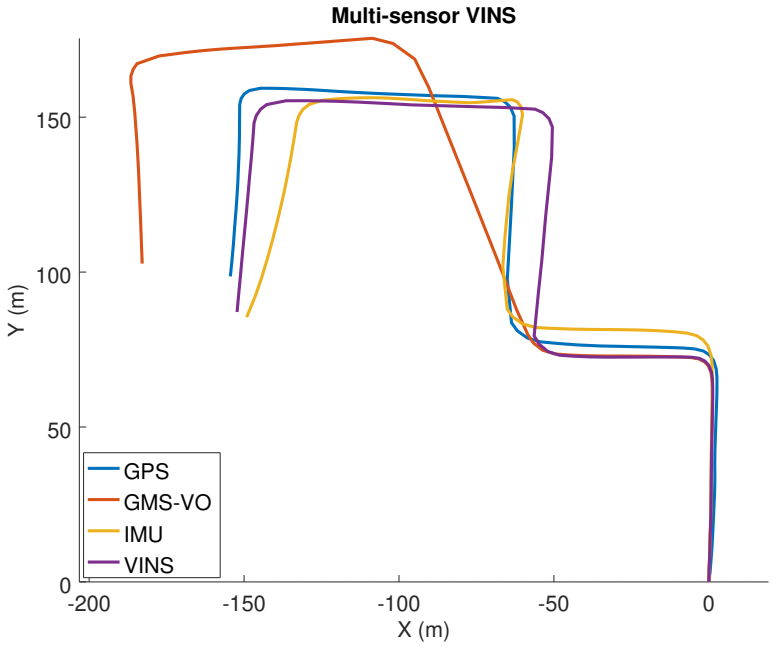
Figure 6.9: Example of a stereo pair where multispectral VO fails.

stereo images is shown in Figure 6.9.

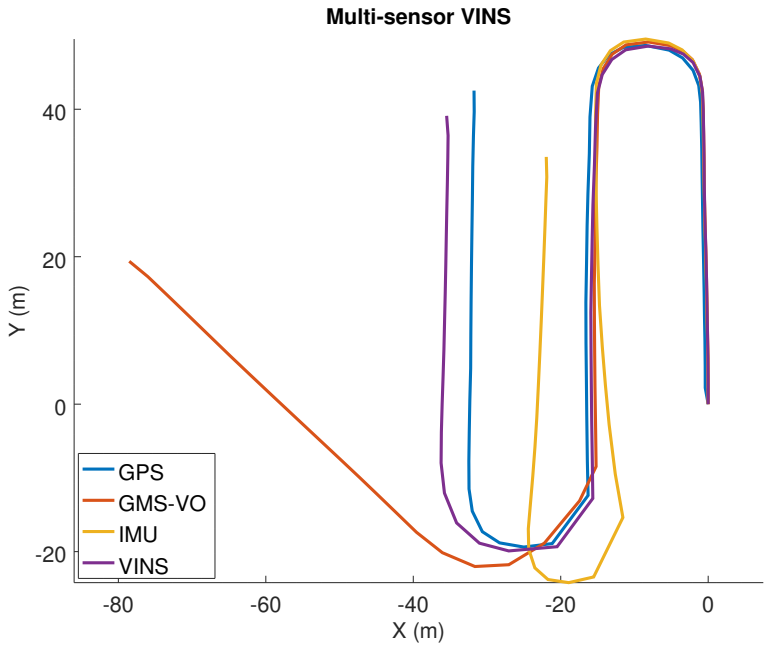
The different trajectories obtained are plotted in Figure 6.10. For both sequences, it can be noted that GMS-VO trajectories are drifting away from the ground truth compared to the other datasets where VO was not failing. This proves that absolute VO measurements cannot be used to correct the filter as the errors are propagated. However, the trajectories generated from pure inertial integration are relatively accurate. Looking at the overall trajectories, it is clear that the IMU can provide a reliable state estimate for the period where VO is not available, that is to say for a few frames.

From the relative error graphs (see Figure 6.11), a clear increase can be observed when failures happen at $t = 42\ s$ and $t = 62\ s$ on the first sequence (Figure 6.11a) and at $t = 37\ s$ on the second one (Figure 6.11b). These errors then remains high for the GMS-VO method because the orientation is miscalculated and accentuates the drift. It is nevertheless, reduced at the end of the first datasets. This is due to the second failure compensating for the first one as the car was turning in the other direction. An increase in relative errors can also be noted for the filter output on the second dataset, but it results from the challenging nature of both motion and scale estimation for that specific part of the dataset.

The results obtained are summarised in Table 6.3. It can be clearly seen that

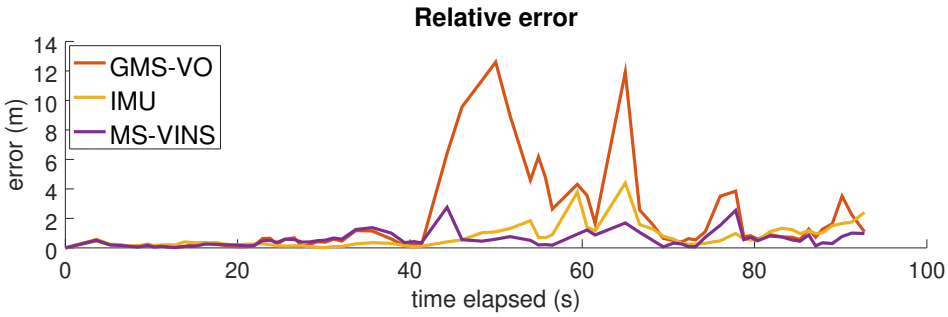


(a) Sequence 1.

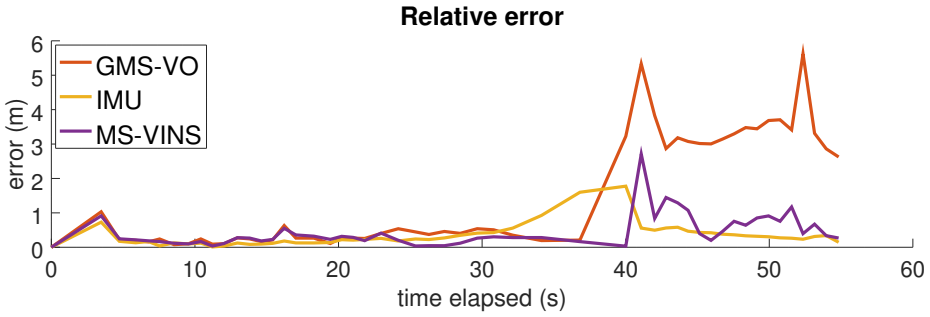


(b) Sequence 2.

Figure 6.10: Fused MS-VINS trajectories handling VO failures.



(a) Sequence 1.



(b) Sequence 2.

Figure 6.11: Relative errors of VINS trajectories for failure recovery.

Table 6.3: Errors compared to VINS filter using relative measurements.

	IMU	MS-VINS	GMS-VO
Sequence 1	357m		
mean (m)	8.713	**6.582**	17.994
std. dev. (m)	6.916	**4.408**	16.928
mean (%)	2.44	**1.84**	5.04
Sequence 2	197m		
mean (m)	4.887	**2.242**	9.864
std. dev. (m)	4.787	**1.572**	14.704
mean (%)	2.48	**1.14**	5.01

the fused solution improves the pose estimation substantially. Not only the filter is able to correct the inertial data properly to provide a better pose estimate, but it is also able to run correctly when measurements are not available and eliminates the drift resulting from the failures of the GMS-VO algorithm. These results are also coherent with the ones obtained previously as a mean error of about **5%** of the total distance was obtained with the GMS-VO technique, with is much larger than the **2%** achieved with the same technique in Chapter 5. However, the trajectories generated by the filter yield less than **2%** mean errors, which corresponds to the same error achieved on the datasets where VO does not fail. The best overall result was obtained with the filtered trajectory of the second sequence with an error as low as **1.1%**.

6.6 Conclusion

In this chapter, a new multispectral visual-inertial filter was presented. Making use of an error-state Kalman filter, it integrates inertial data in the process model to generate a predicted state and propagates the uncertainties in an error-state. The error is then corrected and the state covariance reduced when a new measurement becomes available. Measurements correspond to either absolute camera

poses or relative transformations between two poses. They are obtained from the global multispectral stereo VO algorithm (GMS-VO) presented in Chapter 5. Performance of both types of measurements were evaluated as well as the overall behaviour of the filter. Relative transformations appear to correct the state more accurately, achieving mean errors as low as **1.1%** of the distance travelled and more generally under **2%** of the distance travelled.

Finally, the superiority of multispectral stereo setups was demonstrated once again as not only the setup can rely on one camera when the other fails, but it is also able to run continuously, even when both VO processes fail to produce any pose estimate and need to be reinitialised. It was clearly shown that these kind of multimodal setups could reduce the dramatic error accumulations happening in such critical conditions.

7 | Conclusion

This thesis has investigated various multimodal visual odometry techniques. Focusing on the visible and thermal spectral bands, robust camera pose estimation was computed based on the 2D features extracted from the images. First, the calibration of such multispectral setups was addressed. Then, different motion estimation approaches were studied. A monocular VO method was developed to estimate camera poses from visible and infrared spectral bands independently and two stereo approaches were explored, where both modalities were used in a combined manner. A Kalman filter was then developed to fuse the stereo algorithm output with inertial data. Finally, the benefit of multimodal odometry was demonstrated on extreme cases where at least one camera fails to generate a pose estimate.

7.1 Discussion and improvements

The first contribution of this thesis was presented in Chapter 3 where a new calibration pattern was built to automatically calibrate multispectral stereo rigs. The pattern was created on a simple wooden board where high-power LEDs were placed. Its specific shape was designed to be easily identifiable and automatically detected. It has been shown that camera calibration can be achieved with the same accuracy as other state-of-the-art multispectral calibration techniques and calibration

patterns while being more handy. Moreover, it can be used straight away, does not require any heating system and can be moved freely in front of the cameras. Even though the calibration board and method were validated throughout the thesis, its impact on far features for stereovision was not tested due to hardware limitations. Nevertheless, it would be interesting to know the accuracy at which far features can be triangulated to better select robust points for VO.

In Chapter 4, the use of visible and thermal images for monocular visual odometry was investigated. An automatic feature tracking and keyframe selection algorithm was presented to guarantee suitable geometric properties between consecutive frames. Feature triangulation and camera pose optimisation are then performed in a sliding window containing several images. The importance of bundle adjustment and its superiority to other motion estimation techniques was demonstrated. Then, assuming at least one of the cameras was providing adequate images for VO, it was shown that having two cameras of different modalities can be beneficial for extreme illumination conditions where one camera does not produce acceptable images. Features are tracked simultaneously but VO is only carried out in the most suitable modality. More datasets could also be produced and tested to reinforce the validity of the concept.

Multispectral stereo odometry was then studied in Chapter 5. Two novel methods were proposed to solve the challenging matching problem between images coming from different parts of the electromagnetic spectrum. The first technique is based on conventional stereo odometry algorithms, which consist in performing stereo and temporal matching on consecutive stereo pairs. The relative transformation between the frames is then computed from the triangulated matches. On the other hand, the second method tracks and triangulates features in a sliding window with the same algorithm presented in Chapter 4. The scale ambiguity is then solved in a non-linear optimisation scheme by reprojecting all features in the

corresponding stereo images. While both methods yield satisfying results, it was demonstrated that the sliding window approach was more robust and generates less drift than the local matching technique. Indeed, multispectral images generate less precise stereo matches and numerous outliers, which need to be detected and discarded. This leads to inaccuracies in the triangulation process, which affects the motion estimation directly. With the global technique, motion is however recovered from images of the same modality. Camera poses are then estimated robustly and the stereo matching problem has been moved to the scale estimation process. The main drawback of these two methods however, remains their processing time. Even if the GMS-VO technique is much faster than LMS-VO, it is still time consuming compared to standard stereo visible VO. Because an important amount of features is necessary to tackle the lack of similarity between the stereo images, a more careful feature selection algorithm could be developed to try to reduce the total number of features to be be processed, especially for bundle adjustment. Furthermore, the parameters set empirically during the experiments could be included in the optimisation scheme to improve the performance of the solutions proposed.

The stereo approach was then combined with inertial data in Chapter 6. It has the advantage of providing high-rate pose estimates by integrating inertial data while preserving the accuracy of the VO process. Two types of measurements were tested: absolute camera poses, obtained independently by the VO process, and relative poses where the corrective measurement is generated incrementally based on VO relative transformations and the last state estimate. It was noted that absolute poses are not correcting the IMU drift entirely as their covariance accumulates uncertainties in all directions, whereas relative measurements are able to correct the state estimate more accurately because their uncertainties are much larger in the direction of travel for the whole duration of the dataset. As stated in

Chapter 6, the Kalman filter was employed as a mean of fusing data and neither its stability nor the randomness effect of the data were investigated. Further work could then be done in that regard.

Finally, it was proven that this multi-sensors solution is more robust than standard visual odometry techniques as it is able to cope with failures of any kind because the setup could either be used in a monocular manner, if one camera do not work properly, or it could rely on inertial data when VO fails to estimate any motion at all. Additionally, it was shown in Chapter 4 and Chapter 6 that multimodal visual odometry is able to tackle challenging illumination conditions. Even though several independent datasets were presented in this thesis, it would be interesting to investigate this further, with more extreme cases in order to show the benefit of multispectral imaging. For example, night navigation could be tested in urban environments where street lights provide a source of illumination. Even though it would lead to poor illumination conditions, the thermal camera can help to produce a robust motion estimate when visible images are too dark.

7.2 Future work

While several solutions were proposed to perform visual odometry with multispectral images, multispectral visual localisation remains a new research area and has not been investigated much. A significant amount of research can still be carried out to extend the work presented in this thesis and improve the accuracy of multispectral VO. In addition to the improvements proposed in the previous section, the following works are suggested.

First, all algorithms have been tested with the car setup presented in Section 1.4.1. It could be adapted to other kinds of vehicles such as robots or UAVs. Some datasets were collected from a drone but were not selected for this thesis due to the

quality of the environment and because images could not be stabilised. The focus was made on the processing of multimodal images rather than the development of the platform.

This thesis addresses the problem of multispectral localisation by proposing several VO algorithms. Therefore, this work could be extended with additional robust motion estimation techniques such as loop-closure. Moreover, no mention of Visible-Thermal SLAM was found in the literature. So, it could be an interesting way to go forward.

Chapter 6 presented a visual-inertial filter that uses scaled camera poses obtained from multispectral stereo VO. A new filter could be adapted to handle unscaled measurements from the monocular algorithm presented in Chapter 4. The scale would then be included in the state and estimated alongside the other parameters.

Finally, the filter uses all VO poses as soon as they become available. Instead, several measurements could be stacked together and the resulting pose used during the correction step. It would be interesting to study the effect of such measurements on the filter and how a slower correction rate would affect the fusion scheme.

References

[1] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems.* Artech House, 2007.

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

[3] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[4] S. Perera, D. Barnes, and D. Zelinsky, *Exploration: Simultaneous Localization and Mapping (SLAM).* Springer US, 2014, pp. 268–275.

[5] K. L. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, sep 2006.

[6] V. Ila, L. Polok *et al.*, "Highly Efficient Compact Pose SLAM with SLAM++," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, aug 2016.

[7] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004.

[8] A. Martinelli, "Closed-Form Solution of Visual-Inertial Structure from Motion," *International Journal of Computer Vision*, vol. 106, no. 2, pp. 138–152, jan 2014.

[9] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, jun 2004.

[10] C. Forster, L. Carlone *et al.*, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, feb 2017.

[11] S. Leutenegger, P. Furgale *et al.*, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, jun 2013, p. 19.

[12] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, aug 2018.

[13] B. Triggs, P. F. McLauchlan *et al.*, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, 2000, vol. 1883, pp. 298–372.

[14] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, apr 2007, pp. 3565–3572.

[15] A. Spaenlehauer, V. Fremont *et al.*, "A loosely-coupled approach for metric scale estimation in monocular vision-inertial systems," in *IEEE International*

Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Daegu, Korea, nov 2017, pp. 137–143.

[16] M. Li and A. I. Mourikis, "Improving the accuracy of EKF-based visual-inertial odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, USA, may 2012, pp. 828–835.

[17] G. Lu and B. Fei, "Medical hyperspectral imaging: a review," *Journal of Biomedical Optics*, vol. 19, no. 1, p. 010901, jan 2014.

[18] A. Goldberg, B. Stann, and N. Gupta, "Multispectral, hyperspectral, and three-dimensional imaging research at the U.S. Army research laboratory," in *International Conference of Information Fusion (FUSION 2003)*, Cairns, Australia, 2003, pp. 499–506.

[19] T. Mouats, N. Aouf *et al.*, "Multi-Spectral Stereo Odometry," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1–15, 2015.

[20] T. Mouats, N. Aouf *et al.*, "Thermal Stereo Odometry for UAVs," *IEEE Sensors Journal*, vol. 15, no. 11, pp. 6335–6347, 2015.

[21] P. Borges and S. Vidas, "Practical Infrared Visual Odometry," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 1–9, 2016.

[22] J. Poujol, C. A. Aguilera *et al.*, "A Visible-Thermal Fusion Based Monocular Visual Odometry," in *Advances in Intelligent Systems and Computing*, 2016, vol. 417, pp. 517–528.

[23] A. Beauvisage, N. Aouf, and H. Courtois, "Multi-spectral visual odometry for unmanned air vehicles," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, oct 2016, pp. 1994–1999.

[24] A. Beauvisage and N. Aouf, "Multimodal visual-inertial odometry for navigation in cold and low contrast environment," in *European Conference on Mobile Robots (ECMR)*, Paris, France, sep 2017, pp. 1–6.

[25] T. Mouats and N. Aouf, "Multimodal stereo correspondence based on phase congruency and edge histogram descriptor," in *IEEE International Conference on Information Fusion (FUSION)*, Istanbul, Turkey, 2013, pp. 1981–1987.

[26] S. Hwang, J. Park *et al.*, "Multispectral Pedestrian Detection : Benchmark Dataset and Baseline," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, 2015, pp. 1037–1045.

[27] C. A. Aguilera, F. J. Aguilera *et al.*, "Learning Cross-Spectral Similarity Measures with Deep Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Las Vegas, USA, jun 2016, pp. 267–275.

[28] T. Mouats, N. Aouf *et al.*, "Performance Evaluation of Feature Detectors and Descriptors Beyond the Visible," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 1, pp. 33–63, sep 2018.

[29] D. Rondao, N. Aouf, and O. Dubois-Matra, "Multispectral Image Processing for Navigation Using Low Performance Computing," in *International Astronautical Congress (IAC)*, Bremen, Germanye, 2018.

[30] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[31] G. Guennebaud and B. Jacob, "Eigen V3," 2010. [Online]. Available: http://eigen.tuxfamily.org

[32] S. Agarwal, K. Mierle, and E. al., "Ceres Solver." [Online]. Available: http://ceres-solver.org

[33] J. Adams, "Lectures on exceptional lie groups," *Computers & Mathematics with Applications*, vol. 33, no. 7, p. 136, apr 1997.

[34] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," Tech. Rep., 2006.

[35] J. Solà, "Quaternion kinematics for the error-state Kalman filter," Tech. Rep., 2017.

[36] M. Quigley, K. Conley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[37] J. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," Tech. Rep., 2014.

[38] S. Denman, T. Lamb *et al.*, "Multi-spectral fusion for surveillance systems," *Computers & Electrical Engineering*, vol. 36, no. 4, pp. 643–663, jul 2010.

[39] E. Nilsson, "An Optimization Based Approach to Visual Odometry Using Infrared Images," Ph.D. dissertation, Linkopings University, 2010.

[40] A. Ellmauthaler, E. A. B. da Silva *et al.*, "A novel iterative calibration approach for thermal infrared cameras," in *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, sep 2013, pp. 2182–2186.

[41] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[42] S. Prakash, P. Y. Lee *et al.*, "Robust thermal camera calibration and 3D mapping of object surface temperatures," in *SPIE Defense and Security Symposium*, Orlando, USA, apr 2006, p. 62050.

[43] S. Vidas, R. Lakemond *et al.*, "A Mask-Based Approach for the Geometric Calibration of Thermal-Infrared Cameras," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 6, pp. 1625–1635, jun 2012.

[44] N. Kim, Y. Choi *et al.*, "Geometrical calibration of multispectral calibration," in *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Goyang, South Korea, oct 2015, pp. 384–385.

[45] M. Gschwandtner, R. Kwitt *et al.*, "Infrared camera calibration for dense depth map construction," in *IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, German, jun 2011, pp. 857–862.

[46] A. Beauvisage and N. Aouf, "Low cost and low power multispectral thermal-visible calibration," in *IEEE SENSORS*, Glasgow, UK, oct 2017, pp. 1–3.

[47] D. Claus and A. W. Fitzgibbon, "A rational function lens distortion model for general cameras," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, USA, 2005, pp. 213–219.

[48] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[49] M. I. A. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software*, vol. 36, no. 1, pp. 1–30, 2009.

[50] F. Fraundorfer and D. Scaramuzza, "Visual Odometry : Part II," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2011.

[51] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[52] R. Wang, M. Schworer, and D. Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," in *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 3923–3931.

[53] R. Gomez-ojeda and J. Gonzalez-jimenez, "Robust Stereo Visual Odometry through a Probabilistic Combination of Points and Line Segments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016, pp. 2521–2526.

[54] J. J. Tarrio and S. Pedre, "Realtime Edge Based Visual Inertial Odometry for MAV Teleoperation in Indoor Environments," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 1-2, pp. 235–252, may 2018.

[55] D. Rondao and N. Aouf, "Multi-View Monocular Pose Estimation for Spacecraft Relative Navigation," in *AIAA Guidance, Navigation, and Control Conference*, Reston, Virginia, jan 2018.

[56] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference (AVC)*, Manchester, UK, 1988, pp. 23.1–23.6.

[57] Jianbo Shi and Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, USA, dec 1994, pp. 593–600.

[58] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 430–443.

[59] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, nov 2004.

[60] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 404–417.

[61] M. Calonder, V. Lepetit *et al.*, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision (ECCV)*, Crete, Greece, 2010, pp. 778–792.

[62] E. Rublee, V. Rabaud *et al.*, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011, pp. 2564–2571.

[63] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011, pp. 2548–2555.

[64] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, jun 2012, pp. 510–517.

[65] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *International Joint Conference on Artificial Intelligence (IJCAI)*, Vancouver, Canada, 1981, pp. 674–679.

[66] J. Y. Bouguet, "Pyramidal implementation of the lucas-kanade feature tracker," Tech. Rep., 2000.

[67] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, no. 1, pp. 43–75, jan 1996.

[68] R. I. Hartley, "In Defense of the Eight-Point Algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[69] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, sep 1981.

[70] H. Lim and Y. Sam Lee, "Indoor Single Camera SLAM using Fiducial Markers," *Journal of Institute of Control, Robotics and Systems*, vol. 15, no. 4, pp. 353–364, apr 2009.

[71] J. Zhang, S. Singh, and G. Kantor, "Robust Monocular Visual Odometry for a Ground Vehicle in Undulating Terrain," in *Springer Tracts in Advanced Robotics*, 2014, vol. 92, pp. 311–326.

[72] Z. Dingfu, Y. Dai, and L. Hongdong, "Reliable scale estimation and correction for monocular Visual Odometry," in *IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, jun 2016, pp. 490–495.

[73] S. Weiss, M. W. Achtelik *et al.*, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, sep 2013.

[74] D. Caruso, A. Eudes *et al.*, "A Robust Indoor/Outdoor Navigation Filter

Fusing Data from Vision and Magneto-Inertial Measurement Unit," *Sensors*, vol. 17, no. 12, p. 2795, dec 2017.

[75] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, may 2011, pp. 4531–4537.

[76] S. Lynen, M. W. Achtelik *et al.*, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan: IEEE, nov 2013, pp. 3923–3929.

[77] I. Esteban, L. Dorst, and J. Dijk, "Closed Form Solution for the Scale Ambiguity Problem in Monocular Visual Odometry," in *International Conference on Intelligent Robotics and Applications (ICIRA)*, Shanghai, China, 2010, pp. 665–679.

[78] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, USA, 2011, pp. 2969–2976.

[79] X.-S. Gao, X.-R. Hou *et al.*, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[80] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[81] L. Lemay, "Algorithm for Triangulating Visual Landmarks and Determining Their Error Covariance," Tech. Rep., 2012.

[82] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, Kenting, Taiwan, nov 2007, pp. 1–10.

[83] T. Liu and S. Shen, "High altitude monocular visual-inertial state estimation: Initialization and sensor fusion," in *IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, may 2017, pp. 4544–4551.

[84] C. V. Stewart, "Robust Parameter Estimation in Computer Vision," *SIAM Review*, vol. 41, no. 3, pp. 513–537, jan 1999.

[85] Z. Zhang, "Parameter estimation techniques: a tutorial with application to conic fitting," *Image and Vision Computing*, vol. 15, no. 1, pp. 59–76, jan 1997.

[86] A. Eudes and M. Lhuillier, "Error propagations for local bundle adjustment," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, USA, jun 2009, pp. 2411–2418.

[87] A. Eudes, S. Naudet-Collette *et al.*, "Weighted Local Bundle Adjustment and Application to Odometry and Visual SLAM Fusion," in *British Machine Vision Conference (BMVC)*, Aberystwyth, UK, 2010, pp. 25.1–25.10.

[88] T. Mouats, N. Aouf, and M. A. Richardson, "A Novel Image Representation via Local Frequency Analysis for Illumination Invariant Stereo Matching," *IEEE Transactions on Image Processing*, vol. 24, no. 9, pp. 2685–2700, 2015.

[89] B. Tippetts, D. Jye *et al.*, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 5–25, 2016.

[90] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, jun 2012, pp. 3354–3361.

[91] C. Aguilera, F. Barrera *et al.*, "Multispectral Image Feature Points," *IEEE Sensors*, vol. 12, no. 12, pp. 12 661–72, 2012.

[92] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *IEEE Intelligent Vehicles Symposium (IV)*, San Diego, USA, jun 2010, pp. 486–492.

[93] H. Badino and T. Kanade, "A Head-Wearable Short-Baseline Stereo System for the Simultaneous Estimation of Structure and Motion," in *IAPR Conference on Machine Vision Applications (MVA)*, Nara, Japan, 2011, pp. 185–189.

[94] J. Witt and U. Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, nov 2013, pp. 4164–4171.

[95] D. Scharstein, H. Hirschmüller *et al.*, "High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth," *German Conference on Pattern Recognition (GCPR)*, pp. 31–42, 2014.

[96] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, jun 2015, pp. 3061–3070.

[97] J. Uhrig, N. Schneider *et al.*, "Sparsity Invariant CNNs," in *International Conference on 3D Vision (3DV)*, Qingdao, China, oct 2017, pp. 11–20.

[98] I. Cvišić, J. Ćesić *et al.*, "SOFT-SLAM : Computationally Efficient Stereo Visual SLAM for Autonomous UAVs," *Journal of Field Robotics*, vol. 35, no. 4, pp. 578–595, 2017.

[99] O. Yilmaz, N. Aouf *et al.*, "Using infrared based relative navigation for active debris removal," in *International ESA Conference on Guidance, Navigation & Control Systems (GNC)*, Salzburg, Austria, 2017.

[100] G. Egnal, "Mutual Information as a Stereo Correspondence Measure," Tech. Rep., 2000.

[101] M. Yaman and S. Kalkan, "An iterative adaptive multi-modal stereo-vision method using mutual information," *Journal of Visual Communication and Image Representation*, vol. 26, no. C, pp. 115–131, jan 2015.

[102] S. J. Krotosky and M. M. Trivedi, "Mutual information based registration of multimodal stereo videos for person tracking," *Computer Vision and Image Understanding*, vol. 106, no. 2-3, pp. 270–287, 2007.

[103] A. Torabi and G. A. Bilodeau, "Local self-similarity-based registration of human ROIs in pairs of stereo thermal-visible videos," *Pattern Recognition*, vol. 46, no. 2, pp. 578–589, 2013.

[104] F. B. Campo, F. L. Ruiz, and A. D. Sappa, "Multimodal Stereo Vision System : 3D Data Extraction and Algorithm Evaluation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 5, pp. 437–446, 2012.

[105] P. Viola and W. M. Wells, "Alignment by Maximization of Mutual Information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[106] P. Kovesi, "Phase congruency : A low-level image invariant," *Psychological Research*, vol. 64, no. 2, pp. 136–148, 2000.

[107] M. C. Morrone and R. A. Owens, "Feature detection from local energy," *Pattern Recognition Letters*, vol. 6, no. 5, pp. 303–313, 1987.

[108] P. Kovesi, "Phase Congruency Detects Corners and Edges," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Sidney, Australia, 2003, pp. 309–18.

[109] C. P. Bridge, "Introduction To The Monogenic Signal," Tech. Rep., mar 2017.

[110] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems (RSS)*, Berkeley, USA, jul 2014, pp. 109–111.

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

[112] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, may 2013, pp. 4698–4705.

[113] J. A. Hesch, D. G. Kottas *et al.*, "Camera-IMU-based localization: Observability analysis and consistency improvement," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 182–201, jan 2014.

[114] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman Filtering for INS/GPS," *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, may 1999.

[115] C. Hide, T. Moore, and M. Smith, "Adaptive Kalman Filtering for Low-cost INS/GPS," *Journal of Navigation*, vol. 56, no. 1, pp. 143–152, jan 2003.

[116] A. Almagbile, J. Wang, and W. Ding, "Evaluating the Performances of Adaptive Kalman Filter Methods in GPS/INS Integration," *Journal of Global Positioning Systems*, vol. 9, no. 1, pp. 33–40, 2011.

[117] N. de Palezieux, T. Nageli, and O. Hilliges, "Duo-VIO: Fast, light-weight, stereo inertial odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, oct 2016, pp. 2237–2242.

[118] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization," in *IEEE International Conference on Robotics and Automation (ICRA)*, Prague, Czech Republic, may 2018, pp. 2510–2517.

[119] L. Kneip, M. Chli, and R. Siegwart, "Robust Real-Time Visual Odometry with a Single Camera and an IMU," in *British Machine Vision Conference (BMVC)*, Dundee, Scotland, 2011, pp. 16.1–16.11.

[120] H. He, Y. Li, and J. Tan, "Relative motion estimation using visual–inertial optical flow," *Autonomous Robots*, vol. 42, no. 3, pp. 615–629, mar 2018.

[121] T. Qin and S. Shen, "Robust initialization of monocular visual-inertial estimation on aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, sep 2017, pp. 4225–4232.

[122] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[123] G. Welch and G. Bishop, "An introduction to the Kalman Filter," Tech. Rep., feb 2006.

[124] D. Simon, *Optimal State Estimation.* John Wiley & Sons, Inc., 2006.

[125] N. Trawny and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," University of Minnesota, Tech. Rep., 2005.

[126] Xsens, "MTi-G-710, Turnkey GNSS/INS solution for navigation and stabilization applications," 2019. [Online]. Available: https://www.xsens.com/download/pdf/documentation/mti-g-100/mti-g-710-series.pdf

[127] J. Nocedal and S. J. Wright, *Numerical optimization.* Springer, 1999.

[128] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, jun 1963.

[129] 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*, vol. 24, no. 6, pp. 381–395, jun 1981.

Appendices

A | Non-Linear Least-Squares Optimisation

Optimisation is the process of estimating a set of unknown parameters $\mathbf{x}$ that minimises or maximises a certain objective function, denoted by f. Optimisation can either be *constrained*, when the elements of $\mathbf{x}$ must satisfy some conditions, or *unconstrained* otherwise. A general optimisation problem is defined as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{subject to } \begin{cases} c_i(\mathbf{x}) = 0 \\ c_j(\mathbf{x}) \leq 0 \end{cases} \tag{A.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of n components and $f : \mathbb{R}^n \to \mathbb{R}$. $c(\mathbf{x})$ encapsulates all the constraints the problem is subject to. Because all optimisations performed in this thesis are unconstrained, these constraints are ignored from now on.

Local minimum

A global minimiser is also a local minimiser in its vicinity. However, unless the objective function is convex, there is no simple way to find the global minimiser of an optimisation problem. Nevertheless, it is important to recognise a local minimum and its characteristics. A local minimum is a point for which the objective function returns the smallest value in its neighbourhood. To evaluate mathematically f at a certain point $\mathbf{x}$, the Taylor's theorem is employed. Suppose that f is continuously differentiable, for a small variation $\mathbf{p}$, $f(\mathbf{x} + \mathbf{p})$ can be extended as

follows:

$$f(\mathbf{x}+\mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x}+\mathbf{p})^T\mathbf{p} \tag{A.2}$$

if f is twice continuously differentiable, then:

$$\nabla f(\mathbf{x}+\mathbf{p}) = \nabla f(\mathbf{x}) + \int_0^1 \nabla^2 f(\mathbf{x}+\mathbf{p})\mathbf{p}\ dt \tag{A.3}$$

which can then be substituted to (A.2)

$$f(\mathbf{x}+\mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T\nabla^2 f(\mathbf{x}+\mathbf{p})\mathbf{p} \tag{A.4}$$

From this formulation sufficient and necessary conditions can be derived, assuming that $\mathbf{x}^*$ is a local minimiser, as proved in [127]:

First-order necessary condition. *If x^* is a local minimiser and f is continuously differentiable in an open neighbourhood of x^*, then $\nabla f(x) = 0$.*

Second-order necessary condition. *If x^* is a local minimiser and $\nabla^2 f$ is continuously differentiable in an open neighbourhood of x^*, then $\nabla f(x) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.*

First-order sufficient condition. *If $\nabla^2 f$ is continuously differentiable in an open neighbourhood of x^* and $\nabla f(x) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a local minimiser of f.*

Minimisation problems can be solved numerically with iterative methods. Starting from an initial point $\mathbf{x}_0$ a new point $\mathbf{x}_k$ is computed at each iteration based on f and its derivatives. $\mathbf{x}_k$ is selected such as the objective function is reduced ($f_k < f_{k-1}$). The operation is repeated until the minimiser is approximated with sufficient accuracy or when the algorithm cannot progress any more. To increase its speed and convergence, the optimisation process can be initialised with relevant values if the nature of the problem allows to make a credible guess.

Two main strategies can be used to determine the step between two iterations: *line-search* and *trust-region* approaches. Line-search methods consist in finding an optimal direction which reduces the objective function, and then the magnitude of the step. It is expressed as:

$$\min_{\alpha>0,\mathbf{p}_k} \; f(\mathbf{x}_k + \alpha \mathbf{p}_k) \tag{A.5}$$

Trust-region methods on the other hand create a model m_k which approximates the objective function around $\mathbf{x}_k$. The optimal step is then computed by solving:

$$\min_{\mathbf{p}_k} \; m_k(\mathbf{x}_k + \mathbf{p}_k) \tag{A.6}$$

Least-squares problems

Least-squares problems have a special objective function that can be expressed as:

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x) \tag{A.7}$$

where $r_j(x)$ is the residual or error for each observation j.

This objective function can simply be differentiated as follows:

$$\nabla f(x) = \sum_{=1}^{m} r_j(x) \nabla r_j(x) \tag{A.8}$$

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^{m} \nabla r_j(x) \nabla^2 r_j(x) \tag{A.9}$$

and expressed in matrix form:

$$\nabla f(x) = J_{|\mathbf{x}}\mathbf{r} \tag{A.10}$$

$$\nabla^2 f(x) = J_{|\mathbf{x}}^T J_{|\mathbf{x}} + \sum_{i=1}^{m} \nabla r_j(x) \nabla^2 r_j(x) \tag{A.11}$$

where $\mathbf{r}$ is the vector of all residuals computed at $\mathbf{x}$ and $J_{|\mathbf{x}}$ its Jacobian:

$$\mathbf{r} = \begin{bmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_m(x) \end{bmatrix} \quad J_{|\mathbf{x}} = \left[\frac{\partial r_j(x)}{\partial x_i} \right]_{i \in \{1,n\}, j \in \{1,m\}} \tag{A.12}$$

These equations are useful because it means that the Hessian $\nabla^2 f(x)$ can be approximated from the Jacobian: $H(x) \simeq J(x)J(x)$. Indeed, the second term of $\nabla^2 f(x)$ is usually negligible either because the residual $r(x)$ is small, or close to linearity near the solution ($\nabla^2 r(x)$ is small).

Line-search methods

As mentioned earlier, line-search methods aims at finding a direction vector p and its magnitude α so that the objective function is reduced at the next iteration. The most obvious choice is the direction of *steepest-descent*:

$$p_k = -\alpha \frac{\nabla f(x_k)}{\|f(x_k)\|} \tag{A.13}$$

which can be adapted to the non-linear least-squares problem as:

$$\mathbf{p}_k = -\alpha \frac{J_k \mathbf{r}_k}{\|J_k \mathbf{r}_k\|} \tag{A.14}$$

The steepest descent is the simplest line-search method and does not require the computation of second derivative, however it is usually slow to converge.

The *Gauss-Newton* method on the other hand, makes use of the special struc-

ture of the least-squares gradient ∇f and Hessian $\nabla^2 f$ functions to improve the convergence rate. Gauss-Newton is an adaptation of the Newton method where the step is obtained by solving:

$$\nabla^2 f(x_k)p = -\alpha \nabla f(x_k) \tag{A.15}$$

For the Gauss-Newton method, this equation then becomes:

$$J_k^T J_k \mathbf{p}_k = -\alpha J_k r_k \tag{A.16}$$

this equation is usually referred as *normal equation*.

Gauss-Newton has the advantage of converging much faster than the steepest-descent method on many least-squares optimisation problems [127].

Trust-region methods

Levenberg-Maquardt is the trust-region alternative to Gauss-Newton and steepest descent techniques. In fact, it can be seen as a trade-off between the two line-search methods. Indeed, for a spherical trust-region with radius Δ, if the solution of the Gauss-Newton normal equation lies in the trust region ($\|\mathbf{p}_{GN}\| < \Delta$), then it behaves as such. Otherwise, a damping factor $\lambda > 0$ is added. The Levenberg-Maquardt normal equation can then be expressed as:

$$(J_k^T J_k + \lambda I)\mathbf{p}_k = -J_k r_k \tag{A.17}$$

Levenberg-Marquardt combines the advantage of both line-search methods as it computes a direction close to the steepest-descent when the current solution is far from a local minimum (λ is large) and acts as the Gauss-Newton method when close to a local minimum ($\lambda = 0$) [128]. Therefore, it is guaranteed to converge when behaving like the steepest descent, but it also converges faster when approaching a local minima, as it starts behaving like Gauss-Newton. As a result,

the Levenberg-Maquardt algorithm is usually more robust to bad initialisations [7].

B | RANSAC framework

RANSAC is a framework used to detect outliers in a set of observations. For VO purposes, it means detecting 2D/3D features which have been wrongly matched or triangulated. The algorithm which stands for *RANdom SAmple Consensus*, was originally developed by Fischler and Bolles [129] and consists in classifying the data into two categories (inliers and outliers) from minimal subsets. It is useful for parameter estimation techniques such as optimisation where a model is fit to noisy data. It has become popular due to its speed and robustness to large number of outliers. Indeed, using a small subset makes the optimisation/regression fast to compute and robust estimation can still be performed even when 50% of outliers are present in the data [7], though the more outliers, the more iterations are required.

The RANSAC framework is detailed in Algorithm 2. First, several subset of observations containing just enough points to estimate the model are drawn randomly from the initial data. For example, a minimum set of three 3D points (and their corresponding 2D matches in the images) is required to solve the 6 DoF pose of a camera. A first estimation/optimisation is performed, and the errors between observations and points predicted by the model are then computed for all data points. Each one of them is then classified as inlier or outlier depending on its error. The key idea is that if the set being tested contains weak matches, the model calculated will not fit the data and will not be supported by the other

points. If the operation is repeated sufficiently, at least one subset of good matches will be tested. As a result, the model that best fits the data and produces the largest number of inliers is selected for outlier rejection. The data points that are classified as outliers by this model are then discarded to keep only the most robust points. Finally, a last estimation is performed with the remaining points in order to refine the solution.

Algorithm 2 RANSAC framework

Input:
Ω: set of observations
τ: inlier threshold
N: total number of iterations
i: size of a subset
Output:
$\hat{X}$: parameters estimated

```
for i ← 1 to N do
  S ← extractSubsetFrom(Ω,i)
  X ← optimise(S)
  nb_inliers ← 0
  for all s ∈ Ω do
    ε ← computeResidualError(s,X)
    if ε < τ then
      nb_inliers ← nb_inliers + 1
    end if
  end for
  if nb_inliers > max_nb_inliers then
    S_best ← S
    max_nb_inliers ← nb_inliers
  end if
  Ŝ ← removeOutliers(S_best)
  X̂ ← optimise(Ŝ)
end for
```