Tecnun - University of Navarra
Faculty of Engineering

Monocular Visual Perception Techniques for Augmented Reality and Mobile Robotics Applications in Industry

by

Jon Zubizarreta Gorostidi

October, 2019

Supervised by
Iker Aguinaga Hoyos

Dissertation submitted for the Degree of Doctor of Philosophy of the University of Navarra.

This thesis has been supported by the Basque Government under the predoctoral program.

Tecnun - University of Navarra
Donostia - San Sebastián
Spain

A mis padres

Agradecimientos

Estos años de tesis me han proporcionado muchos conocimientos técnicos, pero aún más importante, me ha hecho comprender el valor del trabajo bien hecho, la constancia, la paciencia, el descanso e incluso el fracaso. Además no hubiera llegado hasta aquí sin el apoyo de toda esa gente que ha estado a mi alrededor y que tanto me ha ayudado. A todos ellos millones de gracias, siempre os lo deberé.

Me gustaría también expresar mis agradecimientos al Gobierno Vasco por la concesión de una de sus becas, más concretamente la financiación obtenida a través del *Programa de Formación Personal Investigador No Doctor*.

Muchas gracias a Alejo Avello por todos sus consejos durante mi etapa universitaria y animarme a realizar la tesis doctoral. A su vez a CEIT y a TECNUN (Universidad de Navarra) por confiar en mí y darme la oportunidad de realizar los estudios de doctorado dentro del grupo de robótica y visión.

Y cómo no, mi director Iker Aguinaga. Agradecido por todo el tiempo y esfuerzo que has invertido en mí. Siempre dispuesto a discutir los problemas y dar tu sincera opinión. Por aportarme calma y tranquilidad en los momentos más dificiles, que los ha habido muchos. Simplemente, ¡gracias!

También querría agradecer a José María Martinez Montiel por acogerme en su grupo de investigación de la Universidad de Zaragoza durante unos meses. Gracias por enseñarme tanto y estar siempre disponible para aconsejarme sobre cualquier cuestión, fuera o no realacionada con el trabajo. La motivación que me has proporcionado ha sido clave en esta última etapa. También a toda la gente del laboratorio que tan bien me acogieron e hicieron sentirme parte del equipo. He conocido gente increible allí con la que seguro seguiré compartiendo grandes momentos.

Y a mis compañeros de trabajo, que han hecho el día a día más llevadero. Por esas conversaciones en el "coffee time", las risas y las cenas en el txoko. Estoy seguro que la tesis hubiera sido mucho menos divertida sin vosotros. Ha sido un placer compartir este viaje con vosotros, ¡gracias!

Muchas gracias a mi familia y amigos. En especial a mis padres por apoyarme siempre. Por todo lo que habeis hecho por mi, no solo durante la tesis sino a lo largo de toda mi vida. Si he llegado hasta aquí es, sin duda alguna, por vosotros. Eskerrik asko bihotz-bihotzez!

Abstract

The current advances in communication and computing technologies are having a large impact in industry, leading to what's known as the fourth industrial revolution or Industry 4.0. One of the challenges being addressed is to augment machines with the intelligence to mimic the cognitive functions of the human mind. In this context, machine perception is one of the core capacities to interpret data related to the world around us. For this purpose, computer vision (CV) is a commonly used solutions due its versatility and low cost implementation of the optical sensors.

This thesis studies two different visual perception problems: *object recognition* and *simultaneous localization and mapping* (SLAM). The proposed solutions focus on single camera (monocular) approaches in industrial environments. This is specially challenging due to the lack of textured surfaces of objects typical in industry, uncontrolled illumination changes, non-Lambertian materials – that render many reflections – and cluttered scenes. Both problems consist in understanding the scene and determining the camera motion as accurately as possible. Object recognition sets its focus on identifying target 3D objects in the scene, whereas SLAM aims to recover the 3D structure of the scene.

The first part of this thesis proposes a novel model-based object recognition method which uses geometric properties. It combines model surface conics and edge templates to reduce the image search space increasing the localization robustness and saving computational time. In addition, the proposed method is integrated into a complete augmented reality (AR) framework for guidance in maintenance in industry, called ARgitu. It generates and presents virtual and augmented information, including the tools required for the development of new contents and adapt AR technology applications into the advanced manufacturing industry.

The second part of this thesis presents a direct monocular SLAM system, called Direct Sparse Mapping (DSM). It uses a direct formulation within a mapping framework to locate the position of the camera in the scene and build a consistent global map. Up to our knowledge, this is the first fully direct SLAM approach to reuse map point reobservations. As a direct method, it does not rely on point matches and it can work with points sampled across image edges – instead of only corners – and obtain a more descriptive reconstruction despite the sparse geometry representation. The system is robust inmscenes with low texture and motion blur. The extensive experimental validation demonstrates that the proposed direct mapping framework outperforms current direct odometry approaches – even with loop closure – both in the estimated trajectory and map accuracy.

Resumen

Los avances actuales en las tecnologías de comunicación y computación están teniendo un gran impacto en la industria, conduciendo a la que se conoce como la cuarta revolución industrial o Industria 4.0. Uno de los principales retos es proporcionar a las máquinas la inteligencia necesaria para imitar las funciones cognitivas de la mente humana. En este contexto, la percepción e interpretación del mundo que nos rodea es una de las capacidades principales. Para este propósito, la visión por computador es una solución muy usada debido a su versatilidad y bajo coste de implementación de los sensores ópticos.

Esta tesis estudia dos técnicas de percepción visual diferentes: *reconocimiento de objetos* y *localización y mapeo simultáneos* (SLAM por sus siglas en inglés). Las soluciones propuestas se centran en una única cámara (monocular) en entornos industriales. Esto es un desafío debido a la falta de superficies con textura en la escena, cambios de iluminación no controlados, materiales no-Lambertianos – que producen muchos reflejos – y escenas abarrotadas. Ambos problemas consisten en comprender la escena y determinar el movimiento de la cámara con la mayor precisión posible. El reconocimiento de objetos se enfoca en identificar objetos objetivo en la escena, mientras que el SLAM pretende recuperar la estructura tridimensional de la escena.

La primera parte de esta tesis propone un nuevo método de reconocimiento de objetos basado en modelos que utiliza propiedades geométricas de los mismos. Combina cónicas de la superficie del modelo y plantillas de aristas para reducir el espacio de búsqueda en la imagen, aumentando la solidez de la localización y reduciendo el tiempo de cálculo. Además, el método propuesto se integra en un sistema industrial completo de realidad aumentada (RA), llamado ARgitu, empleado para el guiado en el mantenimiento. El sistema genera y presenta información virtual y aumentada, incluyendo las herramientas

necesarias para el desarrollo de nuevos contenidos y adaptar las aplicaciones de tecnología RA en la industria de fabricación avanzada.

La segunda parte de esta tesis presenta un sistema de SLAM monocular directo, llamado Direct Sparse Mapping (DSM). El método utiliza una formulación directa dentro de una infraestructura de mapeo para localizar la posición de la cámara en la escena y construir un mapa global consistente. Hasta donde sabemos, es el primer enfoque de SLAM totalmente directo que reutilice reobservaciones de los puntos del mapa. Como método directo, no depende de emparejamientos entre puntos y puede trabajar con puntos muestreados a través de las aristas en una imagen – en lugar de esquinas únicamente – y obtener una reconstrucción más descriptiva a pesar de utilizar una representación de puntos dispersa. Además, el sistema es robusto contra escenas con poca textura y desenfoques debido al movimiento. La extensa validación experimental demuestra que la infraestructura de mapeo directa que se propone supera a los enfoques de odometría directa actuales – incluso con cierre de bucle – tanto en la trayectoria estimada como en la precisión del mapa.

Contents

List of Figures

List of Tables

List of Notation

$\mathbf{u}$	2D point
$\mathbf{x}$	3D point
$\mathbf{T}$	Rigid transformation
f	Focal length
$\mathbf{c}$	Camera centre
$\mathbf{K}$	Camera intrinsic matrix
π	Projection function
I	Image
ρ	Point inverse depth
ξ	Camera pose parameters
(a, b)	Camera affine light
ζ	Optimization parameters
$\mathbf{r}$	Residual vector
$\mathbf{W}$	Weight matrix
μ	Distribution mean value
σ	Distribution scale value
ν	Student's t-distribution DoF
$\boxplus$	Composition operator
$\mathbf{H}$	Hessian matrix
$\mathbf{g}$	Gradient vector
$\bar{(\cdot)}$	Homogeneous coordinates
$\hat{(\cdot)}$	Skew-symmetric matrix
$(\cdot)^\vee$	Inverse of the hat map

Part I

Introduction & Background

Chapter 1

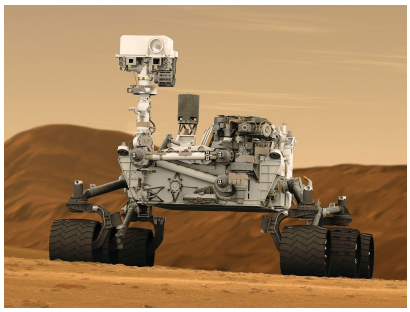
Introduction

1.1 Introduction to Visual Perception

We, as human beings, perceive the world that surrounds us using our senses. This allows us to interact with physical objects and, even, explore the world around us. For example, we are able to localize ourselves in an arbitrary room or recognize many different objects. To do so, we use our senses to receive different physical signals which are interpreted by our brain. Among all our senses, sight is the most important. It captures the 3D world observed by our eyes.

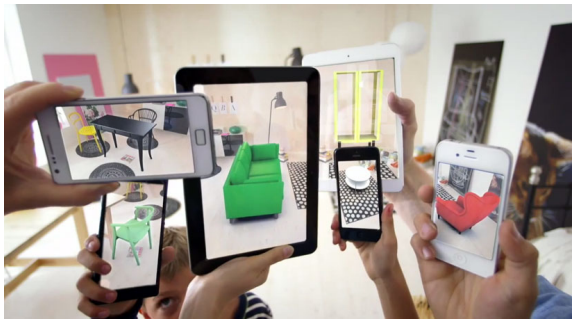
Computer vision is the scientific field that tries to mimic human sight using computers. In computer vision, human eyes and brain are substituted by cameras and computers respectively. Cameras capture and process light to form a computer-friendly representation of the world using digital images. As human eyes, cameras include optical sensors and lenses to help capturing light. Researchers in computer vision study different image processing techniques to extract information from them. The ultimate goal is to obtain an artificial sight sense applicable to many different fields, such as industrial robotics, augmented reality, medicine, cars or even space exploration. Fig. 1.1 presents some real examples of these applications.

Nowadays, digital cameras are in general small, low-power and easy to use. Computer vision applications usually rely only on one or several cameras, a processing unit and their power supply. Many vision applications rely on the light reflected by the environment. Thus, computer vision provides a non-invasive perception solutions which can be adapted to many environments and applications with an affordable and readily available hardware. For instance,



(a) Mars rover. ©Nasa

(b) Autonomous car. ©Uber



(c) Mobile augmented reality. ©Ikea

Figure 1.1: Examples of different computer vision applications. (a) illustrates the NASA rover developed to explore Mars. It includes cameras for obstacle avoidance and autonomous navigation. (b) shows an autonomous vehicle developed by Uber. (c) presents an augmented reality application for interior decoration by Ikea.

mobile devices and mobile robots benefit from the previous properties allowing to build more efficient devices with smaller sizes and longer life operations.

Computer vision applications can be built using many different configurations of cameras obtaining information from different parts of the electromagnetic spectrum. One of the most common types of camera are RGB matricial cameras. These cameras capture color information from the visible spectrum. A chip collects photons in discrete elements, or pixels, forming a 2D matrix. These cameras can be configured for different applications in different configurations: single-camera configurations are also called monocular systems; two-cameras can form a stereo-vision system capable of capturing both the color of each pixel, and its depth as well (the distance of each pixel to the camera set).

RGB-D cameras have the same capabilities, but they usually emit a patterned infrared light or use time-of-flight technology to measure the distance to each pixel. Other systems such as LIDAR, emit an infrared laser light to measure the distance to points very accurately. Nowadays, they are still quite expensive and cumbersome to use due to the high density of information captured.

The specific hardware configuration for a computer vision application should be selected depending on its specific requirements. In this thesis, we focus on monocular RGB systems, as they are one of the most basic configurations and usually all the technology developed for them can be extended to other alternatives.

We focus on how a computer vision system can be used to interpret its surrounding environment from three different points of view:

1. **Camera motion**: refers to the ability to estimate how a camera is moving through the environment. This means to estimate the 6 DoF (3 DoF for position and 3 DoF for orientation) of the camera, also known as *camera pose*. If the camera pose is estimated each new frame, we talk about camera tracking with respect to a either a global or local reference. In this case, the focus is set on camera motion neglecting the elements that compose the scene.

2. **Object recognition**: refers to the ability to detect and localize the position of a 3D target object with respect to the camer from a single image. In this case, the focus is set on specific elements of the scene and their relative pose to the camera.

3. **Scene reconstruction**: refers to the ability to estimate a 3D representation of the environment from 2D images, also known as *3D map*. The map can be represented in many different ways such as a point cloud or a triangular mesh. In any case, its construction requires to estimate the 3D location of the components of the map (points in the case of a point cloud). In this case, the focus is set on the structure of the scene even if the objects contained in the environment are not identified.

In many applications, such as in the case of a robot moving in a room, we need both the camera motion (to locate the robot) and the scene reconstruction (to locate obstacles) simultaneously. In this case, we are discussing about the *visual simultaneous localization and mapping* problem (VSLAM). In contrast to stereo systems or RGB-D cameras, monocular systems cannot perceive the 3D

structure of the environment using a single image without any prior knowledge of the scene. In this case, the stereo image pair needed to reconstruct the scene is obtained from inter-frame motion, which allows to recover the 3D geometry using triangulation. As we will see later, monocular cameras have other difficulties that will have to be addressed.

In summary, this thesis deals with the problem of object recognition and VSLAM using a monocular camera. We address these problems in real-time which requires to be estimated at camera frame rate (usually around 30Hz). In this context, there are two main applications which are directly related with both these technologies: augmented reality and mobile robotics.

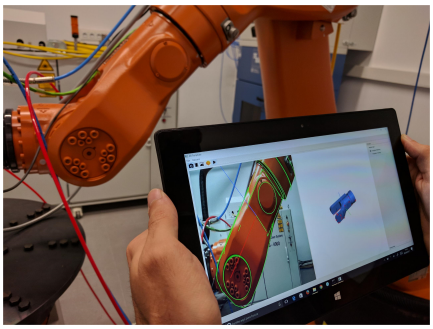
1.1.1 Augmented Reality

Augmented Reality (AR) is the technology that aims to integrate virtual objects within a real image to create the feeling that virtual and real objects coexist in the same world. AR systems require to perceive the environment and its components to properly align virtual elements in the image. One crucial aspect in AR systems is to trick the human brain into thinking that virtual objects belong to the real world while the camera is moving through the environment. This requires to precisely estimate the position and orientation of the camera with respect to the environment in real-time, which we previously labelled as camera motion or object recognition. Once we know the camera pose, we can realistically render virtual object with the appropriate perspective projection.

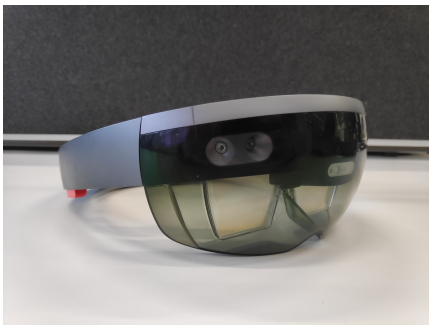
AR technology can be applied in many applications, such as games and films. The following sections present some of the industrial fields in which AR has spread.

Assembly and Maintenance

AR holds an important promise by helping workers in an evermore challenging workplace. Regarding advanced manufacturing, many authors (Palmarini et al., 2018) have demonstrated the benefits of AR-based solutions for guidance in maintenance and assembly tasks in industry. The use of AR systems allows a technician to visualize the spatial layout of all the objects that compose a task and any relevant information about them. Virtual annotations assist during the whole process, e.g. the maintenance of complex machinery (see Fig. 1.2a). For



(a) AR with a robot arm



(b) Mixed reality glasses. ©Hololens

Figure 1.2: On the left an example of AR maintenance task and on the right the Hololens head-mounted display.

now, the prototype systems use mobile devices or head-mounted displays such as the Hololens of Microsoft (see Fig. 1.2b).

Medicine

Medicine is another promising field for AR technology. In this case, virtual annotations assist the surgeon during an operation, e.g. to reduce the number of incisions. In contrast to assembly and maintenance tasks, these applications require to handle with deformable environments and achieve higher accuracy levels. For example, during an engine assembly task the accurate location of a screw could not be so important but during a surgical procedure the location of an specific tissue could be crucial for the success of the procedure (see Fig. 1.3). Currently, there are some works that allow tracking the camera position (Lamarca and Montiel, 2018) and obtaining real-time reconstructions of the tissue during its deformation (Leizea et al., 2017).

1.1.2 Mobile Robotics

Mobile robotics refers to a robot with locomotion capacities (see Fig. 1.4). This type of robots includes many kinds of autonomous systems such as AGV (Autonomous Guided Vehicles) that transport materials, tools or other robots in industry, autonomous cars and drones.

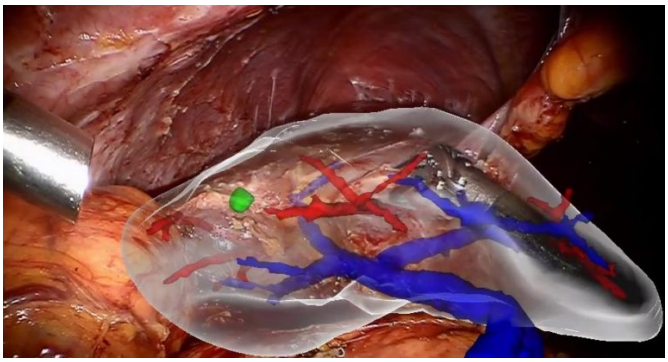


Figure 1.3: Augmented reality based surgery (Plantefève et al., 2016).

In contrast to most industrial robots, the base is not anchored to an specific space location and, thus, they have the ability to move through the environment. When we say that a mobile robotic is *autonomous* we mean that it has the ability to navigate in an arbitrary and unknown environment without requiring any additional external input. An autonomous mobile robot needs the capability of locating itself and understand the 3D structure of its surrounding environment in order to take the right decisions, such as obstacle avoidance or path planning. Thus, autonomous mobile robots are highly dependant on SLAM. Ultimately, a mobile robot could interact with humans and physical objects in the environment, which would require to understand the elements contained in the world.

One of the most common examples is the cleaning robot. Initially, a cleaning robot was designed to arbitrary change its direction when it collided with an obstacle. Currently, however, they build a 3D model of the environment and they plan the most efficient path to optimally clean all the area. This implies exploring unknown areas, path planning, obstacle avoidance, etc. Consequently, it is very important to accurately localize the mobile robot and obtain the best representation of the structure so it can take the most optimal decisions. It is also important to take into consideration the computational time so the estimations are updated throughout the operation without delay.

Finally, autonomous vehicles are also a promising field where SLAM plays an important role. An autonomous car should be able to navigate from an unknown environment without colliding with external objects, other vehicles and, more importantly, persons. At the same time, it should understand the 3D structure of the road and its components, such as other vehicles, traffic signs or pedestrians. Similar to medical applications, in the case of autonomous vehicles

Figure 1.4: Left: an autonomous mobile robot equipped with an RGB-D camera and a laser scan. Right: an autonomous flying drone equipped with an RGB camera ©DJI.

the visual system must provide reliable estimates because it could lead to serious consequences otherwise. There are some works related to this field (Usenko et al., 2015) but they are still far from being robust and complete solutions with all the required capacities.

1.2 Motivation

Nowadays, many companies see AR and autonomous mobile robotics as two important tools to provide new services related to their products and manufacturing processes. Indeed, the so called Industry 4.0 includes AR and autonomous mobile devices as part of their main components. The idea is to create factories in which machines are improved with new sensors and connectivity capacities, connected to a central system that can control, visualize and take decisions of the whole production line on its own.

In this context, AR works as a novel interface between people and machines to provide communication and cooperation capabilities in real-time. For example, the factory itself could inform a technician about an expected failure and AR would assist the technician during the whole maintenance process with virtual information in situ. Consequently, the impact of the failure in the production line would be reduced saving production errors and cost. On the other hand, autonomous mobile robotics can improve the flexibility of the factories with automatic and modular structures. In this way, the system would be able to make decisions on its own and adapt the production to each situation as autonomously as possible.

As we have already seen, AR and autonomous mobile robotics match perfectly with computer vision. Consequently, there is no doubt that any of these technologies should include cameras in their assemblies. Besides, both augmented reality and autonomous mobile robotics have the same problem in common: establishing the pose of a moving device, such as a headset or a robot, and understanding the world around it. For instance, the quality of the user experience in an AR application is directly related with the stability of the virtual annotations anchored in the image.

Despite many recent technological advances, several challenges remain that limit the wide adoption of these technologies in industrial applications:

- AR-based systems require suitable authoring tools for the development of new contents, such as virtual annotations and their animations. Current frameworks usually require the use of complex animations and design tools that in many cases require advanced programming. While this is a minor issue in some applications, such as games, it limits the adoption of AR in small and medium size companies that may lack personnel with the required skills. In addition, there are also limitations in the

available hardware for AR. Many devices such as head-mounted displays are, in many cases, intrusive and not suitable for a long use in ergonomic conditions or in situations that require safety.

- Autonomous mobile robots require suitable algorithms to understand their surroundings and take decisions on their own. Current mobile robotics are designed to follow a predefined path to complete a task. This requires to re-program the robot each time the task is modified which needs qualified personnel with the required qualifications.

The following presents a more detailed explanation of the technical challenges – regarding visual perception technologies – that limit the establishment of AR and autonomous mobile robotics in the industry.

Object recognition for AR in industry

Object recognition enables to understand the elements contained in the environment using just a single image. It does not require to adapt the environment and allows artificial devices, such as mobile robots, to interact with the environment. A very common solution is to use visual landmarks (*features*) detected in the image that come from textured surfaces in the scene. The object position is established by matching those features with a preprocessed database of images of the object. However, these alternatives are optimized for objects with patterned surfaces which are not usual in industrial environments. Besides, the large amount of highly structured data required during the training phase makes them unsuitable for a direct industry application.

Industrial environments are characterized by cluttered scenes with uncontrolled illumination changes (see Fig. 1.5). They usually contain objects with non-Lambertian surfaces (for example metallic). In this case, the brightness of a point of the surface varies with respect to the viewing angle. These surfaces are not well suited to most common object detection algorithms. However, in this context, it is very common to know the CAD model of the objects of interest of the scene. CAD model are a rich source of distinctive elements such as edges and corners that can help overcome the challenges of detection due to reflexions.

Figure 1.5: Example of a real industrial environment. Image provided by Ekin S. Coop.

Visual SLAM for autonomous mobile robotics in industry

The nature of an autonomous mobile robot is to explore unknown environments. This implies to perceive the world around the robot and take decisions based on the structure of the 3D world around it, such as path planning. Besides, an autonomous mobile robot usually ignores any prior information regarding the elements of the scene and, thus, object recognition technologies are not a suitable solution. In this context, visual SLAM provides a straight solution where the structure is estimated from the robot motion and vice versa.

In contrast to object recognition that works with a single image, visual SLAM exploits video streams which nowadays are easily obtainable with consumer cameras. They take advantage of the temporal coherence and assume small camera motions between consecutive frames. They are able to apply predictive algorithms to reduce the computational budget while maximizing the information gain. However, using only temporal constraints leads to very inaccurate results.

It is very common to apply non-linear optimizations taking into consideration past information too. As a result, current approaches to visual SLAM can be referred as online methods where the estimates are incrementally corrected as new information is obtained.

Like other computer vision algorithms, visual SLAM algorithms have traditionally reduced the image data to a sparse set of feature observations correlated across different images. Although this framework provides enough information to accurately estimate the camera motion, the resulting reconstructions are rather poor. The estimated map is usually composed of a very sparse set of features which limits its applicability to real problem. For instance, an autonomous vehicle could estimate its position precisely, but it would not be able to identify if there are any obstacles along its trajectory. This is a severe limitation for many applications and it must be dealt with. In addition, this framework relies on feature repeatability and, thus, it requires enough textured surfaces in the environment.

1.3 Objectives

The main objective of this thesis is the development of new computer vision based perception algorithms using monocular systems that enable new augmented reality and robotics – especially mobile robotics – applications in industrial settings. As discussed above, two main problems have been identified and addressed in this work:

1. Regarding object recognition, industrial settings usually contain objects whose characteristics are not easily handled with state-of-the-art methods. They lack textured surfaces, their appearance can change due to dirt, illumination changes or the presence of non-Lambertian surfaces, and are usually placed in occluded positions with cluttered backgrounds. We propose the following sub-objectives for this problem:

   - Development of a monocular solution for the detection of objects in the industry based on the CAD model and without the requirement of complex training phases or costly capturing data processes.
   - Integration of the detection algorithm in a complete tracking framework for the development of AR applications.
   - Development of a full industrial AR application for guidance in maintenance operations.

2. Regarding visual SLAM methods, current approaches recover a low point density 3D map with a limited capability to describe the structure of the environment.

   In general, they are focused on estimating the camera location as accurately as possible, but they neglect the quality of the map. We propose the following sub-objectives for this problem:

   - Development of a robust and accurate monocular visual SLAM solution capable of handling challenging industrial situations such as motion blur and untextured scenes.
   - Development of a mapping solution with the ability to generate more descriptive, accurate and consistent reconstructions of the environment for practical applications.
   - Evaluation of the proposed solution with respect to both the camera localization and accuracy of the reconstructed map.

1.4 Contributions

On the basis of the previous objectives, this thesis focuses on researching solutions for object recognition and visual SLAM in industrial environments. The main contributions for each of the research developments of this thesis are detailed below.

1. **Review of 3D computer vision techniques**

   We present a complete and lightweight summary of different 3D computer vision techniques, including the results provided by the community during many years of research that led to the most influential algorithms in computer vision. Normally, these works appear spread over the massive literature, but we have gathered them up in a chapter. We start from the most basic concepts to the most complex ones, giving in each case the required details to make a straightforward implementation. We believe it can serve as a guide for both new and experienced researchers, and contribute to the wide adoption of advanced techniques, such as direct methods, in real world computer vision applications.

2. **Model-based object recognition for guidance in industrial maintenance**

   We propose a novel model-based object recognition method that uses geometric properties of the CAD model. More precisely, it uses a combination of model circles and edge templates which are automatically extracted during a pre-processing stage. Thus, it does not require user intervention. The method uses correspondences between model circles and image ellipses to reduce the search space and estimate an initial object location hypothesis. Then, it uses the model shape in the form of edge templates to solve the revolution symmetry ambiguity. The resulting method does not rely on the texture of the scene and it is able to handle the challenging conditions found in industrial environments. We additionally integrate the proposed approach into a tracking framework that exploits the temporal coherence using the same geometric features. Finally, we present a full AR application for guidance in maintenance, called ARgitu. It generates and presents virtual and augmented information, including

the tools required for the development of new contents. ARgitu uses the proposed recognition and tracking pipeline to align virtual elements in the image.

3. **Direct Sparse Mapping (DSM)**

   We propose DSM, a novel direct monocular SLAM systems. As a direct formulation, it does not rely on traditional feature matches (with descriptors) and works directly with pixel intensity values of images. In contrast, points are associated to only one frame and correspondences are recomputed as part of the optimization. Thus, it does not require points to be recognizable on their own and can work with points with a locally high gradient module, such as edges and weak intensity variations. As a result, DSM can handle strong motion blur and low textured environments compared to traditional indirect approaches.

   In contrast to current state-of-the-art direct approaches that are only able to perform visual odometry with a temporary map, DSM uses a mapping framework to build a consistent global map. It uses the same objective function and map points for all the tasks: initialization, tracking and mapping. Up to our knowledge, it is the first fully direct SLAM approach to reuse map point reobservations. To obtain this, DSM builds a persistent map and combines photometric bundle adjustment (PBA) with covisibility constraints to handle map point reobservations from already visited scene regions. Instead of using feature matches as indirect approaches, the covisibility is obtained from a novel combination of geometric and photometric constraints. The result is a more consistent, complete and dense reconstruction with provides a richer description of the environment.

   We have extensively evaluated the solution in a public available dataset achieving the most accurate results up to date for a direct method. For the first time, we have measured both the precision of the camera trajectory and map reconstruction. Finally, we have published our implementation as open-source code.

1.5 Dissemination

1.5.1 Publications

The research carried out in this thesis has generated the following peer-reviewed publications:

- Zubizarreta, J., Aguinaga, I., Amundarain, A. **A framework for augmented reality guidance in industry.** In *The International Journal of Advanced Manufacturing Technology* (2019) doi:10.1007/s00170-019-03527-2
- Zubizarreta, J., Aguinaga, I., Montiel, J.M.M., **Direct Sparse Mapping.** In *arXiv:1904.06577* (submitted to IEEE Transactions on Robotics, 2019).

1.5.2 Open-Source Software

We have released the following open-source software:

- **DSM** (`https://github.com/jzubizarreta/dsm`), *Direct Sparse Mapping*

1.5.3 Videos

Demonstrating videos of DSM:

- EuRoC MAV dataset: `https://youtu.be/sj1GIF-7BYo`

1.6 Thesis Outline

This thesis is divided into 6 chapters. Chapter 1 has introduced the situation of visual perception technologies and the research fields in which this thesis is focused. We have also presented the factors that have motivated this work together with the main objectives and the application fields. As this thesis is based on object recognition and visual SLAM technologies, Chapter 2 presents a more in-depth classification of the approaches currently in the state of the art. Chapter 3 presents an overview of the technical foundations which are relevant for the remainder of the thesis. In particular, we describe the intrinsic geometry involved in computer vision problems as well as different representations for cameras and 3D structure. Furthermore, we provide the required optimization tools to solve the proposed problems. Chapter 4 deals with the problem of object recognition in industrial environments and presents a complete framework for AR guidance in industry. Afterwards, Chapter 5 presents a fully direct SLAM approach with map reuse capabilities. Finally, Chapter 6 presents the conclusions of this work and a number of interesting future research directions.

Some additional technical concepts are presented in detail in the appendices at the end of this document. They have been excluded from the main body of the document to facilitate the reading flow of the work.

Chapter 2

Background

In the last decades the popularity of computer vision has grown considerably due to its large range of possibilities. For instance, almost any mobile device includes one or multiple cameras and integrates computer vision algorithms to increase its capacities, such as screen unlock using face recognition. Another example are filming drones which are able to identify and follow a target actor enabling impressive recordings with very difficult shoots.

This has lead to an increase of the research effort with the development of many different techniques. In this chapter we provide a classification of different computer vision techniques focused on the two main research areas of this thesis: object recognition and visual SLAM. As many of the techniques can be applied to many different visual sensors, we maintain the classification independent of the selected sensor.

2.1 Object Recognition

The main goal of object recognition is to locate the camera position and orientation, also known as camera pose, with respect to target objects in the scene using a single image. This requires to distinguish the elements that compose the scene and estimate their position and orientation in the 3D space. In general, it is very common to previously have some kind of information about the object of interest, such as specific patterns or 3D shape. This knowledge is exploited beforehand to extract information about the object (training) and obtain the most descriptive representation to recognize the object in an image. Thus, it is important to evaluate not only the recognition performance but the whole process, including the training. For instance, the training could become

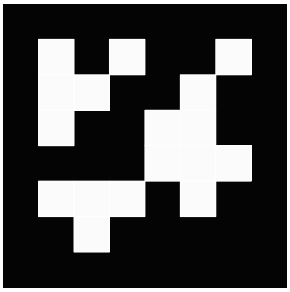


Figure 2.1: Marker example formed of black and white squares.

impractical for real applications due to the amount of user experience required for its implementation in industry. Finally, this section deals only with rigid objects which implies that the size and shape of the object do not change over time, even when external forces are applied. However, the camera, the object or both can move freely in space.

2.1.1 Marker-based

Marker-based systems work with easily identifiable patterns (see Fig. 2.1) that are artificially added to the scene. The idea behind markers is to add predefined visual features to the world so they facilitate the recognition task. They have been widely used in industry and AR applications due to their simplicity, low cost and good performance. The main drawback of marker-based systems is that they require to adapt the environment which is not always possible. Besides, marker-based systems cannot handle occlusions and they fail when the marker is not completely visible.

The marker pattern is normally composed of simple geometries, such as squares and circles, printed in black and white stickers (Pagani et al., 2011). The configuration of the geometric elements inside the marker codify a unique identifier that allows to distinguish one marker from the others. In order to obtain the position of the camera with respect a maker it is common to perform the following steps: (1) threshold the input grayscale image to segment the marker from the rest of the image; (2) match the maker with respect to a pre-compiled dataset comparing the unique identifier; (3) estimate the camera location knowing the correspondences between 2D-3D vertices in the marker. As we know the marker is planar, it is easy to obtain the camera location using an homography (Hartley and Zisserman, 2004).

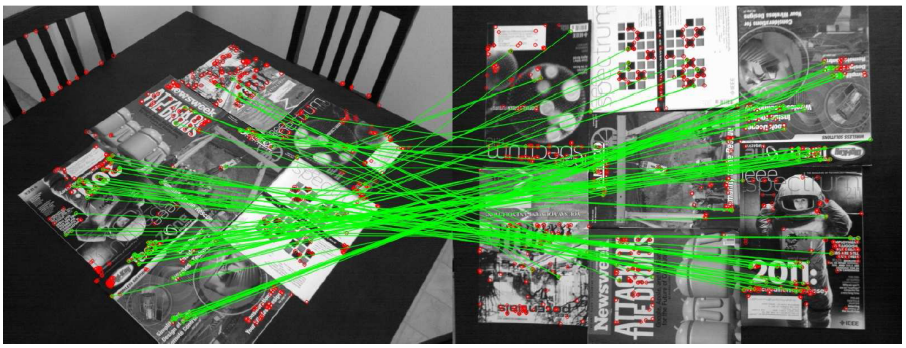


Figure 2.2: ORB feature matching (Rublee et al., 2011).

2.1.2 Textured-based vs. Geometry-based

Textured-based methods rely on the object's own surface pattern. In this case, the surface must contain distinctive appearance features in order to make it possible to recognize the object in the image. These methods proceed in two steps: First, the input image is processed to extract salient visual features (Rosten and Drummond, 2006, Shi and Tomasi, 1994) which are locally represented using an appearance vector called descriptor, such as SIFT (Lowe, 2004), SURF (Bay et al., 2006) or ORB (Rublee et al., 2011). Descriptors are used to distinguish each visual feature from the others by measuring the distance between the vectors. Second, the extracted visual features are matched against a pre-compiled dataset comparing the descriptors (see Fig. 2.2) and the camera pose is estimated from the 2D-3D correspondences (Lepetit et al., 2009). During the last step, it is very common to use an outlier removal strategy such as RANSAC (Szeliski, 2010). The main drawback of feature-based approaches is that they rely on the repeatability of the selected features and, thus, fail in low textured scenarios, such as industrial environments.

Geometry-based methods do not rely on the texture of the surface of the object but on its shape. In general, they work with edges of the object which are easily identifiable using image gradients (Canny, 1986, K.Vairalkar and S.U.Nimbhorkar, 2012, Topal and Akinlar, 2012). In this approaches, it is very common to compare the input image with a pre-compiled dataset of images of the object in different positions, also called templates. Dominant Orientation Templates (DOT) is a popular method (Hinterstoisser et al., 2010). It measures the similarity between two images by comparing the orientation of the gradients. This method was extended to handle depth sensors in (Hinterstoisser

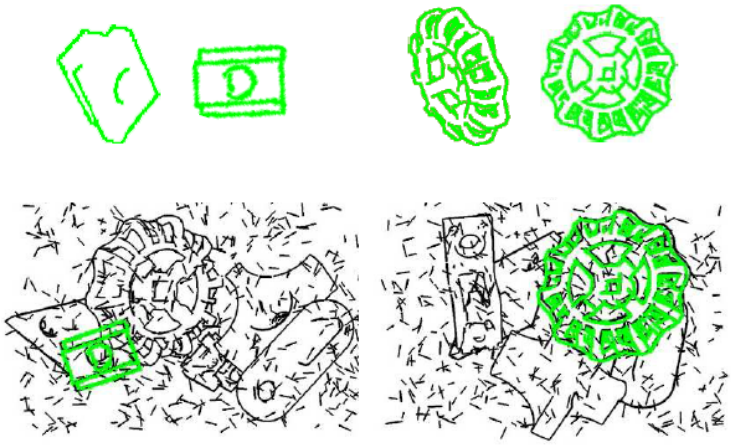


Figure 2.3: Pose estimation of different objects using chamfer matching techniques (Liu et al., 2010). In green the object templates.

et al., 2012b) and color information in (Peng, 2015). Chamfer Matching is another popular method which evaluates the distance between image edges (see Fig. 2.3). It can be efficiently computed using image Distance Transform (DT) (Felzenszwalb and Huttenlocher, 2004). Later, Liu et al. (2010) extended the chamfer matching formulation to include edge orientations which increases its robustness. The main drawback of these approaches is that their performance degrades significantly with cluttered backgrounds and their large search space. As they do not perform any previous matching scheme, the registration is executed by brute force, which considerably slows down their performance. However, geometry-based approaches are more suitable for untextured objects.

2.1.3 Model-based vs. Image-based

As we have already seen, most approaches work against a pre-computed dataset, which is the training step that we have mentioned before. Thus, we can classify the recognition approaches depending on how the training is performed.

Model-based approaches use a 3D virtual model of the object of interest. The model is available beforehand and contains information about the 3D geometry of the model. Sometimes it also contains information about the object surface, such as its texture or material. The model is usually represented as a triangular mesh, which is normally obtained using a computer-aided design software or

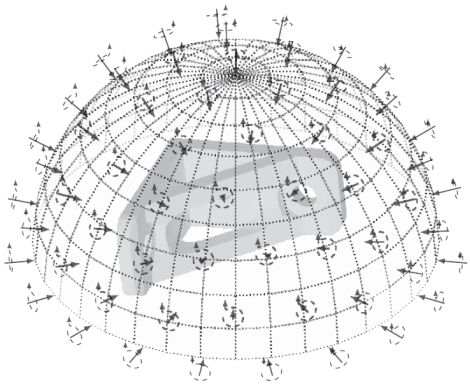


Figure 2.4: Sphere training of a virtual model using an sphere path Bratanič et al. (2015). Each arrow represents a virtual camera from which rich information about the model is extracted.

a 3D scanner. These approaches normally generate a set of virtual cameras around a sphere centered in the object (see Fig. 2.4) (Álvarez and Borro, 2013, Bratanič et al., 2015, Imperoli and Pretto, 2015). For each virtual camera, the required information is extracted, e.g. visual features, 3D visible geometry, edge templates, etc. In this way, the pre-computed dataset will contain a detailed description of the object from many different points of view.

Instead of using virtual cameras, *image-based* approaches work with real images of the object of interest. Thus, these approaches require the intervention

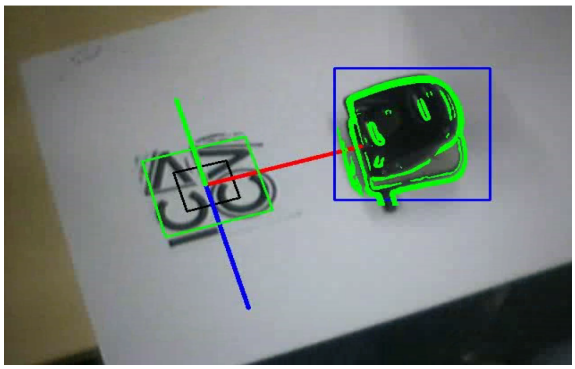


Figure 2.5: Image-based training where a marker is used to estimate the camera pose (Hinterstoisser et al., 2010).

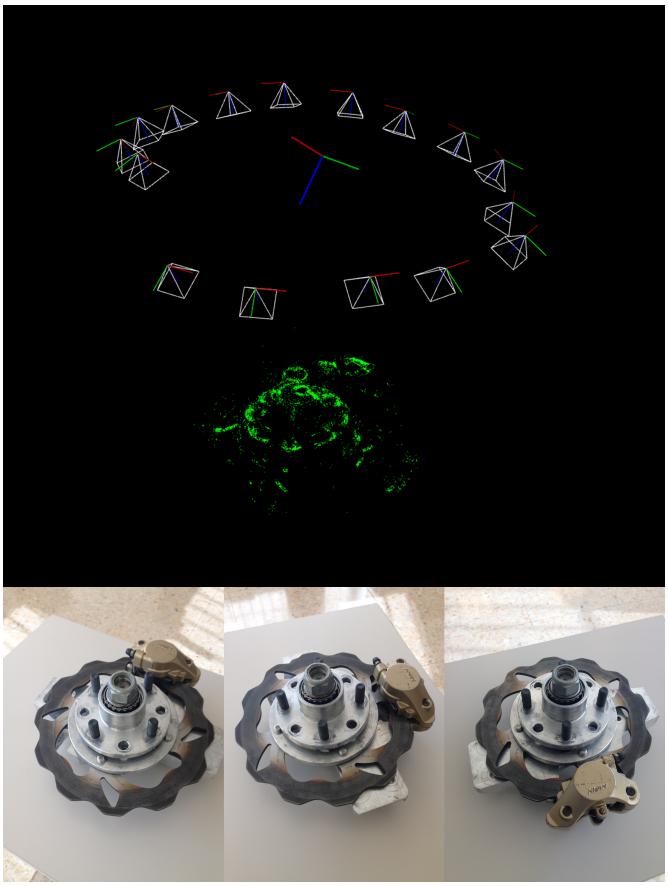


Figure 2.6: Image-based training where the 3D geometry is estimated from SfM techniques. Top row shows the final reconstruction with all the cameras. Bottom row shows some of the images captured by the user and used for the reconstruction.

of the user to capture images of the object from different points of view. In order to obtain the camera location for each view, it is common to use the following two alternatives: (1) use a marker with a predefined configuration with respect the object (see Fig. 2.5) (Hinterstoisser et al., 2010); (2) use Structure from Motion (SfM) techniques (see Sec. 3.4) to reconstruct the 3D model from the images (see Fig. 2.6) (Pillai and Leonard, 2015, Wang et al., 2018). For the latter, it is very usual to use features with descriptors to obtain 2D correspondences between the images and triangulate the 3D position of each feature. Consequently, this kind of solutions are typically oriented to rich textured objects. However, it is

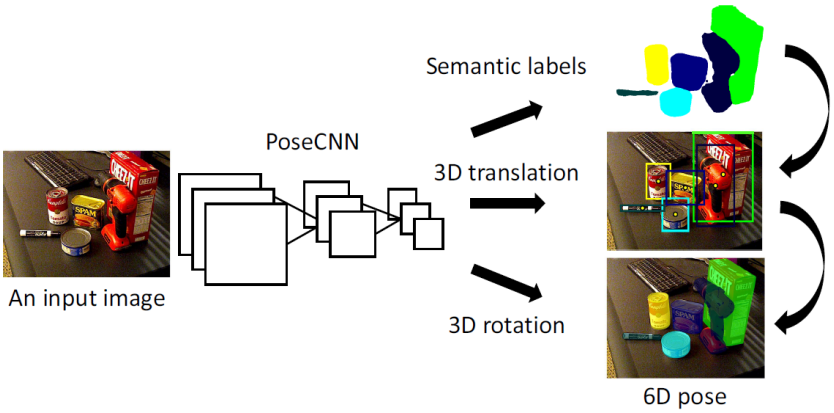


Figure 2.7: PoseCNN for 6D pose estimation. The output of the network is trained to provide semantic labeling, 3D translation estimation and 3D rotation regression (Xiang et al., 2017).

also possible, for example, to extract edge templates from the images and use geometric-based solutions.

2.1.4 Machine Learning-based

Nowadays, there is a large amount of data and computational power at our disposal. *Machine learning* approaches take advantage of this situation to develop algorithms based on statistical models that provide artificial systems the ability to automatically learn and improve from experience, without being explicitly programmed. *Automatically learn* means that the algorithm is able to find the optimal parameters that fit the model without the user intervention, and *experience* refers to the large amount of data (examples) that feeds the algorithm. As a results, machine learning approaches are highly dependant on very large datasets that are required to train the statistical model and are crucial to obtain a general solution.

One of the most popular approaches inside machine learning is *Deep Learning* due to its impressive results. These are more complex techniques that use multiple layers with the aim of analyzing different characteristics in the data. Regarding computer vision, we can talk about Convolutional Neural Networks (CNN) which apply different convolutions in each layer to analyze the images

(Garon and Lalonde, 2017, Li et al., 2018, Tekin et al., 2017, Xiang et al., 2017). For instance, lower layers could identify edges in the image while higher layers could detect human faces. Fig. 2.7 shows an schematic diagram of a 6D pose estimation network. Currently, however, it has been demonstrated (Garon and Lalonde, 2017) that these approaches fail when an object is occluded more than 20%. Besides, the large amount of highly structured data required during the training phase makes them unsuitable for a direct industry application.

2.2 Visual SLAM

The goal of Visual SLAM (VSLAM) is twofold: reconstruct the 3D world and obtain the camera pose within the reconstruction. In general, VSLAM is an incremental process in which new data – in the form of frames – arrives sequentially in time while the camera is moving freely in the space. The problem is stated as a probabilistic model that takes the camera noisy measurements and estimates the 3D geometry (camera pose and structure) without any prior knowledge of the scene. This section presents a classification of VSLAM approaches depending on how the above problem is formulated and solved. Besides, if we want to classify and evaluate VSLAM approaches, we should take both the localization and mapping performance into account. The core of the following classification is inspired by Engel (2016).

2.2.1 Direct vs. Indirect

Indirect approaches work with an intermediate representation of the image, normally in the form of a sparse set of features. These features are matched across images to establish 2D correspondences. In this case, image features are treated as noisy measurements and inserted into a probabilistic model to estimate the 3D geometry (Gomez-Ojeda et al., 2019, Mur-Artal et al., 2015). In contrast to the textured-based methods presented above, VSLAM indirect approaches can also use other kind of matching strategies that exploit the video stream such as optical flow techniques (Buczko and Willert, 2016). Then, the 3D camera motion and feature positions are optimally estimated from the 2D feature correspondences. During optimization, indirect approaches optimize a geometric error, i.e. the reprojection error, which has good convergence properties. The main limitation of these approaches is that they rely on feature repeatability. As a result, lack of texture or motion blur degrade considerably their performance.

In contrast, *direct* approaches work directly with pixel intensities and skip the pre-processing step (see Fig. 2.8). Thus, they do not rely on feature repeatability and tend to be more accurate and robust when the scene contains little texture and with motion blur. This property allows direct approaches to sample across all available data, such as edges and weak intensity variations, which generates a more complete representation of the 3D scene structure (Engel et al., 2016a). During optimization they optimize a photometric error, i.e. the intensity error, following the Lukas-Kanade framework (Baker and Matthews,

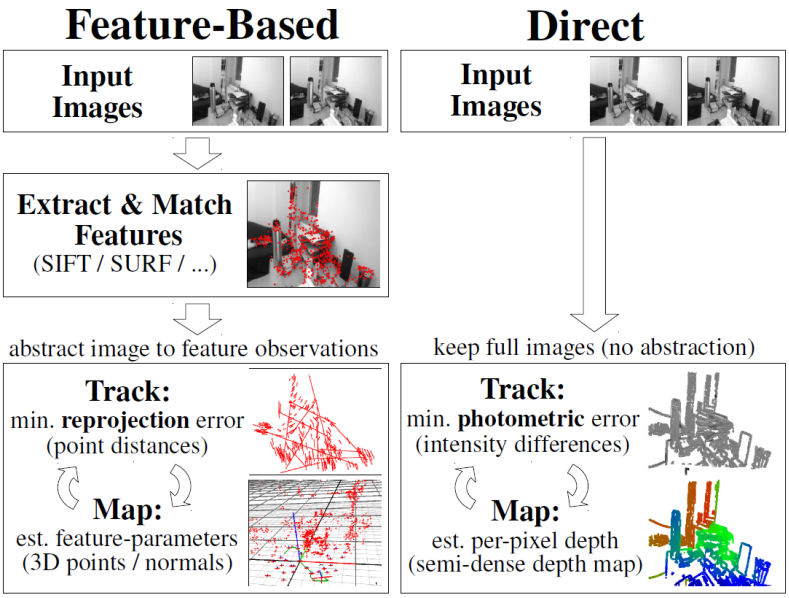


Figure 2.8: Comparison between feature-based (indirect) and direct approaches (Schops et al., 2014). Direct approaches skip the feature extraction and matching steps.

2004). They do not work with explicit correspondences and use image gradients to guide the optimization in the right direction. As a result, the main limitation of direct approaches is their small convergence radius due to the high non-linearity of image data. In this case, it is very common to use a framework where only consistent data is used, e.g. close in time estimates that are almost not affected by accumulated error (drift).

2.2.2 Dense vs. Sparse

Dense methods use all the available pixels in the image. They obtain a very representative reconstruction at the cost of high computational and memory demands. As a result, they require GPU's to run the algorithms in real-time. It is very common to see this kind of approaches together with direct techniques (Stühmer et al., 2010). In contrast, *sparse* methods use only a small percentage of pixels in the image which allow using CPU's only. However, they obtain a very poor representation of the scene. Traditionally, sparse approaches have been

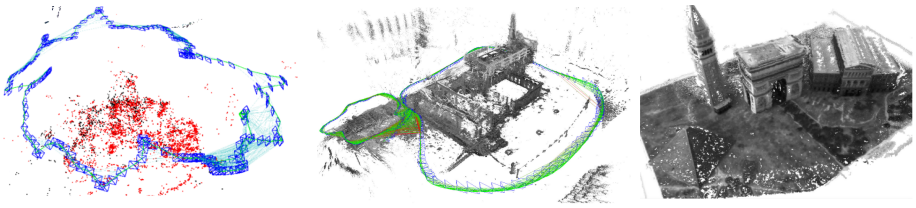


Figure 2.9: Density comparison. From left to right: sparse (Mur-Artal et al., 2015), semi-dense (Engel et al., 2014) and dense (Pizzoli et al., 2014).

used together with indirect techniques using corners as image features (Klein and Murray, 2007). Recently, *semi-dense* approaches have been presented which use all pixels with high gradient magnitudes. These are not so computationally demanding and estimate a fairly complete reconstruction. Similar to dense approaches, it is usual to combine semi-dense methods with direct techniques (Engel et al., 2014). Fig. 2.9 shows a comparison of the obtained reconstruction for different density techniques.

When working with dense approaches, there are usually many pixels without texture information, such as points in a white wall. In this case, it is very common to add a geometric prior into the formulation which connects pixels around a local region. Typically, a planar or an smooth surface condition is applied (Concha and Civera, 2015, Newcombe et al., 2011).

2.2.3 Filtering vs. Bundle Adjustment

The solution to a VSLAM problem involves finding the full maximum likelihood which estimates the model parameters that maximize the probability of obtaining the actual measurements. This framework grows every new frame and, thus, becomes quickly intractable in real-time. Typically, there are two possibilities to overcome this issue, filtering and bundle adjustment, that can be distinguished in how they manage the problem structure internally. Fig. 2.10 illustrates the overall problem and these two alternatives in the form of a graph. A more in-depth analysis of these two techniques can be found in Strasdat et al. (2010), in which this section is inspired.

Filtering methods continuously update a joint probability distribution over all the selected parameters following the Kalman filter (Welch and Bishop, 2006). New measurements are inserted into the distribution with high uncertainty and

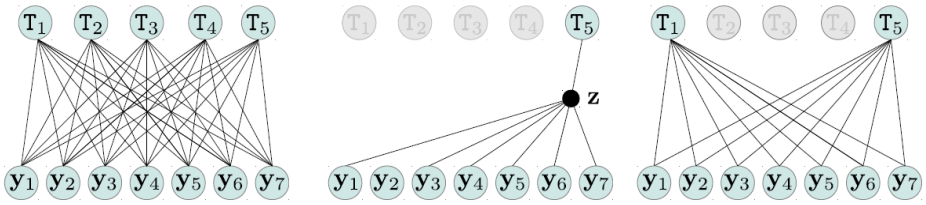


Figure 2.10: Optimization comparison in the form of a graph. The nodes T_i and y_j represent the camera poses and the 3D position of each feature respectively. Edges between nodes represent the 2D image measurements. Left: full VSLAM problem where all cameras and features are inserted into the optimization. Middle: sequential filtering where only the latest camera is retained. Since old cameras are removed, an statistical correlation is set between 3D features to avoid losing information. Right: windowed bundle adjustment where only a set of cameras and features are inserted into the optimization. (Strasdat et al., 2010)

contribute to reduce the uncertainty of the whole model. All cameras other than the current one are marginalized and removed from the state vector. This creates an statistical correlation between world features to avoid losing past information and requires to fix their linearization (i.e. Jacobians are not re-evaluated). In this way a compact representation is achieved that does not grow over time but the graph quickly becomes fully interconnected. Consequently, filtering approaches scale poorly with the number of feature variables ($\mathcal{O}(m+n)^3$ where m is the number of cameras and n the number of features) as they require to store and update a joint distribution over all the interconnected variables. Thus, the efficiency of filtering approaches is limited by the number of features in the map (Davison et al., 2007).

Bundle adjustment approaches retain all the information in the form of a non-linear objective function (Triggs et al., 2000). To handle the optimization in real-time, only a small set of past cameras is retained (keyframes), typically in the form of a sliding window of most recent cameras. The rest of cameras are kept fixed and do not contribute to estimates. In contrast to filtering, the resulting optimization remains relatively efficient even if the number of features grows ($\mathcal{O}(m^3+m^2n)$). However, bundle adjustment methods require to re-evaluate each observation whenever they are updated, which limits the number of observations that can be inserted into the model (Klein and Murray, 2007). Strasdat et al. (2010) concluded that bundle adjustment approaches achieve higher accuracy

than filtering ones, specially for large problems where significantly more features are used.

More recently, several works have been presented that mix both alternatives (Engel et al., 2016a, Leutenegger et al., 2015, Qin et al., 2018). These approaches use a sliding window, where the bundle adjustment is performed, and marginalize all the rest of parameters that leave the optimization window. In this way, they have the ability to summarize and take into account all the old information in the marginalized term while continuously updating a large number of parameters in the windowed optimization. As we show in the following section, using a marginalization strategy has further disadvantages that makes it unsuitable for maintaining a global map.

2.2.4 Odometry vs. Mapping

Odometry methods are only interested in estimating the camera location as accurately as possible each time step. Although both localization and mapping go hand in hand, odometry approaches neglect the quality of the map. This means that they do not care about the map accuracy or its usage in real world applications, e.g. obstacle avoidance during navigation. Typically, odometry approaches build a local map to precisely estimate the camera pose. To obtain this, they use a sliding window with a marginalization strategy as presented in the previous section, which maintains a lightweight optimization (Engel et al., 2016a, Leutenegger et al., 2015, Qin et al., 2018). The main drawback of odometry approaches is that if the camera revisits already mapped areas, they cannot reuse map features as they are included in the marginalization term and they are forced to duplicate them. This causes motion drift and structure inconsistencies. However, in some applications, such as autonomous driving in a highway, this may not be an issue as it is probable that we will not traverse two times the same area and other sensors, such as GPS, could complement this task.

Mapping approaches on the other hand aim to estimate both the camera location and the structure as accurately as possible. They build a persistent map and continuously process map feature re-observations. As a result, they reduce the drift in the estimates. Instead of using a sliding-window and marginalization, they retain old parameters fixed in the windowed bundle adjustment and select active parameters according to covisibility criteria, i.e. cameras that observe several map features in common. This strategy allows Jacobians to be re-evaluated and, thus, they are able to reuse existing map information. Typically,

mapping methods have been used together with indirect approaches since they can correlate far away images and easily correct the accumulated drift due to their good convergence properties (Klein and Murray, 2007, Mur-Artal et al., 2015, Strasdat et al., 2011).

2.3 Discussion

Despite recent advances in this two kind of technologies, it is still challenging to adapt them to real applications. This section presents the main advantages and drawbacks of the above technologies and provides the reasons why we have selected or discarded each of them for this thesis.

Object recognition. This work is focused on object recognition in industrial environments. These are characterized by cluttered scenes with uncontrolled illumination changes and they usually contain objects with non-Lambertian surfaces. Thus, it is very important to evaluate each technology in this situation.

Markers provides a simple and accurate solution in real-time. However, these solutions require to adapt the environment. Besides, they are sensitive to dirt and occlusions: they only work well when the markers are totally visible by the camera. In fact, this is a common scenario in manufacturing, considering that hands and tools can easily occlude the working space.

Texture-based methods are optimized for objects with patterned surfaces which are not usual in industrial environments. In contrast, geometry-based approaches provide a more robust solution but they fail with cluttered backgrounds and require higher computational times. In our opinion, fusing geometry-based approaches with a previous matching scheme could be the best solution. The matching step would provide an initial guess of the object location and limit the search space only to its vicinity. As a result, the system would be able to skip many wrong object locations by searching locally around the initial guess using geometric-based techniques. It also would reduce the computational time due to the reduced search space. Note that the matching approach should also be developed with geometric-based techniques in order to guarantee a robust performance under untextured environments.

Regarding the training, it is very common to have access to the CAD model of the target object and these are a rich source of distinctive geometric elements such as edges and corners that can help overcome the challenges of detection due to reflections. For example, a large number of man-made objects contain revolution elements such as holes or cylinders, which are easily identifiable in this kind of environments and that are stable under changes of illumination. For these reasons, a model-based training would be more appropriate than an

image-based one that requires the intervention of the user and usually relies on image features with descriptors.

Finally, learning-based methods require very large datasets and cannot handle severe occlusions. They are very sensible to the training data which currently limits their applicability out of controlled laboratory conditions. Although their results are impressive in many areas, their framework is still a limitation for a direct application in the industry, specially in small and medium size companies.

Visual SLAM. This thesis is also about developing VSLAM techniques to obtain not only an accurate localization of the camera but a useful 3D representation of the environment too. From our point of view, this would extend the possibilities of VSLAM approaches to new applications, such as full autonomous navigation with path planning, obstacle avoidance and collaborative capacities.

Direct techniques have proven to be an effective formulation for estimating the scene geometry and camera motion. They avoid all intermediate steps of feature detection and matching, and produce accurate geometry estimates even in poorly textured scenes where indirect methods fail. Besides, since they do not rely on feature repeatability they can work with all image pixels that contain meaningful information such as edges and textured surfaces. As a result, direct approaches use higher density of features even with sparse algorithms and obtain a more complete representation of the 3D structure where the shape of the elements contained in the scene are more easily identifiable.

In this context, bundle adjustment is a very efficient scheme to accurately estimate all the involved parameters and allows working with higher number of features than filtering techniques. Currently, the photometric bundle adjustment (PBA), i.e. bundle adjustment with direct techniques, has only been developed for visual odometry. This is due to the fact that using a mapping scheme would include cameras and features distant in time and, hence, affected by the estimation drift. As a result, PBA would not compensate the drift because of the small convergence radius of direct techniques. In this cases, a very common alternative is to use a multiscale framework that may overcome this kind of issues. In addition, the lack of explicit feature matches makes it even more difficult to select which cameras and features should be included in the photometric bundle adjustment as far in time cameras are not visually correlated in any way. Despite

all these drawbacks, we believe in the potential of using a PBA with a mapping scheme to build an accurate and consistent global map.

Part II

Technical Foundations

Chapter 3

Technical Foundations

A critical step in any computer vision application is understanding how the 3D world maps into the 2D image captured by a camera. Thus, before we start exploiting image data, we need to understand how 2D image pixels are related to the 3D world.

In this thesis, a camera is represented as a 3D coordinate system in space and its corresponding image a 2D plane as defined in Fig. 3.1.

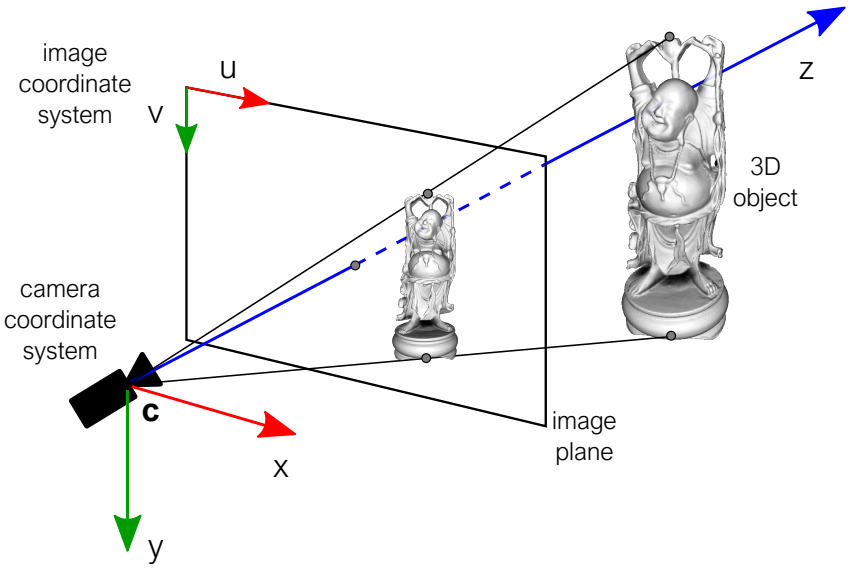


Figure 3.1: Definition of coordinates systems involved in a camera sensor[1].

This chapter presents the notation and basic 3D computer vision concepts used throughout this thesis. First, Sec. 3.1 and 3.2 establish the mathematical tools used to represent the 3D scene geometry. Later, Sec. 3.3 explains how camera sensors capture information from the 3D world and Sec. 3.4 presents how the 3D geometry relates in different images. Finally, Sec. 3.5 introduces the required optimization tools to accurately estimate all the involved geometry parameters.

3.1 3D Geometric Primitives

The real world contains elements with many different and complex shapes. In the context of computer vision, it is very common to simplify scene elements using a more basic geometric representation. This section presents some of the most common geometric primitives used in 3D computer vision.

3.1.1 Points

A point is the basic geometric entity representing a dimensionless object whose only attribute is its position in space. 2D points $\mathbf{u} \in \mathbb{R}^2$ can be represented with two coordinates u, v and we will use them to define pixel locations in the image as

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}. \tag{3.1}$$

A very common option to represent a point in a 3D space $\mathbf{x} \in \mathbb{R}^3$ is to use Cartesian coordinates as

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.2}$$

In principle, the capturing range of a camera sensor is not limited in depth and it is able to capture 3D points at infinity (points in the horizon). The Cartesian representation of points contains a discontinuity at $z = \infty$ which makes handling points close to the horizon more complex. As an alternative,

[1] Buda model obtained from `http://graphics.stanford.edu/data/3Dscanrep/`

(Civera et al., 2008) propose to use the inverse depth, $\rho = 1/z$, given by

$$\mathbf{x} = \frac{1}{\rho} \begin{bmatrix} m_u \\ m_v \\ 1 \end{bmatrix}, \tag{3.3}$$

where $\mathbf{m} = [m_u, m_v]^T$ represents the projected coordinates of a 3D point (see Sec. 3.3.1). The inverse depth has the following useful properties:

- The inverse depth is continuous at $z = \infty$. This enables handling far points, i.e. point close to the horizon, which in some cases can provide very rich information about camera orientation.

- The inverse depth contains a discontinuity at $z = 0$, but this is a rare situation since those point should not be seen by a camera (see Fig. 3.1).

- The inverse depth representation produces a more linear observation model (Sec. 3.4.2). Thus, it suits optimization problems better (Sec. 3.5.5) provinding faster convergence rates.

- If we fix the location of a pixel, the inverse depth allows to represent a 3D point with just one parameter which reduces the dimension of the optimization problem.

If we use the last property and fix the pixel location, this representation requires to associate each point to a reference frame (usually the first one in which it was observed). As a result, using an inverse depth representation of points limits the management of points and cameras separately. If we want to treat points and cameras independently, we need to transform points back to Cartesian coordinates.

Besides, note that the Eq. 3.3 contains a discontinuity at $\rho = 0$, which could make the reader think that, after all, we will still have the same issue as with Cartesian coordinates. However, Sec. 3.4.2 presents an elegant solution to obtain a linear mapping function between two camera views using the inverse depth and exploit all the positive characteristics described above. The inverse depth parameterization will be used in the proposed visual SLAM method of Chapter 5.

Finally, we define the $\bar{(\cdot)}$ operator to express a point in homogeneous coordinates given by

$$\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \text{and} \qquad \bar{\mathbf{u}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{3.4}$$

which is going to be useful for many operations such as rigid body transformations.

3.1.2 Lines

2D lines can be represented using a normal vector perpendicular to the line $\mathbf{n} = (n_u, n_v)^T$ and the distance d to the origin. Any point $\mathbf{u}$ in the line is given by

$$\mathbf{u}^T\mathbf{n} + d = n_u u + n_v v + d = 0. \tag{3.5}$$

The normal vector can also be expressed using polar coordinates as $\mathbf{n} = (\cos\theta, \sin\theta)^T$ where θ is the angle with respect to x-axis.

3D lines can be represented using two auxiliary points $\mathbf{p}, \mathbf{q}$ that belong to the line. Any point in the line can be obtained as a linear combination of these two points as

$$l = (1 - \lambda)\mathbf{p} + \lambda\mathbf{q}. \tag{3.6}$$

Note that this representation has six DoF (three for each point), while lines in 3D space only have four. To overcome this problem many other alternatives have been proposed. However, their presentation is out of the scope of this thesis. The interested reader can find further about them in chapter 3 of Hartley and Zisserman (2004).

3.1.3 Planes

3D planes can be represented using the normal vector to the plane $\mathbf{n} = (n_x, n_y, n_z)^T$ and the distance d to the origin. Any point $\mathbf{x}$ in the plane must follow:

$$\mathbf{x}^T\mathbf{n} + d = n_x x + n_y y + n_z z + d = 0. \tag{3.7}$$

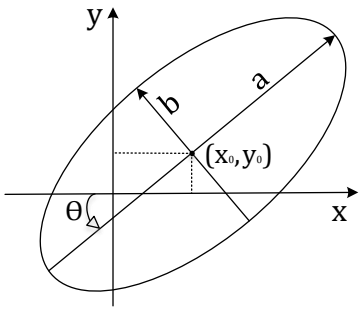


Figure 3.2: Ellipse definition: a and b are the major and minor semi-axis lengths, θ the ellipse orientation relative to the $\mathbf{x}$ axis and (x_0, y_0) the coordinates of the ellipse center.

Similar to 2D lines, the normal vector can be expressed relative to two angles θ, ϕ using spherical coordinates.

3.1.4 Conic Sections

Conics result from the intersection of the surface of a cone with a plane. A conic section can be represented using a quadratic equation as

$$Ax^2 + Bxy + Cy^2 + Dxz + Eyz + Fz^2 = 0 \tag{3.8}$$

or in matrix form as

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0. \tag{3.9}$$

Conic sections can be classified in three different types depending on the value $B^2 - 4AC$:

- if < 0 is a hyperbola,
- if $= 0$ is a parabola, and

- if > 0 is a ellipse,

Circles are a special case of an ellipse when $A = C$ and $B = 0$. In this thesis, ellipses and circles are used as robust geometric features for object recognition and pose estimation in Chapter 4. For a ellipse defined as in Fig. 3.2, the coefficient values are given by

$$\begin{aligned} A &= b^2 \cos^2\theta + a^2 \sin^2\theta, \\ B &= 2(b^2 - a^2) \sin\theta \cos\theta, \\ C &= b^2 \sin^2\theta + a^2 \cos^2\theta, \\ D &= -2Ax_0 - By_0, \\ E &= -2Cy_0 - Bx_0, \\ F &= Ax_0^2 + Bx_0y_0 + Cy_0^2 - a^2b^2. \end{aligned} \tag{3.10}$$

3.2 Rigid Transformations

In this work, we consider the case of a camera moving in space. In this case, the same 3D elements can be observed from different viewpoints. After presenting how 3D world elements are represented mathematically, we can proceed to establish the geometric transformations of those elements between different camera views (see Fig. 3.3). This section presents how the position and orientation of world elements can be established relative to each viewpoint. For simplicity, we will use points but the formulation can be easily extendable to other geometric primitive forms (Hartley and Zisserman, 2004).

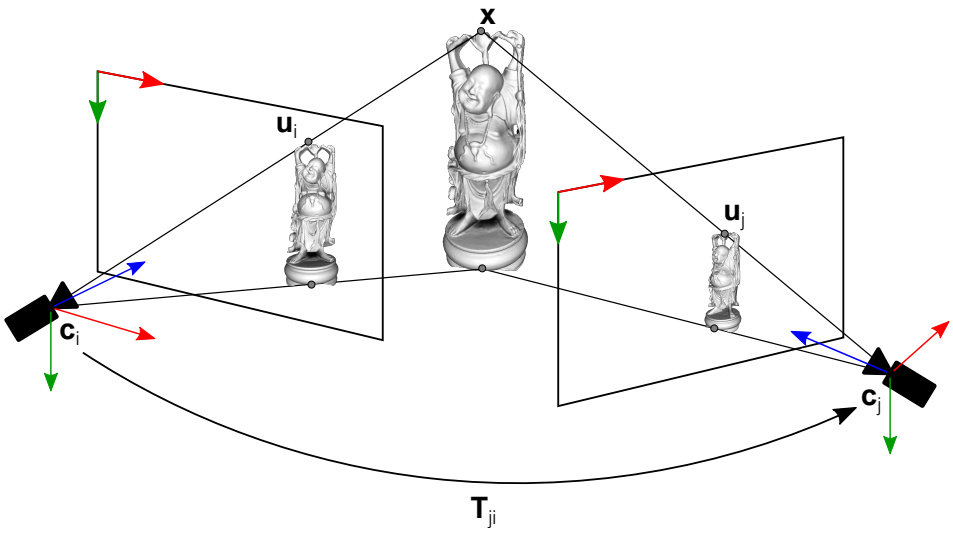


Figure 3.3: Relative representation of 3D geometry.

Given two cameras, the 3D coordinate transformation from camera i to j can be expressed using a 4×4 matrix representation as

$$\mathbf{T}_{ji} = \left[\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right], \tag{3.11}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the relative rotation matrix from the special orthogonal group $SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ is the relative translation vector. As a result, the relative

transformation $\mathbf{T}_{ji}$ is from the the special Euclidean group $SE(3)$ with an overall of 6 DoF (3 for rotation and 3 for translation). The inverse transformation $\mathbf{T}_{ij}$ is defined as

$$\mathbf{T}_{ij} = \mathbf{T}_{ji}^{-1} = \left[\begin{array}{ccc|c} & \mathbf{R}^T & & -\mathbf{R}^T\mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{3.12}$$

In this thesis we will solve different numerical problems that involve estimating the parameters of SE(3) transformations, such as the pose of 3D objects or a moving camera. In the case of using a matrix representation, it means that we have to estimate 12 parameters (9 for the rotation and 3 for translation) with only 6 DoF, which is a clear over parameterization. In the rest of this section, we will present different parameterization solutions that simplify optimization problems with a better representation (less parameters).

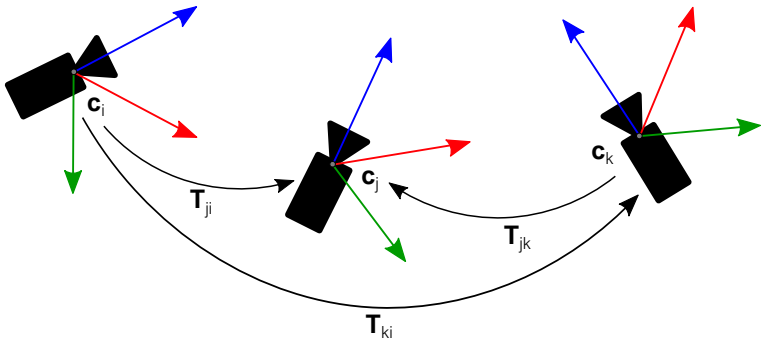


Figure 3.4: Rigid transformations between different poses of a moving camera.

Finally, we define the composition operator $\boxplus$ which transforms the relative coordinates of a given 3D point $\mathbf{x}$ or concatenates two transformations as (see Fig. 3.4)

$$\mathbf{x}_j = \mathbf{T}_{ji} \boxplus \mathbf{x}_i, \tag{3.13}$$

$$\mathbf{T}_{ji} = \mathbf{T}_{jk} \boxplus \mathbf{T}_{ki}. \tag{3.14}$$

Using the matrix representation of the Eq. 3.11, the composition is directly achieved as a matrix-vector multiplication or matrix-matrix multiplication as

$$\bar{\mathbf{x}}_j = \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i \qquad \rightarrow \qquad \mathbf{x}_j = \mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}, \tag{3.15}$$

$$\mathbf{T}_{ji} = \mathbf{T}_{jk} \cdot \mathbf{T}_{ki}. \tag{3.16}$$

3.2.1 Euler Angles

A 6D pose can be represented using Euler angles and a translation vector. These angles are the minimal representation of a pose orientation with only 3 parameter: yaw α, pitch β and roll γ. The geometrical representation of the angles is shown in Fig. 3.5. Thus, the full pose is defined with only 6 parameters as

$$\boldsymbol{\xi} = \left[\alpha, \beta, \gamma, t_x, t_y, t_z\right]^T \tag{3.17}$$

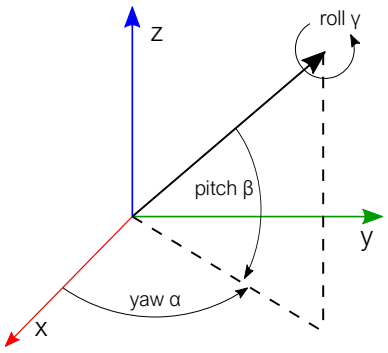


Figure 3.5: A common convention for the Euler angles.

There are many alternatives on how these angles are sequentially concatenated to form the final orientation $\mathbf{R}$. One of the most common options is to first apply yaw (around Z axis), then pitch (around Y axis) and finally roll (around X axis). As a result, the rotation matrix is given by

$$\mathbf{R} = \mathbf{R}_x(\gamma)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\alpha), \tag{3.18}$$

with

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3.20}$$

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{3.21}$$

One of the main drawbacks of the Euler angles is that they suffer from singularities. In certain configurations, e.g. when pitch approaches $\pm 90°$, the representation looses a DoF (Gimbal Lock). This means that for some specific configurations there is not a unique solution of a triplet of angles. This is unsuitable for optimization since we may move along the set of degenerated solutions and get stuck. Thus, the Euler angles representation requires to detect and handle these situations individually. For this reason, this representation has been discarded from this thesis since there are more suitable alternatives described below.

3.2.2 Quaternions

As an alternative, a 6D pose can be expressed with a quaternion and a translation vector. In contrast to Euler angles, quaternions do not suffer from singularities but they are represented with 4 parameters, giving a 6D pose representation with 7 parameters as

$$\boldsymbol{\xi} = \left[q_x, q_y, q_z, q_w, t_x, t_y, t_z\right]^T, \tag{3.22}$$

where the quaternion is always normalized to unit length $q_x^2+q_y^2+q_z^2+q_w^2 = 1$. Therefore, a quaternion has only three independent parameters and it can also be represented with three parameters plus the unit length constraint. The rotation matrix associated to a quaternion is given by

$$\mathbf{R} = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix}. \tag{3.23}$$

The main drawback of quaternions arise during numerical optimizations. In these situations, the representation may drift from the unit length constraint and, consequently, a normalization is required each few iterations (or after a relative big increment is estimated). In this thesis, 6D poses will be internally represented as quaternions since it is a very compact and efficient representations for pose composition (Eq. 3.13 and 3.14). However, other parameterization based on Lie groups (Sec. 3.2.3) will be used to represent rotations during optimizations.

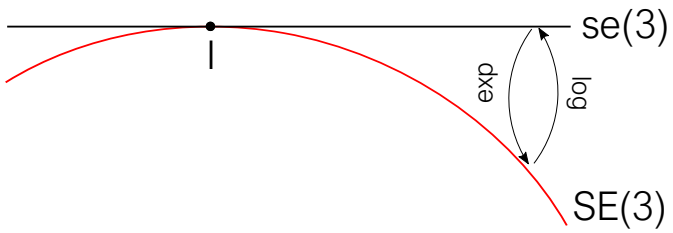


Figure 3.6: Lie group $SE(3)$ and Lie algebra $se(3)$ representation.

3.2.3 Lie Groups and Lie Algebra

As mentioned earlier, a 6D pose belongs to the special Euclidean group SE(3) which is a Lie group. During numerical optimization problems is very common to exploit the mathematical properties of Lie groups to represent 6D poses. This section does not present Lie groups in detail but the required tools to apply their theory to optimization problems. A more in-depth introduction to Lie groups can be found in (Varadarajan, 1984).

A Lie group is a smooth manifold where the product and inverse operations are smooth functions. Every Lie group has an associated Lie algebra, which is the tangent space around the identity element of the group (see Fig. 3.6). Any tangent vector is given by the linear combination of the basis elements of the Lie algebra, which are called generators. In the case of the Lie group SE(3), its corresponding Lie algebra se(3) is given by

$$se(3) = \{\omega_x \mathbf{G}_1 + \omega_y \mathbf{G}_2 + \omega_z \mathbf{G}_3 + t_x \mathbf{G}_4 + t_y \mathbf{G}_5 + t_z \mathbf{G}_6\}, \tag{3.24}$$

where $\boldsymbol{\omega} = \left[\omega_x, \omega_y, \omega_z\right]^T \in \mathbb{R}^3$ represent the rotation coordinates from so(3) and $\mathbf{t} = \left[t_x, t_y, t_z\right]^T \in \mathbb{R}^3$ the translation coordinates, with a total of 6 parameters:

$$\boldsymbol{\xi} = \left[\omega_x, \omega_y, \omega_z, t_x, t_y, t_z\right]^T. \tag{3.25}$$

The generators are expressed as

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In addition, we define the exponential map of a Lie group as the function that converts elements from the algebra to the manifold. Its inverse, the logarithm map, converts elements from the manifold to the algebra (see Fig. 3.6) as

$$exp : se(3) \rightarrow SE(3) \tag{3.26}$$

$$log : SE(3) \rightarrow se(3) \tag{3.27}$$

The closed-form expression for the exponential map is given by

$$\begin{aligned} exp(\boldsymbol{\xi}) &= \begin{bmatrix} e^{\hat{\omega}} & \mathbf{V}\mathbf{t} \\ 0 & 1 \end{bmatrix} \\ e^{\hat{\omega}} &= \mathbf{I}_3 + \frac{\sin\theta}{\theta}\hat{\boldsymbol{\omega}} + \frac{1-\cos\theta}{\theta^2}\hat{\boldsymbol{\omega}}^2 \\ \mathbf{V} &= \mathbf{I}_3 + \frac{1-\cos\theta}{\theta^2}\hat{\boldsymbol{\omega}} + \frac{\theta-\sin\theta}{\theta^3}\hat{\boldsymbol{\omega}}^2 \\ \theta &= |\boldsymbol{\omega}|, \end{aligned} \tag{3.28}$$

where the hat operator $\hat{(\cdot)}$ corresponds to the skew-symmetric matrix of the vector. The closed-form expression for the logarithm map is given by

$$
\begin{aligned}
log(\mathbf{T}) &= \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{t}' \end{bmatrix} \\
\boldsymbol{\omega} &= \ln(\mathbf{R})^{\vee} = \frac{\theta}{2\sin\theta} \cdot (\mathbf{R} - \mathbf{R}^T)^{\vee} \\
\mathbf{t}' &= \mathbf{V}^{-1}\mathbf{t} \\
\mathbf{V}^{-1} &= \mathbf{I}_3 - \frac{1}{2}\hat{\boldsymbol{\omega}} + \frac{1}{\theta^2}\Big(1 - \frac{\theta}{2\tan(\theta/2)}\Big)\hat{\boldsymbol{\omega}}^2 \\
\theta &= \arccos\Big(\frac{tr(\mathbf{R}) - 1}{2}\Big),
\end{aligned}
\tag{3.29}
$$

where $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector of the $SE(3)$ pose, and the vee operator $(\cdot)^{\vee}$ is the inverse of the hat map.

Following the notation of Eq. 3.14 the composition of a rigid transformation $\mathbf{T} \in SE(3)$ with a tangent vector $\boldsymbol{\xi} \in se(3)$ is directly estimated using the exponential map as

$$
\boldsymbol{\xi} \boxplus \mathbf{T} = exp(\boldsymbol{\xi}) \cdot \mathbf{T}. \tag{3.30}
$$

This representation provides the optimal space in which to represent differential quantities on a rigid transformation, such as derivatives or uncertainty. Thus, it is a very appropriate parameterization for numerical optimizations. The most important properties of the tangent space are:

- The tangent space is a vector space with the same dimensions as the number of DoF of the group, thus, providing a minimal representation.
- The exponential map exactly maps elements from the tangent space se(3) to the group SE(3). Thus, we can estimate incremental updates that maintain the final state in the manifold.
- The adjoint linearly and exactly maps elements from one tangent space to another, which simplifies many operations.

In this thesis the Lie algebra is used throughout all the rigid transformation optimizations. The implementation in C++ of all the above functions are freely available in the open-source library Sophus[2], which is used in this thesis.

Adjoint

The adjoint transforms elements from one tangent space to another. The most important property of the adjoint is that the transformation is linear and exact, which guarantees that the tangent space has the same structure at all points on the manifold. Given a transformation $\mathbf{T}$ and a tangent vector $\boldsymbol{\xi}$, the adjoint representation Adj_T is defined as

$$\widehat{Adj_T \cdot \boldsymbol{\xi}} = \mathbf{T} \cdot \hat{\boldsymbol{\xi}} \cdot \mathbf{T}^{-1}. \tag{3.31}$$

Taking the exponential map on both sides of the Eq. 3.31 we get to the expression:

$$\mathbf{T} \cdot exp(\boldsymbol{\xi}) = exp(Adj_T \cdot \boldsymbol{\xi}) \cdot \mathbf{T}, \tag{3.32}$$

which is a direct application of the adjoint that transforms an algebra element from the right tangent space to the left tangent space of $\mathbf{T}$. The adjoint representation of a SE(3) transformation is given by

$$Adj_T = \begin{bmatrix} \mathbf{R} & \hat{\mathbf{t}} \cdot \mathbf{R} \\ 0 & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{3.33}$$

In this thesis the adjoint is used to simplify the jacobian expressions in the proposed visual SLAM approach (App. C), but it has many other uses such as pose uncertainty propagation (Engel et al., 2014).

Jacobians

Another advantage of Lie representation is that jacobian expressions are simplified. Following the Eq. 3.24, the Lie algebra is defined as the coefficients of the linear combination of the group generators. Thus, the jacobian of a Lie grop element with respect to its algebra around zero is directly given by the group generators as

$$\left.\frac{\partial \mathbf{T}}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} = \left.\frac{\partial exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} = [\, \mathbf{G}_1 \mid \ldots \mid \mathbf{G}_6 \,]. \tag{3.34}$$

[2] https://github.com/strasdat/Sophus

where $[\cdot|\ldots|\cdot]$ represents the complete jacobian with all the partial derivatives stacked in matrix form.

This property allows us to easily obtain the partial derivatives of a point transformation with respect to its algebra. This can be done by differentiating the equation $\bar{\mathbf{x}}_j = \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i$ (Eq. 3.13) with respect to a small increment composed to $\mathbf{T}_{ji}$ around zero as

$$\begin{aligned} \left.\frac{\partial(exp(\boldsymbol{\xi}) \cdot \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} &= \left[\ \mathbf{G}_1\bar{\mathbf{x}}_j \mid \ \ldots\ \mid \mathbf{G}_6\bar{\mathbf{x}}_j\ \right] \\ &= \left[\ \mathbf{I}_3 \quad -\hat{\mathbf{x}}_j\right] \qquad \in \mathbb{R}^{3\times 6}. \end{aligned} \tag{3.35}$$

Note that we have used the left-composition convention, but similarly the right-composition could be used using the adjoint of $\mathbf{T}$:

$$\begin{aligned} \left.\frac{\partial(\mathbf{T}_{ji} \cdot exp(\boldsymbol{\xi}) \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} &= \left.\frac{\partial(exp(Adj_{T_{ji}} \cdot \boldsymbol{\xi}) \cdot \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} \\ &= \left.\frac{\partial(exp(\boldsymbol{\xi}') \cdot \bar{\mathbf{x}}_j)}{\partial \boldsymbol{\xi}'}\right|_{\boldsymbol{\xi}'=0} \left.\frac{\partial exp(Adj_{T_{ji}} \cdot \boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} \\ &= \left[\ \mathbf{G}_1\bar{\mathbf{x}}_j \mid \ \ldots\ \mid \mathbf{G}_6\bar{\mathbf{x}}_j\ \right] \cdot Adj_{T_{ji}} \\ &= \left[\ \mathbf{I}_3 \quad -\hat{\mathbf{x}}_j\right] \cdot Adj_{T_{ji}} \qquad \in \mathbb{R}^{3\times 6}. \end{aligned} \tag{3.36}$$

3.3 Image Formation

The sensor of a camera captures the light energy coming from the surfaces of the objects of the 3D world as a 2D discretized image, where each element, or pixel, integrates the amount of energy during the exposure time. This process, also known as image formation, explains the principle from which the irradiance of a 3D surface point is mapped into a 2D image pixel intensity value.

The camera calibration is the pre-processing step by which the internal parameters of a camera that take part in the image formation process are estimated. We can distinguish two parts in the calibration process: (1) the **geometric calibration** which estimates the parameters involved in the transformation of a 3D point into a pixel location in the image plane; (2) the **photometric calibration** which estimates the parameters that describe how the emitted energy by scene surfaces is transformed into pixel intensity values. Fig. 3.7 shows an schematic representation of the different elements involved in the image formation process.

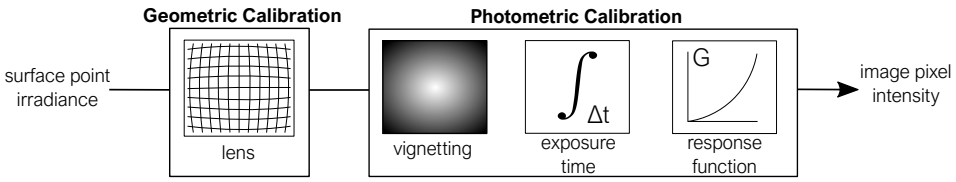


Figure 3.7: Image formation process. Inspired by (Newcombe, 2012)

3.3.1 Geometric Calibration

The geometric calibrations describes the model by which the 3D world is projected onto the image plane. In this work, we will only discuss central camera models, this is, models where all light rays intersect in a unique point, the projection center. The perspective projection is one example of central camera because all light rays intersect in the camera center.

Real cameras require sufficient light to capture the scene information. This is obtained modifying the aperture of the optics, which is the opening area through which the light travels. The non-zero aperture violates the central camera assumption and, as a result, the central model is never completely

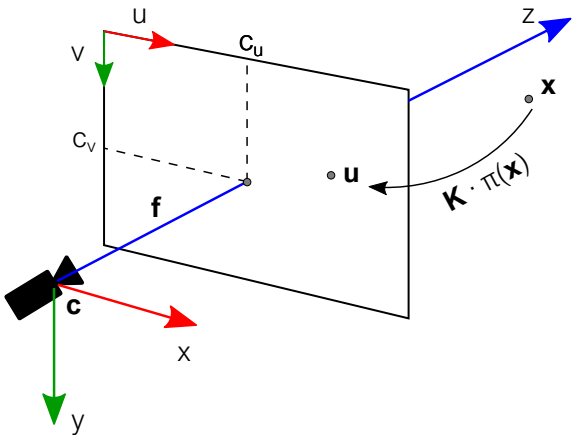


Figure 3.8: Geometric projection involved during the image formation process.

fulfilled in practice. However, the aperture value is usually very small and the central model accurately models real cameras.

Generally, we define $\Theta \subset \mathbb{R}^3$ as the set of 3D points $\mathbf{x}$ with a valid projection on a camera and $\Omega \subset \mathbb{R}^2$ the image domain in pixels. Given a 3D point $\mathbf{x} = (x, y, z)^T \in \Theta$ in a camera coordinates, we define the central projection function into the 2D image plane $\pi(\mathbf{x}) : \Theta \to \mathbb{R}^2$ as

$$\pi(\mathbf{x}) = \begin{bmatrix} m_u \\ m_v \end{bmatrix} = \mathbf{m}, \tag{3.37}$$

where the close-form expression depends on the selected camera model.

Once we project a 3D point onto the image plane, we must transform the resulting coordinates to pixels and their relative position of the image plane to the origin as shown in Fig. 3.8. This can be achieved by the intrinsic calibration matrix $\mathbf{K} : \mathbb{R}^2 \to \Omega$. Using homogeneous coordinates it is given by

$$\bar{\mathbf{u}} = \mathbf{K}\pi(\mathbf{x}) = \mathbf{K}\bar{\mathbf{m}} = \begin{bmatrix} f_u m_u + c_u \\ f_v m_v + c_v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{3.38}$$

with

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.39}$$

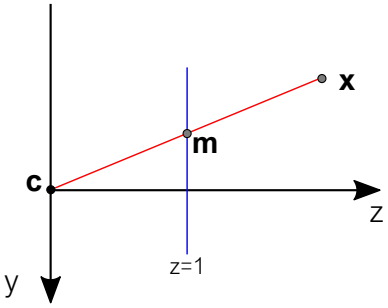


Figure 3.9: Pinhole camera model.

where $\mathbf{f} = [f_u, f_v]^T$ stands for the sensor focal lengths in pixels, c_u, c_v denote the optical center projection in pixels, also known as principal point, and $\mathbf{u} = (u, v)^T \in \Omega$ the projected point in pixels. The focal length transforms point coordinates to pixel units and the principal point changes the origin to the upper left corner.

Note that when projecting a point, we loose the z dimension. Thus, the 3D coordinates of the point cannot be recovered directly. However, we can define the inverse projection function (unprojection) $\pi^{-1}(\mathbf{m}) : \mathbb{R}^2 \rightarrow \mathbb{S}^2$ which defines the ray by which all points are projected to these image coordiantes. The 3D coordinates are back-estimated for points with known depth z as:

$$\mathbf{x} = z \cdot \pi^{-1}(\mathbf{m}) = \frac{1}{\rho} \cdot \pi^{-1}(\mathbf{K}^{-1}\mathbf{u}), \tag{3.40}$$

where $\mathbf{K}^{-1}$ is expressed as

$$\mathbf{K}^{-1} = \begin{bmatrix} 1/f_u & 0 & -c_u/f_u \\ 0 & 1/f_v & -c_v/f_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.41}$$

Pinhole Model

The ideal perspective projection is modeled using the pinhole camera model. Fig. 3.9 shows a geometric representation of the pinhole projection. The projection of a 3D point onto the image plane can be obtained by dividing the point coordinates by their z component as

$$\pi_p(\mathbf{x}) = \frac{1}{z} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3.42}$$

which is defined for $\Theta = \{\mathbf{x} \in \mathbb{R}^3 \mid z > 0\}$. Its inverse is given by

$$\pi_p^{-1}(\mathbf{m}) = \begin{bmatrix} m_u \\ m_v \\ 1 \end{bmatrix}. \tag{3.43}$$

This model assumes a linear projection function between the 3D world and the 2D image plane. This implies that straight lines in the world remain as straight lines in the image. However, real world lenses do not obey the linear model and contain non-linear distortions, specially wide-angle and fisheye lenses. The latter increase the field-of-view by introducing significant distortion to the image, which is easily appreciable in the image by the curvature in the projection of straight lines (see Fig. 3.10).

A frequent approach to solve this is to apply a non-linear function to the pinhole-based projected point $\mathbf{m}$ and then multiply it by the camera intrinsic matrix $\mathbf{K}$. However, implemented in this manner it has a singularity at $z = 0$, which makes it unsuitable for cameras with a field-of-view close or larger than 180°. In these cases, the non-linear distortion should be taken into account inside the projection function, which implies an independent implementation to the pinhole model. This reasoning is appreciable in the wide-angle and fisheye models presented in the App. A.

3.3.2 Image Undistortion

In many computer vision applications it is convenient to remove the geometric distortion from the input image and simplify subsequent operations using a simpler model, such as the pinhole one. This is normally estimated in a pre-processing step using image undistortion techniques, which find the warped image that fits the new projection model. In the rest of this thesis, we will use the pinhole model and, unless stated otherwise, assume all images to be undistorted. Consequently, before we apply any of the proposed approaches, applying the following undistortion technique to the input image will be required.

Given an input image I with its geometric calibration $\mathbf{K}$ and $\pi(\mathbf{x})$, the rectified image I_r is obtained finding for each pixel in the rectified image $\mathbf{u}_r$ the corresponding pixel location in the original one $\mathbf{u}$. This can be efficiently done unprojecting each pixel from I_r with the output pinhole model and projecting onto the input image I as

$$\mathbf{u} = \mathbf{K}\pi(\pi_r^{-1}(\mathbf{K}_r^{-1}\mathbf{u}_r)). \tag{3.44}$$

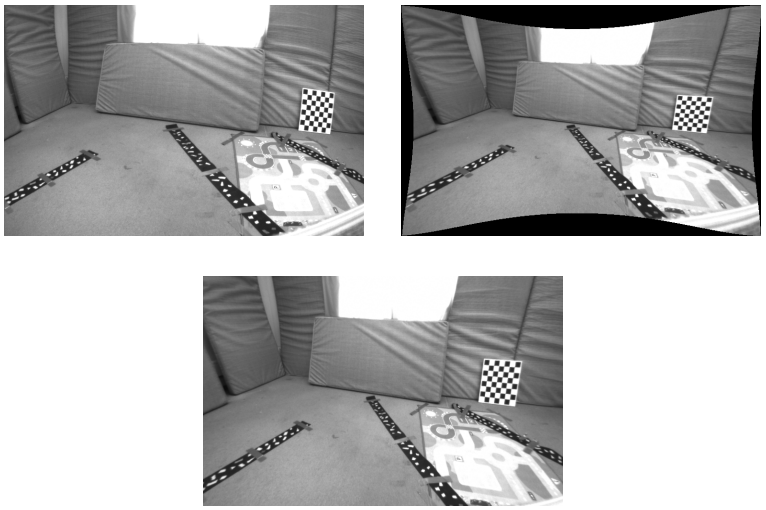


Figure 3.10: Image undistortion. Top left: the original distorted image where straight lines in the world are projected as curves. Top right: the undistorted image will all the pixels from the original one. It contains many invalid pixels shown in black but straight lines in the world remain as straight lines in the image. Bottom: the undistorted image cropped to fit the maximal square area with valid pixels.

Then the rectified image intensity values are estimated using an interpolation function $f(I, \mathbf{u})$ over the input image as follows:

$$I_r[\mathbf{u}_r] = f(I, \mathbf{u}), \tag{3.45}$$

where usually a bilinear interpolation function is used.

The undistorted image may contain pixels that were outside of the boundaries of the original image (see Fig. 3.10) and, thus, the output image has to be cropped. Normally, the output projection function is selected such the maximum valid image area is covered. Note that the output image might contain interpolation artifacts, such as blurring and aliasing. Thus, it is essential to select the most suitable projection and interpolation models for each case. Finally, it is not advisable to apply undistortion techniques to wide-angle lenses since the field-of-view is reduced significantly due to cropping. In theses cases, it is recommended to include the projection model into the mathematical formulation of the algorithm and preserve the advantages of using this kind of lenses.

3.3.3 Photometric Calibration

The photometric calibration describes how the energy emitted by a surface point, also known as irradiance, is mapped into the observed pixel intensity value. The photometric capturing process makes the same surface point be observed with different pixel intensity values, for example, due to the exposure time changes.

Indirect methods normally ignore the photometric model of the sensor since feature detectors are usually invariant to variations in intensity. Consequently, they do not require any photometric correction to associate points from different images. On the contrary, direct approaches make use of the so called photo-consistency assumption, which states that a surface point is observed with the same color from different viewpoints. Thus, they require to transform pixel intensity values to measurements independent of the photometric capturing process.

Assuming a scene composed of Lambertian surfaces with a constant illumination, we can consider the irradiance of each point consistent in time. For this reason, the irradiance can be used as a direct measurement to associate points over different frames. Given a projected surface point $\mathbf{u}$, its irradiance $B(\mathbf{u})$ can be estimated from its pixel intensity value $I[\mathbf{u}]$ as

$$I[\mathbf{u}] = G(tV(\mathbf{u})B(\mathbf{u})), \tag{3.46}$$

where:

- The exposure time t controls the amount of light that reaches the image sensor. For example, when the scene ligh level is low, a longer exposure time is required to guarantee enough information is captured, and vice-versa. In many cameras this value is adaptable or even automatic.
- The lense vignetting V is the brightness attenuation towards the image periphery (Szeliski, 2010).
- The response function G is the non-linear physical response of the sensor that maps irradiance values to intensity ones (Debevec and Malik, 1997). It also models other user defined operations such as gamma correction.

In this thesis, we will only compensate the automatic exposure changes, but the complete photometric model could also be included as a pre-processing step. A more detailed explanation of the photometric calibration can be found in (Engel et al., 2016b).

3.4 Multiple View Geometry

This section presents the intrinsic geometry that is behind multiple camera views. We will start from the simple two-view case and extend the formulation to multiple cameras at the end of this section. Since the following techniques are extensively used in computer vision problems, we will not provide an in-depth explanation of their mathematical background which can be found in (Hartley and Zisserman, 2004).

3.4.1 Epipolar Geometry

The epipolar geometry is the intrinsic projective geometry between two camera views, which can be obtained from a stereo system or from a moving camera. For simplicity, we will consider a monocular camera moving relative to a scene with already calibrated intrinsic parameters.

Given two cameras with a known relative pose $\mathbf{T}_{ji}$, we define the epipolar geometry as the intersection of the image planes with the set of planes with the line that joins the camera centers (baseline) as axis. Consequently, the epipolar geometry only depends on the relative pose (Sec. 3.2) and the projective model (Sec. 3.3) between the two cameras. Fig. 3.11 shows an specific example where a 3D point $\mathbf{x}$ constraints the epipolar plane. In the epipolar geometry we can distinguish the following elements:

- The **epipole**, $\mathbf{e}$, is the point of intersection of the line that joins the camera centers with the image planes. It can also be seen as the projection of one camera center in the image plane of the other camera.
- An **epipolar plane** is a plane from the family of planes containing the baseline. If it is constrained with a 3D point, the plane is unique.
- An **epipolar line** is the line intersection of the epipolar plane with the image planes. All the epipolar lines intersect in one point, the epipole.

It can be seen in Fig. 3.11 that for each pixel location $\mathbf{u}_i$ in one image, there is a corresponding epipolar line in the other image. As a consequence, if we want to obtain the corresponding pixel $\mathbf{u}_j$ we only need to search along the epipolar line rather than the entire image. This is a very useful property of the

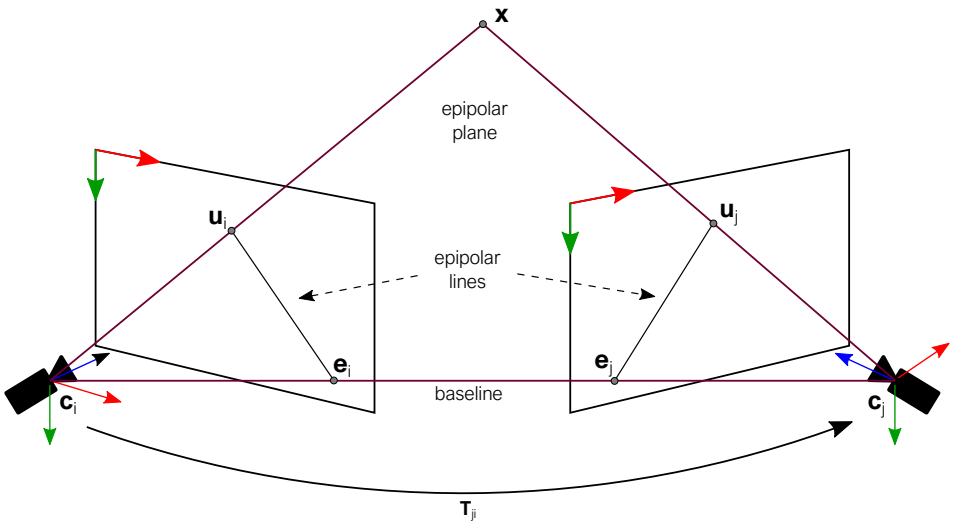


Figure 3.11: Epipolar geometry

epipolar geometry that we will exploit in the proposed visual SLAM approach of Chapter 5 to find pixel point correspondences between images.

The correlation that maps a point in one image to a corresponding epipolar line in the other image is mathematically expressed as the **fundamental matrix** $\mathsf{F} \in \mathbb{R}^{3\times3}$. Given a pair of corresponding image pixels $\mathbf{u}_i$ and $\mathbf{u}_j$, the mapping condition is given by

$$\bar{\mathbf{u}}_i^T \cdot \mathsf{F} \cdot \bar{\mathbf{u}}_j = 0. \tag{3.47}$$

This is the most basic property of the fundamental matrix. Importantly, it states that it is possible to relate pixel points from two camera views without any knowledge of the cameras calibration and position, just the fundamental matrix. As a result, it can be computed up to scale directly from pixel point correspondences (in general at least 7). The details of the fundamental matrix calculus can be found in the chapter 11 of (Hartley and Zisserman, 2004).

Another option is to use the **essential matrix** E when the calibration of the cameras is already known. In this case, the relationship between the fundamental matrix and the essential matrix is given by

$$\mathsf{E} = \mathsf{K}^T \cdot \mathsf{F} \cdot \mathsf{K}. \tag{3.48}$$

Similar to the fundamental matrix, the essential matrix can be estimated from pixel point correspondences (in general at least 5) and the calibration

matrix $\mathbf{K}$. The essential matrix can be estimated up to scale using the five-point algorithm in (Nistér, 2004). Once the essential matrix is known, we can recover the relative pose between the two camera views up to scale and a four-fold ambiguity. However, it is easy to decide the correct solution by selecting the option with the largest number of points in front of both cameras. This test is also known as cheirality test.

The fundamental and essential matrices are very useful when working directly with pixel points. Thus, it is a very efficient way of obtaining the initial relative pose (up to scale) between consecutive images in a monocular systems, where we initially miss the 3D structure. Once we have recovered the 3D motion, we can estimate the 3D structure by triangulation. In this thesis, we will use the essential matrix to initialize the proposed monocular visual SLAM approach of Chapter 5 from an input video sequence.

However, in this thesis we are interested in recovering the 3D structure of the scene. Consequently, we will work with 3D primitives rather than 2D pixel points in the image. The mapping function of a pixel point with known depth z from one image to the other using Cartesian coordinates is given by

$$\mathbf{u}_j = \mathbf{K} \cdot \pi(\mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i) = \mathbf{K} \cdot \pi(\mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}), \tag{3.49}$$

with

$$\mathbf{x}_i = z \cdot \pi^{-1}(\mathbf{K}^{-1}\mathbf{u}_i). \tag{3.50}$$

Apart from the definition given before, the epipolar line can also be understood as the projection of the optical rays of one camera in the image plane of the other camera. As a result, using the Eq. 3.49 it is possible to traverse the epipolar line with a 1-dimensional search varying the values of the point depth z.

3.4.2 Inverse Depth Mapping

Previously, in Sec. 3.1, we have discussed that using Cartesian coordinates complicates the management of points at infinity. In contrast, the inverse depth provides a better representation with continuous values for $z > 0$. Using the inverse depth, the pixel mapping formulation of the Eq. 3.49 is transformed to:

$$\mathbf{u}_j = \mathbf{K} \cdot \pi(\mathbf{R} \cdot \frac{1}{\rho} \cdot \pi^{-1}(\mathbf{K}^{-1}\mathbf{u}_i) + \mathbf{t}). \tag{3.51}$$

However, although $\rho = 1/z$ is continuous for $z > 0$, the Eq. 3.51 contains a discontinuity for $\rho = 0 \rightarrow z = \infty$. Thus, the question is: why are we using the inverse depth? Luckily for us, we can scale the whole problem by ρ and obtain a linear expression in inverse depth as

$$\mathbf{u}_j = \mathbf{K} \cdot \pi(\mathbf{R} \cdot \mathbf{K}^{-1}\mathbf{u}_i + \rho \cdot \mathbf{t}), \tag{3.52}$$

where the final pixel location $\mathbf{u}_j$ is the same as in the original case. As a result, the Eq. 3.52 maps pixels from one camera to another using a linear transformation in inverse depth. This formulation allows to exploit all the benefits of the inverse depth (Sec. 3.1) with a linear observation model which suits better for optimization problems (Sec. 3.5.5). In this thesis, we will extensively use this formulation in the proposed visual SLAM approach of Chapter 5.

3.4.3 Bundle Adjustment

We have already seen that the epipolar geometry from the two-view case is enough to recover the structure and motion. However, the estimation of the 3D geometry is a chicken-and-egg problem where the structure is required to estimate the motion and vice-versa. As a consequence, the two-view solution is conditioned by projective geometry ambiguities which lead to inaccurate results.

The bundle adjustment (BA) provides a more accurate solution by jointly estimating the structure and motion from multiple views. Its name refers to the *bundles* of rays connecting camera centres to 3D features, which are *adjusted* optimally with respect to both the structure and camera pose parameters (Triggs et al., 2000). Fig. 3.12 shows a scheme of a traditional BA problem with three cameras and a single 3D point.

The BA problem is formulated as a non-linear least squares optimization. Starting from and initial solution, which is commonly provided by the two-view epipolar geometry, the BA iteratively refines the structure and motion parameters by minimizing a certain cost function. Overall, the cost function quantifies the error between the observed and predicted feature measurements. We can distinguish two types of BA formulations: (1) the **geometric BA (GBA)** – the most traditional one – that minimizes the reprojection error between the image locations of observed and predicted image points; (2) the **photometric BA (PBA)** that minimizes the intensity error between the image pixels of observed and predicted image points. In the GBA when the predicted location gets closer to the observed one, the error is reduced. However, in the PBA this may not happen

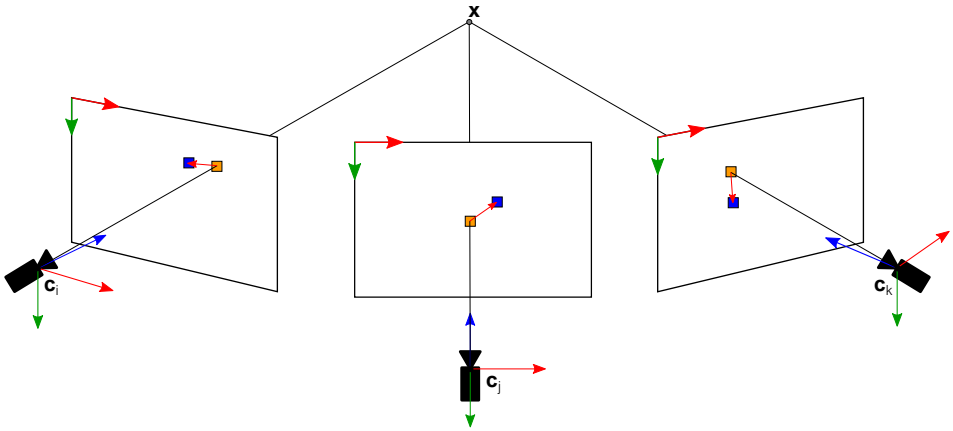


Figure 3.12: Geometric bundle adjustment example with orange and blue squares representing predicted and observed image locations respectively. The minimized reprojection error is represented as a red arrow in the image plane.

due to the fact that images are highly non-linear. As a result, the convergence radius of the PBA is quite small, around 1-2 pixels (Engel et al., 2016a), meaning that the initial parameters should be close to the solution.

In general, given a set of observations obs_i and their corresponding predicted values $pred_i(\boldsymbol{\zeta})$ with respect to motion and structure parameters $\boldsymbol{\zeta}$, the BA objective function is formulated as

$$f(\boldsymbol{\zeta}) = \frac{1}{2}\sum_{i=1}^{n}(obs_i - pred_i(\boldsymbol{\zeta}))^2, \tag{3.53}$$

which can be solved using non-linear optimization techniques such as the ones presented in Sec. 3.5. It is very common to use the Levenberg-Marquardt algorithm presented in Sec. 3.5.3 with a robust influence function described in Sec. 3.5.4 to take into account the presence of outliers.

Furthermore, the observations are usually independent from each other and, thus, point parameters are not correlated. This property provides a sparse structure of the problem, which can be exploited to solve the BA problem faster. Fig. 3.13 shows the sparse structure of the hessian matrix when camera and point parameters are ordered. As can be seen, the point block is diagonal and, hence, easily invertible. The Schur complement (Sec. 3.5.5) takes advantage of this sparsity, also called primary structure, to obtain a reduced problem with only

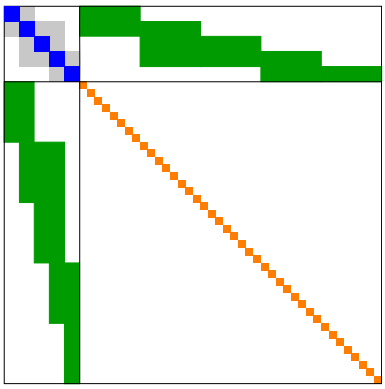


Figure 3.13: Sparsity of the hessian structure in a bundle adjustment problem. Camera pose blocks are represented in blue, point location blocks in orange and camera-point blocks in green. The grey colour indicates the fill-in that occurs when the structure points are eliminated using the Schur complement trick.

the camera parameters. As a consequence, a fill-in occurs in the camera pose block. However, the reduced problem is much faster to solve than the original one, specially with large scale problems with thousands of points.

It is important to note that all the measurements in a BA problem are relative. This means that if we move all cameras and points with the same rigid transformation, we get exactly the same cost function value. Consequently, there are six DoF that we cannot observe (gauge freedom), i.e. the absolute pose. To handle this situation, it is very common to fix the pose of one camera, e.g. the first camera pose. This trick allows us optimize the rest of camera and point parameters while maintaining the absolute pose of the whole problem fixed.

Furthermore, if we are working with a monocular system there is still one DoF that we cannot observe, i.e. the problem scale (absolute distance). This is due to the fact that a camera is an angular measuring sensor that does not measure distances (see Fig. 3.14). One possible solution is to use a stereo system where the baseline between both cameras is known and the absolute distance could be observed. However, similar to the absolute pose, there is still an scale ambiguity when we jointly estimate the motion and structure parameters. A certain BA problem and its corresponding scaled counterpart give exactly the same cost function value. As a result, the solution may drift in the scale direction if we do not fix the gauge freedom. Another common solution, is to fix several

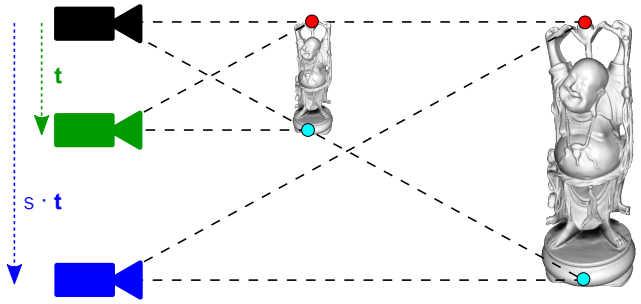


Figure 3.14: A monocular camera measures angles and, thus, it is impossible to distinguish if a camera has moved from black to green position or from black to blue position. Consequently, the absolute scale is unobservable. Inspired by (Strasdat, 2012)

cameras which defines a fixed relative scale during optimization, e.g. the first two camera poses.

In this thesis, we will include a monocular PBA in the proposed visual SLAM approach (Chapter 5) and we will show how to increase its convergence radius with a multiscale formulation. Besides, we will increase the robustness of the optimization to outliers using a probabilistic formulation. Finally, we will apply all the described implementation tricks to efficiently solve the problem equations and remove the gauge freedoms.

3.5 Non-linear Least Squares

In many computer vision and robotics problems the solution is estimated as the minimum of an objective function $f(\boldsymbol{\zeta})$ with respect to some model parameters $\boldsymbol{\zeta}$. In this thesis, we extensively use non-linear least squares optimizations to estimate different model parameters, such as camera poses, formulating both geometric and photometric cost functions. This section presents the optimization fundamentals required to solve non-linear least squares problems independently of the cost function type since the same optimization techniques can be applied in both cases.

In non-linear least squares problems, the objective function is of the form:

$$f(\boldsymbol{\zeta}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\boldsymbol{\zeta}), \tag{3.54}$$

where each $r_i(\boldsymbol{\zeta})$ is a non-linear cost function and $\boldsymbol{\zeta}$ the cost function arguments. By minimizing this function $f(\boldsymbol{\zeta})$, we estimate the values of the parameters $\boldsymbol{\zeta}$ that best fit the model of the data. The above expression can also be expressed in vectorized form as

$$f(\boldsymbol{\zeta}) = \frac{1}{2} \parallel \mathbf{r}(\boldsymbol{\zeta}) \parallel_2^2, \tag{3.55}$$

where $\mathbf{r}(\boldsymbol{\zeta}) = [r_1, \ldots, r_n]$ is the stacked cost vector and $\| \cdot \|_2$ de L2 norm.

In general, we cannot guarantee to find a global minimum, so we are forced to find a local minima in the vicinity of an initial guess $\boldsymbol{\zeta}^{(0)}$. In practice, estimating a closed-form solution to the cost function is usually too complicated and most often impossible, so it is locally approximated using the Taylor expansion around the current model parameters $\boldsymbol{\zeta}$. As a consequence, we are usually not able to estimate the exact minimum and we need to iteratively improve the initial parameters until convergence:

$$\boldsymbol{\zeta}^{(t+1)} = \delta\boldsymbol{\zeta}^{(t)} + \boldsymbol{\zeta}^{(t)}, \tag{3.56}$$

where $\delta\boldsymbol{\zeta}^{(t)}$ is the estimated increment that reduces the cost function.

The rest of this section presents various alternatives to solve the optimization problem, which differ in the way they approximate the cost function (Hartley and Zisserman, 2004). Besides, we also explain several mathematical tricks to improve both the solution quality and the computational performance (Nocedal and Stephe, 2006, Triggs et al., 2000).

3.5.1 Newton's Method

The Newton's method approximates the objective function $f(\boldsymbol{\zeta})$ from the Eq. 3.55 using the second-order Taylor expansion as

$$f(\boldsymbol{\zeta} + \delta\boldsymbol{\zeta}) \approx f(\boldsymbol{\zeta}) + \mathbf{g}^T \delta\boldsymbol{\zeta} + \frac{1}{2}\delta\boldsymbol{\zeta}^T \mathbf{H} \delta\boldsymbol{\zeta}, \tag{3.57}$$

where $\mathbf{g} = \frac{\partial f(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}}$ is the gradient vector and $\mathbf{H} = \frac{\partial^2 f(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}^2}$ is the Hessian matrix. The minimum of the quadratic function is calculated equating the derivatives to zero as

$$\delta\boldsymbol{\zeta} = -\mathbf{H}^{-1}\mathbf{g}, \tag{3.58}$$

where $\delta\boldsymbol{\zeta}$ is the Newton step. The Newton's method is given by iterating the Newton step. Assuming that $\mathbf{H}$ is positive definite, the solution can be estimated using linear algebra, e.g. Cholesky decomposition. The gradient and the Hessian of $f(\boldsymbol{\zeta})$ can be expressed in term of the Jacobian $\mathbf{J}(\boldsymbol{\zeta}) = \frac{\partial \mathbf{r}(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}}$ as

$$\mathbf{g} = \frac{\partial f(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}} = \mathbf{J}(\boldsymbol{\zeta})^T \mathbf{r}(\boldsymbol{\zeta}), \tag{3.59}$$

$$\mathbf{H} = \frac{\partial^2 f(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}^2} = \mathbf{J}(\boldsymbol{\zeta})^T \mathbf{J}(\boldsymbol{\zeta}) + \mathbf{r}(\boldsymbol{\zeta}) \frac{\partial^2 \mathbf{r}(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}^2}. \tag{3.60}$$

If the function is quadratic, Newton's method converges in one iteration. However, Newton's method requires the function to be twice differentiable, which is not always possible. Besides, the derivative expressions of $\mathbf{H}$ could be very complicated for complex cost functions and computationally inefficient. If the residuals are close to linear (i.e. $\frac{\partial^2 \mathbf{r}(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}^2}$ is small) or small (i.e. $\mathbf{r}(\boldsymbol{\zeta})$ is small) the Hessian can be simplified using an approximated value. The following methods exploit this property from the Hessian matrix.

3.5.2 Gauss-Newton Method

The Gauss-Newton method is extensively used in computer vision due to its simplicity. It takes the previous structural properties of the Hessian into consideration and drops the second term. This gives the Gauss-Newton approximation to the least squares Hessian

$$\mathbf{H} \approx \mathbf{J}(\boldsymbol{\zeta})^T \mathbf{J}(\boldsymbol{\zeta}), \tag{3.61}$$

which gives the following linear system, also known as normal equations:

$$\mathbf{J}(\boldsymbol{\zeta})^T \mathbf{J}(\boldsymbol{\zeta}) \delta\boldsymbol{\zeta} = -\mathbf{J}(\boldsymbol{\zeta})^T \mathbf{r}(\boldsymbol{\zeta}). \tag{3.62}$$

This equations are only an approximation of the second-order Taylor expansion and can also be obtained linearizing the residual $\mathbf{r}(\boldsymbol{\zeta})$ instead of $f(\boldsymbol{\zeta})$ (Nocedal and Stephe, 2006):

$$f(\boldsymbol{\zeta} + \delta\boldsymbol{\zeta}) \approx \frac{1}{2} \parallel \mathbf{r}(\boldsymbol{\zeta}) + \mathbf{J}(\boldsymbol{\zeta}) \delta\boldsymbol{\zeta} \parallel_2^2 . \tag{3.63}$$

Solving the normal equations gives the Gauss-Newton step as

$$\delta\boldsymbol{\zeta} = -(\mathbf{J}(\boldsymbol{\zeta})^T \mathbf{J}(\boldsymbol{\zeta}))^{-1} \mathbf{J}(\boldsymbol{\zeta})^T \mathbf{r}(\boldsymbol{\zeta}). \tag{3.64}$$

If the initial residuals are small (i.e. we are close to the solution) and the problem is well parametrized (i.e. locally near linear) the method leads to rapid local convergence. However, when the initial estimate is far from the solution, it may converge to a saddle point rather than a minimum, and for large steps the approximation may be inaccurate. In this cases, it is advisable to use other alternatives that guarantee that the estimated step follows a local descent direction, such as Levenberg-Marquardt, and that the objective cost will certainly decrease.

3.5.3 Levenberg-Marquardt Method

Newton's and Gauss-Newton methods provide good convergence properties when the initial guess $\boldsymbol{\zeta}^{(0)}$ is near the solution. Otherwise, they tend to get stucked in a saddle point or do not even converge. In this cases, it would be useful to have an step control procedure to guarantee that the optimization method chooses a descent direction.

One possible solution is to use the **Gradient Descent** method. It approximates the Hessian with a multiple of the identity matrix, i.e. $\mathbf{H} \approx \alpha \mathbf{I}$ and uses the gradient vector as the most rapid decrease of the cost function as

$$\alpha\delta\boldsymbol{\zeta} = -\mathbf{J}(\boldsymbol{\zeta})^T \mathbf{r}(\boldsymbol{\zeta}) = -\mathbf{g}, \tag{3.65}$$

where α controls the step size. However, the gradient descent method has slow convergence since it tends to zigzag.

The Levenberg-Marquardt method is a combination between a Gauss-Newton and a gradient descent, where the step is adaptively controled to move smoothly between both. For example, the Levenberg-Marquardt step will move to a gradient descent step when the Gauss-Newton step fails. In this case, the normal equations (Eq. 3.62) are replaced by a regularized system given by

$$(\mathbf{J}(\boldsymbol{\zeta})^T\mathbf{J}(\boldsymbol{\zeta}) + \lambda\mathbf{D})\delta\boldsymbol{\zeta} = -\mathbf{J}(\boldsymbol{\zeta})^T\mathbf{r}(\boldsymbol{\zeta}), \tag{3.66}$$

where λ is the dampening parameter that controls the step and $\mathbf{D}$ is some positive definite matrix. A very common option is to choose $\mathbf{D} = \text{diag}(\mathbf{J}(\boldsymbol{\zeta})^T\mathbf{J}(\boldsymbol{\zeta}))$.

The value of λ changes to guarantee that the estimated $\delta\boldsymbol{\zeta}$ gives a cost decrease. When the obtained $\delta\boldsymbol{\zeta}$ leads to an increase of the error, λ is multiplied by a factor and the normal equations (Eq. 3.66) are solved again. Otherwise, λ is divided by a factor and the increment is accepted. Note that when λ is small, the hessian matrix $\mathbf{H} \approx \mathbf{J}(\boldsymbol{\zeta})^T\mathbf{J}(\boldsymbol{\zeta})$ and the computed step will be close to the original Gauss-Newton step. On the other hand, when λ is large, the Hessian matrix $\mathbf{H} \approx \lambda\mathbf{D}$ and the computed step will be close to a gradient descent step. In this way, the Levenberg-Marquardt "interpolates" between a Gauss-Newton iteration, which has a fast convergence in the local vicinity of the solution, and a gradient descent iteration, which guarantees a decrease in the cost function.

3.5.4 Robustified Least Squares

Least squares problems are very sensitive to outliers, i.e. measurements that do not fit the model and have high residual values. These measurements have a high impact in the optimization and can corrupt the system. It is already known that just one outlier measurement can ruin the whole optimization (Leys et al., 2013). One possible solution is to weight each residual differently and reduce the influence of high residual measurements. The robustified least squares function is of the form:

$$f(\boldsymbol{\zeta}) = \frac{1}{2}\sum_{i=1}^{n}(\omega_i \cdot r_i^2(\boldsymbol{\zeta})), \tag{3.67}$$

where ω_i is the weight for each individual residual $r_i(\boldsymbol{\zeta})$. The corresponding system of equations for the robustified Gauss-Newton model is given by

$$(\mathbf{J}(\boldsymbol{\zeta})^T\mathbf{W}\mathbf{J}(\boldsymbol{\zeta}))\delta\boldsymbol{\zeta} = -\mathbf{J}(\boldsymbol{\zeta})^T\mathbf{W}\mathbf{r}(\boldsymbol{\zeta}), \tag{3.68}$$

where $\mathbf{W}$ is a diagonal matrix with the weights ω_i. The Eq. 3.68 is solved equivalently to the normal equations by fixing the weights each iteration.

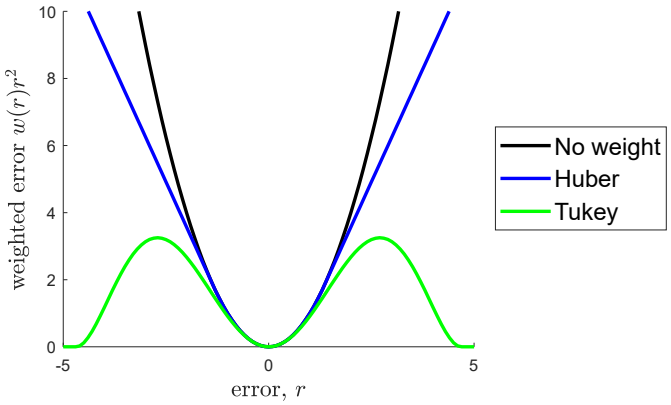


Figure 3.15: Comparison of the weighted errors obtained from two of the most common M-estimators: Tukey ($k = 4.685$) and Huber ($k = 1.345$).

Optimizing the Eq. 3.68 is equivalent to minimizing the negative log-likelihood given independent measurements $\mathbf{r}(\boldsymbol{\zeta})$. If we assume Gaussian distributed measurements, the weights ω_i can be approximated as the square root of the inverse covariance of the measurements. However, if we assume equally Gaussian distributed measurements, we will treat all measurements equally and outliers cannot be neutralized. In this case, it is advisable to assume other probability models for the measurements. In Chapter 5 we explain how to extend the above formulation with a t-distribution model (see App. D) and improve the performance of the proposed visual SLAM approach.

Sometimes, however, it is not possible to know how the measurements are distributed or they are assumed to be Gaussian distributed. In this case, there are different heuristic models, also known as M-estimators, that mitigate the influence of outlier measurements (see Fig. 3.15):

- **Huber**: It is a hybrid between the $L1$ and the $L2$ norm. It gives linear influence to outlier as

$$\omega(r) = \begin{cases} 1 & \text{if } |r| < k \\ \frac{k}{|r|} & \text{otherwise,} \end{cases} \tag{3.69}$$

  where k is usually fixed or dynamically changed with the value $k = 1.345\sigma$ for $\mathcal{N}(0, \sigma^2)$.

- **Tukey**: It is more aggressive than Huber and directly eliminates measurements that exceed a certain residual threshold as

$$\omega(r) = \begin{cases} (1 - \frac{r^2}{k^2})^2 & \text{if } |r| < k \\ 0 & \text{otherwise,} \end{cases} \tag{3.70}$$

  where k is usually fixed or dynamically changed with the value $k = 4.685\sigma$ for $\mathcal{N}(0, \sigma^2)$.

Importantly, any robust influence function $\omega(r)$ has to be carefully implemented. Measurements with high residual are actually the ones with rich information about how should the model parameters $\boldsymbol{\zeta}$ be corrected and they may not be strictly outliers. If we totally remove them, we may drastically slow the convergence or even not get a good solution. One possible implementation is to start iterating with a large outlier threshold and reduce it each iteration. In this way, we will systematically detect outliers measurements that do not fit the model and mitigate their influence.

3.5.5 Implementation Strategies

The following sections describe some implementation strategies to speed up the optimization progress. A more in-depth explanation of the presented strategies together with additional useful approaches can be found in (Nocedal and Stephe, 2006, Triggs et al., 2000).

The Schur Complement

All the optimization methods presented above are suitable for small number of parameters. However, when optimizing a large scale problem, they become extremely expensive. In this cases, solving the normal equations is the computational bottleneck of the optimization. Luckily for us, there are many problems, such as the Bundle Adjustment, where the normal equations have a certain sparse block structure that can be exploited (see Fig. 3.13).

Given an optimization problem with two block of parameters, we can write the normal equations (Eq. 3.62) as

$$\begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\zeta}_1 \\ \delta\boldsymbol{\zeta}_2 \end{bmatrix} = \begin{bmatrix} -\mathbf{g}_1 \\ -\mathbf{g}_2 \end{bmatrix}. \tag{3.71}$$

If each individual cost function $r(\boldsymbol{\zeta})$ depends on any number of parameters block from $\boldsymbol{\zeta}_1$ but just one parameters block from $\boldsymbol{\zeta}_2$, it is easy to see that $\mathbf{H}_{22}$ is a block diagonal matrix and, thus, easily invertible. For instance, in a standard Bundle Adjustment problem, each cost function depends on one or more cameras but just one 3D point. As a consequence, we can use the Schur complement to eliminate the $\delta\boldsymbol{\zeta}_2$ parameter blocks from the optimization and reduce the linear system as

$$(\mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{-1}\mathbf{H}_{21})\delta\boldsymbol{\zeta}_1 = \mathbf{H}_{12}\mathbf{H}_{22}^{-1}\mathbf{g}_2 - \mathbf{g}_1. \tag{3.72}$$

As a result, the system can be solved using a reduced system with just the dimensions of $\boldsymbol{\zeta}_1$. Once again, if we think in a Bundle Adjustment problem, it is very common to have fewer cameras ($\boldsymbol{\zeta}_1$) than points ($\boldsymbol{\zeta}_2$). In this case, the $\dim(\boldsymbol{\zeta}_1) \ll \dim(\boldsymbol{\zeta}_2)$ and solving the reduced system is much faster.

Once the $\delta\boldsymbol{\zeta}_1$ increments are estimated, we can back-substitute the $\delta\boldsymbol{\zeta}_2$ increments as

$$\mathbf{H}_{21}\delta\boldsymbol{\zeta}_1 + \mathbf{H}_{22}\delta\boldsymbol{\zeta}_2 = -\mathbf{g}_2. \tag{3.73}$$

Parameterization

In numerical optimization problems we can distinguish two types of variable parameterizations:

- **Global parameterization**: it is the internally used parameterization to store problem variables. It is preferable to choose a global parameterization without singularities. In this thesis, for example, we will use quaternions to store camera pose orientations and perform pose-point and pose-pose compositions (see Sec. 3.2.2).

- **Local parameterization**: it is the parameterization used during optimizations. It should be locally as nearly linear as possible to guarantee that the cost function is locally nearly quadratic (see Sec. 3.5.2). At the same time, it is desirable to avoid overparameterized representations to remove null directions of the cost function. Similar to the global parameterization, it should not contain singularities. All these properties help to obtain a more stable optimization with faster convergence rates. In this thesis, we will use Lie algebra to represent camera poses during optimizations (see Sec. 3.2.3) because it provides a minimal representation without singularities around zero.

Preconditioning

It is possible to accelerate an optimization problem by transforming the linear system to improve the eigenvalue distribution of $\mathbf{H}$. Sometimes it is advisable to use an ideal Hessian rather than the observed one. This section does not provide a detailed explanation of the mathematical background around preconditioning but presents some useful tools to be applied in real optimization problems. Usually the problem is transformed applying as a linear change in the variables via a non-singular matrix $\mathbf{C}$ as

$$\boldsymbol{\psi} = \mathbf{C} \cdot \boldsymbol{\zeta}. \tag{3.74}$$

Consequently, the transformed linear system is given by

$$(\mathbf{C}^{-T}\mathbf{H}\mathbf{C}^{-1})\delta\boldsymbol{\psi} = -\mathbf{C}^{-T}\mathbf{g}, \tag{3.75}$$

where the convergence rate will depend on the eigenvalues of the matrix $\mathbf{C}^{-T}\mathbf{H}\mathbf{C}^{-1}$ rather than those of $\mathbf{H}$.

The most simple form of preconditioning is a **variable scaling**. This is useful when the objective function is highly sensitive to small changes in a certain direction and relatively insensitive in another direction. Another possible situation arises when the problem variables have different magnitudes. In this case, the matrix $\mathbf{C}$ is a diagonal matrix in which the coefficients are selected to equal the magnitudes of the variables. For instance, in this thesis we need to estimate rigid transformations. In this case, it is important to scale the transformation parameters to take into account the impact of the differences between a unit change in rotation and translation parameters.

Another simple form of preconditioning is the **Jacobi preconditioner**. Similar to the variable scaling, the Jacobi preconditioner uses a diagonal matrix using the values of the hessian as $\mathbf{C} = \text{diag}(\mathbf{H})^{1/2}$. In this case, the preconditioner matrix sets the diagonal of the transformed hessian to one. As a result, it is a very efficient solution for diagonal dominant hessian matrices, such as the ones in a bundle adjustment problem.

3.5.6 Generalized Optimization on Manifolds

Gauss-Newton, Gradient Descent and Levenberg-Marquardt methods have been explained in the context of Euclidean spaces. However, as presented in Sec. 3.5.5,

the usage of an appropriate local parameterization during incremental estimation is very important. Consequently, the local parameterization may not belong to the Euclidean space. This section generalizes the above approaches to work on non-Euclidean manifolds, such as Lie groups (see Sec. 3.2.3).

When optimizing in the Euclidean space, we want to solve the equation:

$$\left.\frac{\partial f(\boldsymbol{\zeta} + \delta\boldsymbol{\zeta})}{\partial \delta\boldsymbol{\zeta}}\right|_{\delta\boldsymbol{\zeta}=0} = 0, \tag{3.76}$$

which is iteratively solved by updating the current parameters will small increments, as proposed in the Eq. 3.56, using one of the above optimization approaches.

Note that the composition of the current state vector with the estimated increment is performed using the vector addition. In order to generalize the above expressions, we will use the $\boxplus$ operator defined in Sec. 3.2, which describes the composition in a general form. As a result, the general expression for a optimization is given by

$$\left.\frac{\partial f(\delta\boldsymbol{\zeta} \boxplus \boldsymbol{\zeta})}{\partial \delta\boldsymbol{\zeta}}\right|_{\delta\boldsymbol{\zeta}=0} = 0, \tag{3.77}$$

with a general iterative update rule according to:

$$\boldsymbol{\zeta}^{(t+1)} = \delta\boldsymbol{\zeta}^{(t)} \boxplus \boldsymbol{\zeta}^{(t)}. \tag{3.78}$$

In the case of an optimization in the Euclidean space, it will be directly the vector addition as defined in Eq. 3.56. In contrast, if the optimization is performed on a non-Euclidean manifold, the composition must be defined. For instance, the iterative update rule for a $SE(3)$ pose optimization is performed using the matrix multiplication together with the exponential map as define in Eq. 3.30. Finally, mention that we have used the left-composition convention during this section. However, the right-composition convention could also be used, as they are linearly related by the Adjoint (see Sec. 3.2.3).

Part III

Proposal

Chapter 4

Object Recognition

The problem of object recognition aims to obtain the position and orientation of 3D world elements using a single image. This is a very challenging task in the case of industrial environments since these are characterized by cluttered scenes, uncontrolled illumination changes, non-Lambertian untextured surfaces and dynamic scenes (see Chapter 2). The popularity of object recognition technologies is growing in the industry in conjunction with AR. Once a 3D object has been recognized, its pose is used to initialize a tracking system that allows superimposing virtual annotations properly aligned with the object. This allows a technician to visualize any relevant information related with a task associated to its context. For instance, Platonov et al. (2007) presents an AR system used to maintain and repair a combustion engine. Hanson et al. (2017) presents a system for guiding workers in a kit preparation tasks, while Makris et al. (2016) presents an AR based interaction system for collaborative robotics.

As discussed in Chapter 2 there are still many open challenges to integrate current state-of-the-art approaches into real industrial applications. Nowadays, marker based solutions are still used in advance manufacturing due to their reliability and accuracy in untextured environments. Ragni et al. (2018) propose an augmented reality tool to guide an operator during the alignment of the raw material with respect to the machine reference. Mendikute et al. (2017) propose an efficient and accurate portable solution to measure and align large raw parts before machining. These approaches require to adapt the environment and, thus, the technician intervention. While this may be a minor issue in some applications, it limits the widely integration of AR into real industrial applications. On the other hand, learning-based methods have shown impressive results for some datasets created in laboratory conditions. However, they still fail to obtain the 6D pose under uncontrolled industrial conditions, where the training settings are

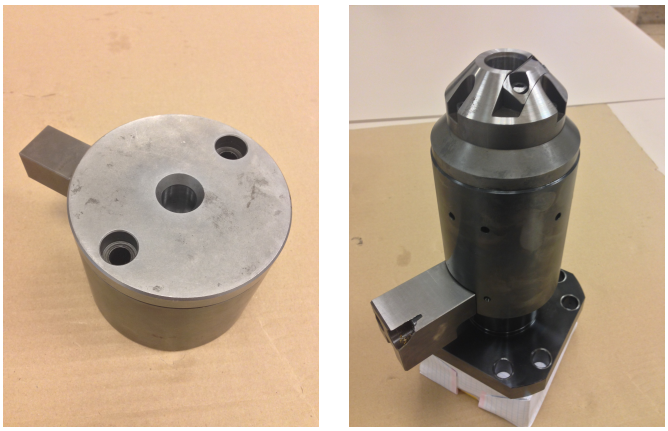


Figure 4.1: Example of some components of a real broaching machine. Note how they contain many holes and cylinders that are easily detectable in the image. The objects have been provided by Ekin S. Coop.

usually far from the real operation ones. Besides, they usually require training stages of many hour, days or even weeks with arduous capturing processes, which is unsuitable for real advanced manufacturing processes.

One of the main goals of this thesis is the development of a robust object recognition approach to handle real operations in real industrial conditions. More precisely, we aim to develop an AR tracking framework for guiding in maintenance for advanced manufacturing. This is a very specific practical application and, thus, we will focus our research on this particular topic. For all the previously mentioned reasons, we have centred the attention of our research on traditional geometry-based solutions that have proven to be robust in industrial environments (Álvarez et al., 2011). In particular, we have observed that many of the components in a manufacturing machine contain revolution elements, such as holes or cylinders (see Fig. 4.1), which we will exploit in the proposed solution. They are easily identifiable in this kind of environments and are stable under changes of illumination. At the same time, we have created our own dataset with real 3D industrial objects under many different configurations – instead of a third-party dataset – to obtain laboratory simulations close to real operating conditions.

This chapter addresses the challenge of object perception in industrial environments and the integration of AR technologies into a real industrial

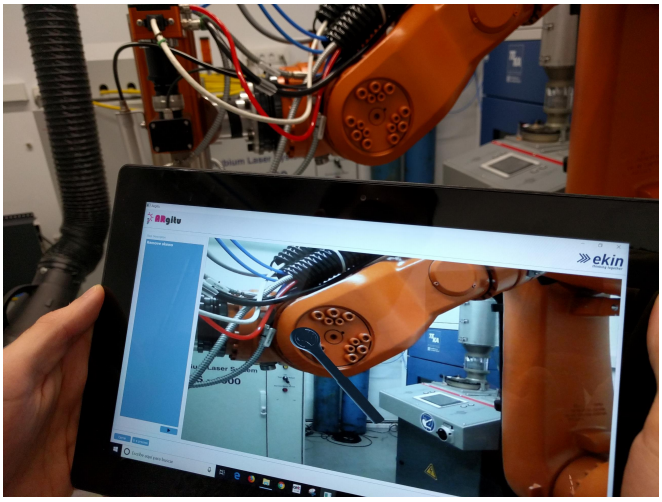


Figure 4.2: ARgitu application applied to the maintenance of a robot arm. The proposed AR pipeline estimates the 6-DOF of rigid parts of a robot arm. This allows superimposing augmented information like an animation of a ratchet attached to this body.

maintenance guidance tool as shown in Fig. 4.2. We present a novel model-based 3D object recognition method which combines model surface conics and edge templates. This helps reducing the image search space, increasing localization robustness and saving computational time. Additionally, we integrate the proposed approach into a full AR pipeline including a feasible 3D object training, recognition and tracking tasks. Finally, we present a full industrial framework for guidance in maintenance, called ARgitu, which includes a general easy-to-use authoring tool for the development of new contents and adapt AR technology applications into the advanced manufacturing industry.

The chapter is organized as follows. We overview the related literature in Section 4.1. We present the novel AR pipeline (Section 4.2) based on the proposed 3D object recognition in Section 4.2.2. The full AR application framework is described in Section 4.3. We report the experimental results of the proposed approach and compare it with the state-of-the-art in Section 4.4.

4.1 Related Work

AR systems are increasingly becoming an important tool to provide new services to companies in advanced manufacturing. The adaptation of AR technologies to industrial environments is currently, however, an open challenge. The quality of the user experience is directly related with the stability of the annotations anchored in the image. There has been an extensive progress in the development of new vision based recognition methods in the last years. This section presents a brief study of current recognition methods which are focused on detecting non-lambertian objects due to their applicability in industrial settings.

Line segments are attractive for object recognition since many man-made objects contain them (Akinlar and Topal, 2011, Brown et al., 2015, Rafael et al., 2012). In Bay et al. (2005), the authors use a color based histogram for line matching, and geometric restrictions to reduce outliers and add new matches. Wang et al. (2009a) and Zhang and Koch (2013) develop a SIFT-like line descriptor (Wang et al., 2009b) dividing the line segments in subregions and extracting gradient information from each one. Using a similar technique, Liu and Marlet (2012) and Zhang and Koch (2012) introduce geometric consistency in a graph matching-based scheme. Zhao et al. (2016) present a new geometric descriptor to match images captured under different image conditions. In Lu Wang et al. (2009) and Tombari et al. (2013), the geometric configuration of line segment groups is used instead. They both develop a semi-local descriptor using geometric information between edgelets and gradients. Later, Verhagen et al. (2014) introduce a scale invariant approach for improving the descriptiveness of line segments. Moreover, Damen et al. (2012) present a method for learning and detecting rigid non-lambertian 3D objects by representing each object as a clusters of edge segments. However, they show that the method lacks precision. In Micusik and Wildenauer (2015a), line segments are used for indoor localization by comparing 3D model lines (Micusik and Wildenauer, 2015b, Zhang and Koch, 2014) with image lines using the chamfer distance. Using any of these line matching approaches, the pose of the camera can be easily recovered (Ababsa and Mallem, 2008, Zhang et al., 2013, 2012). The main weakness of line-based methods is that the repeatability of line segment detection is low and the performance decreases in cluttered scenes. In addition, using only geometric information makes it difficult to match and thus, texture data is required.

Many authors try to overcome these limitations using more robust geometric features such as conics. They are reliable features against illumination changes

and they can provide rich information about the object location. Besides, many man made objects include revolution elements such as circles and cylindrical holes that are projected into the image as ellipses. These are usually detected in the image as contiguous set of edge segments that are joined together to fit circles and ellipses (Fitzgibbon et al., 1999, Rosin, 1998). Ellifit (Prasad et al., 2013), EDCirles (Akinlar and Topal, 2013) and ELSD (Patraucean et al., 2012) are some of the most popular detectors. Ellis et al. (1992) propose a method for ellipse matching considering that they are projected circles, viewed obliquely. They use this information to constraint the position, viewpoint and scale of the model. In (Ayad et al., 2010), three coplanar ellipses are used to track a C-arm using homographies, while Usabiaga et al. (2009) work with multiple cameras for hand pose estimation using two coplanar ellipses. Alternatively, since at least two ellipses are needed to recover the location of the camera, some works have developed new methods to fuse other features with ellipses. Wang et al. (2008) developed a method for estimating the pose from a single view of one circle with two orthogonal lines. In Costa and Shapiro (2000), the authors recover the camera location by estimating an initial location with ellipse and then, obtain a unique solution with points.

Shape based methods have shown good performance. Their main drawback is the large search space, as long as, they do not perform any previous matching scheme. The registration is executed by brute force, which considerably slows down their performance. Dominant Orientation Templates (DOT) (Hinterstoisser et al., 2010) is a popular method. It measures the similarity between two images by comparing the orientation of the gradients. The method was extended to handle depth sensors in (Hinterstoisser et al., 2012a) and color information in (Peng, 2015).

Chamfer Matching is another popular method. It evaluates the similarity between two image edges. It can efficiently be computed using image Distance Transform (DT) but it cannot handle cluttered backgrounds. For this reason, Liu et al. (2010) propose a new variation that takes into account edge directions, increasing its robustness. In Choi and Christensen (2012), the Fast Directional Chamfer Matching (FDCM) approach is used to initialize a particle filter tracking. More recently, some authors introduce the FDCM technique to bin-picking applications (Imperoli and Pretto, 2015, Liu et al., 2012). Liu et al. (2012) uses a multi-flash camera to extract robust image edges reducing the influence of reflections.

All these methods perform well on their specific systems but they are unsuitable for industrial environments. Most of them reduce significantly their robustness under uncontrolled scenarios, such as illumination changes, cluttered backgrounds or occlusions. Moreover, the majority of them require hard manual training phases which makes them unsuitable for a direct industry application.

In this chapter, we propose a new method to improve the registration phase of chamfer matching approaches using conic priors, which are obtained matching model surface circles with image ellipses. In that way, we combine the robustness of ellipse detection with the model shape, represented by the chamfer distance. Our method is able to recognize objects containing circles, such as holes, in industrial environments improving the performance of the current state-of-the-art methods as much in accuracy as in computational cost.

4.2 Model based AR Pipeline

This section presents the complete AR pipeline based on the proposed recognition method presented in Section 4.2.2. The workflow is separated in two main parts:

1. An offline training phase where CAD model geometry features are extracted (Section 4.2.1). This step is executed only once per model and it is fully automatic. Therefore, it can be directly applied to the industry without requiring the participation of any technician.

2. An online phase where the 3D pose of the object is detected and it is tracked in real-time. First, the object is recognized and located using the proposed approach (Section 4.2.2) and, then, the position is tracked using the information from previous frames (Section 4.2.3). The same CAD model geometric features are used for recognition and tracking.

Tracking allows the operator to move the camera and the object freely in space while virtual annotations remain anchored to the object. Whenever the tracking is lost, the recognition module is activated and the position of the object is relocalized. Finally, using the relative position from the online phase virtual guiding annotations are superimposed onto the image. An overview of the AR pipeline is shown in Fig. 4.3.

4.2.1 Offline training

CAD models contain rich geometric information that is used during the recognition and tracking modules. Before starting up the online AR application, we need to extract and store all the required information. This section presents the offline phase where visual geometric features and edge templates are computed. Fig. 4.4 shows the main ideas and overview of the CAD model training.

Given the 3D CAD model of an object and its 3D triangle mesh, we extract high curvature edges measuring the angle between neighbor faces, and considering as straight edges the face boundaries with angles larger than a predefined threshold. We have found that curved surfaces usually contain triangles whose relative angle is smaller than 40 degrees. We also detect 3D surface circles. If at least n sharp straight edges turn to the same direction smoothly, we fit them to a circle using a least-squares error minimization. If

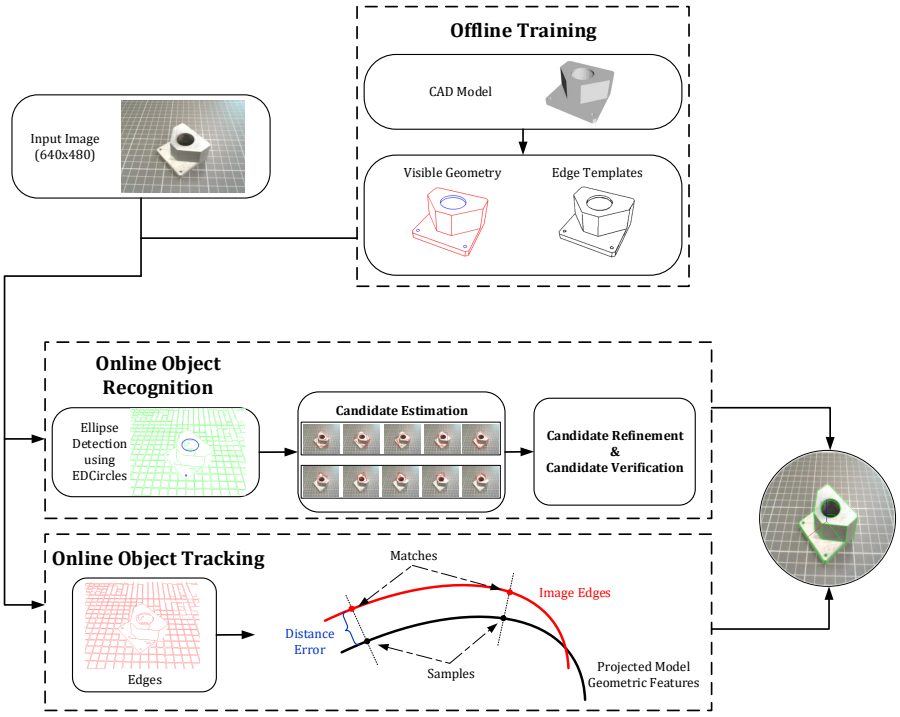


Figure 4.3: Overview of the AR pipeline.

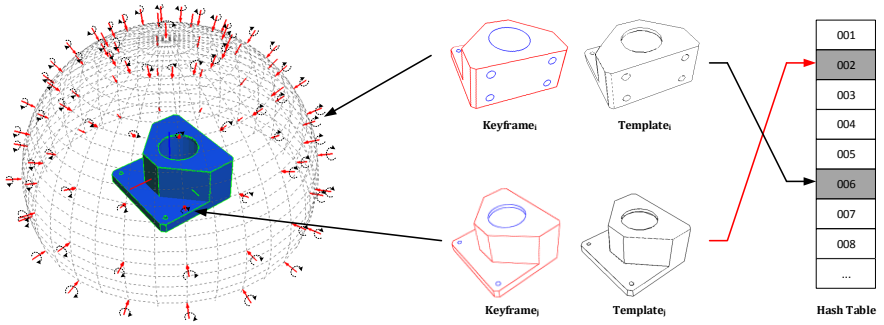


Figure 4.4: Training of the CAD model. A set of virtual cameras is generated on a sphere. For each camera, the visible geometry and template edges are extracted. Finally, this information is indexed in a hash table using the camera location information

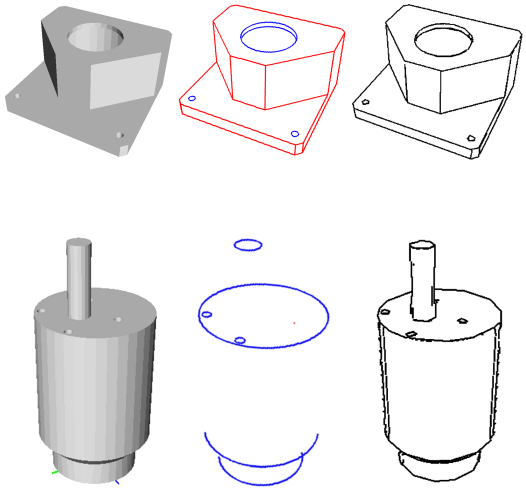


Figure 4.5: Training example for a possible camera location. The left column shows the CAD model with its coordinate system. The middle column shows the visible straight edges (red) and circles arcs (blue). Finally, the last column shows the model edge template. Whereas geometric features provide only real edges, templates allow us to work with virtual edges which are generated by colour contrast with the background and do not belong to the model geometry itself.

the fit error stays under a threshold, those segments are replaced by the circle equation. We found that considering at least 10 straight edges achieves a good performance in circle detection. In this way, we obtain visible 3D straight edges and circles from the CAD model. However, this method identifies both surface and interior features (i.e. hidden segments not actually visible from the exterior). To discard interior features, we render all features and detect the interior ones using occlusion queries based on z-buffer tests. Some results are shown in Fig. 4.5.

We generate a set of virtual cameras around a sphere with the z-axis pointing towards the model. For each camera, we detect the visible geometry features for the current pose applying occlusion queries based on the z-buffer. Using the same buffer, we extract Laplacian edges to create the model templates. The Douglas-Peucker algorithm (Douglas and Peucker, 2011) allows to represent the edge images as a collection of line segments, transforming edge images to line-based representation chains which permits applying integral image

optimization techniques. Moreover, the algorithm allows us to simplify the edge map filtering noise. In addition, it reduces the database memory requirements, since only line end-points are stored. We also compute the 2D direction of the edge segments and store the camera locations.

After processing all cameras, we group all the visible edges, circle-arcs and templates in a one dimensional hash table using the information of the pose. We use the Euler angles of the rotation matrix as the key of the hash table to index them. In this way, we acquire visible geometry features and templates for each position in an efficient manner. In addition, we also add the possibility of rotating the camera z-axis (pointing towards the model) to create a more realistic simulation (See Fig. 4.4). At the end of this algorithm, our database contains the following information:

- Indexed visible 3D model geometry (edge lines and circle arcs) for each camera,
- Indexed linear representation of 2D model templates for each camera.

4.2.2 Online Object Recognition

The first step during the online phase of the pipeline is the localization of the target objects in space without any previous information of its pose. Our method combines chamfer matching techniques (Liu et al., 2010) with conics based pose estimation (De Ma, 1993) to improve upon the weaknesses of each other.

Template matching methods use a brute force approach during the search step which normally consists in a sliding window. Thus, it is computationally costly to find possible candidates in the scene. Besides, working with complex shapes demands thousand of templates, which increases memory and computing requirements. In addition, brute force matching techniques cause false positives even with sophisticated optimizations. This is why they are used under controlled illumination conditions, with accurate depth edges and planar objects such in (Liu et al., 2012).

All these reasons makes them unsuitable for real-time applications in arbitrary environments. In order to sidestep these difficulties, we introduce a new matching scheme using corresponding conics between model surface circles and image ellipses. This allows reducing the search space and the number of outliers, specially in the case of cluttered backgrounds. At the same time, template based

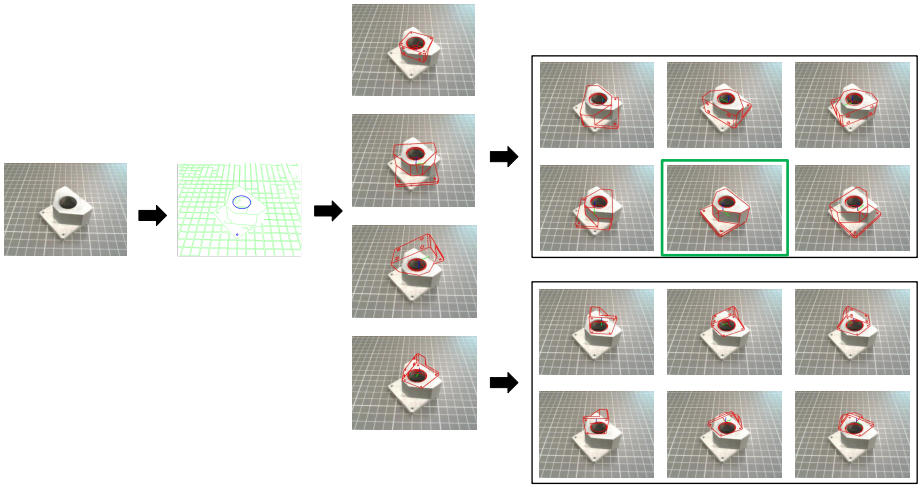


Figure 4.6: Object recognition pipeline. From left to right: the inpute camera image; ellipses (blue) and edges (green) detected in the image; conic-based pose hypothesis computation; chamfer distance minimization and best candidate estimation by refinement and verification (represented by the green box).

optimization allows us to find a unique solution using a single circle. In this case, templates allow us to solve the camera location using the whole model shape, whereas points and lines contain only partial information. Besides, we include a refinement and verification steps which improve the accuracy and robustness of the detection against challenging environments with cluttered backgrounds.

Candidates estimation based on model surface conics and edge templates

First of all, we obtain image ellipses using EDCircles (Akinlar and Topal, 2013) which is one of the most efficient and robust approaches to detect circles and ellipses. It searches for edge segments in the input image and joins them to form circular arcs using heuristics. The ellipses are finally verified using an *a contrario* validation step due to the Helmholtz principle, rejecting false positives. Our method only requires ellipse equations, so it can also work with partial elliptical arcs. In addition, we store the edges computed by EDCircles since they are required in further steps of the algorithm.

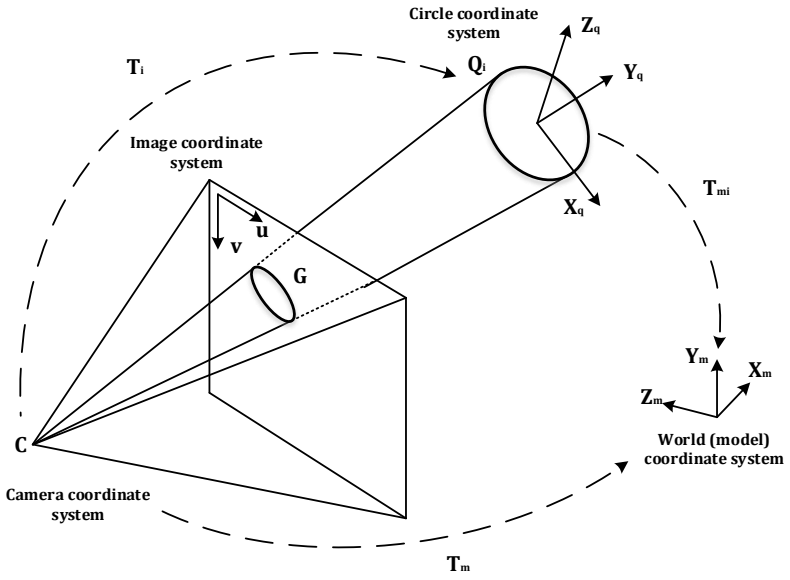


Figure 4.7: Rigid-body transformations applied during the method.

Since we cannot guarantee which ellipse belongs to the model surface, we compute a pose hypothesis $\mathbf{T}_i = [\mathbf{R}_i \mid \mathbf{t}_i]$ for each ellipse-to-circle pair $(\mathbf{G}, \mathbf{Q}_i)$ as proposed in (De Ma, 1993) and described in App. B. As they prove, it is only possible to recover the circle normal axis and center point using a single ellipse, since the in-plane axes (x and y) are not uniquely determined. Thus, we need an additional step. We propose to rotate the object about the normal axis of the 3D circle (z axis). This new step guarantees that the non-symmetrical characteristics of the object are taken into account to obtain a unique solution (see Fig. 4.6).

In the case of solids of revolution, $\mathbf{T}_i$ corresponds to a valid solution, since there are infinitely many solutions. Therefore, it is not possible to identify a unique pose and we select the initial estimation as the solution.

There are infinitely many rotations around the 3D circle normal axis that provide the exact same 2D projection of the circle. We propose to discretize the possible solutions in a finite number of rotations $\mathbf{R}_k = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_n\}$,

which are member of SO(3). For each pose hypothesis, we obtain n additional hypothesis by right multiplication as:

$$\begin{aligned} \mathbf{R}_j &= \mathbf{R}_i \cdot \mathbf{R}_k, \\ \mathbf{t}_j &= \mathbf{t}_i. \end{aligned} \tag{4.1}$$

This camera locations correspond to local 3D circle coordinate systems. Before evaluating the hypothesis with templates, we transform it to the model coordinate system $\mathbf{T}_m = [\mathbf{R}_m \mid \mathbf{t}_m]$ (see Fig. 4.7). We assume that the 3D rigid-body transformations $\mathbf{T}_{mi}$ from 3D circles to model coordinate system are already known from the training phase. The camera location relative to the model is estimated as $\mathbf{T}_m = \mathbf{T}_{mi} \cdot \mathbf{T}_j$:

$$\begin{aligned} \mathbf{R}_m &= \mathbf{R}_{mi} \cdot \mathbf{R}_j, \\ \mathbf{t}_m &= \mathbf{R}_{mi} \cdot \mathbf{t}_j + \mathbf{t}_{mi}. \end{aligned} \tag{4.2}$$

To evaluate each possible solution, edge templates provide a good measurement of the similarity between image edges and model shape, since they contain rich information of the model geometry for each camera view. We propose to minimize the chamfer distance between the closest template and image edges:

$$\arg\min_{U_m} d_{CM}(U_m, V) \qquad \text{with} \quad U_m \in U, \tag{4.3}$$

with

$$d_{CM}(U_m, V) = \frac{1}{n} \sum_{\mathbf{u} \in U_m} DT_V(\mathbf{u}), \tag{4.4}$$

where U_m is the closest model template in the space to the current location hypothesis $\mathbf{T}_m$ and DT_V is the Distance Transform of the input image edges V. Since templates are indexed in a hash table using Euler angles, we only evaluate the spatially closest template. In that way, we can efficiently access each template and the method will only evaluate locations that have been trained skipping the rest. This allows to significantly speed up the computation while increasing its robustness, since we are able to skip known unreal positions. This is good in practice since parts in an assembly are usually in the same configuration. For instance, the cutting head in a horizontal lathe is always in the same direction and, thus, vertical arrangements can be left behind.

Before evaluating each selected template, we need to transform template edges to the image plane correctly. This is, it is necessary to estimate the scale

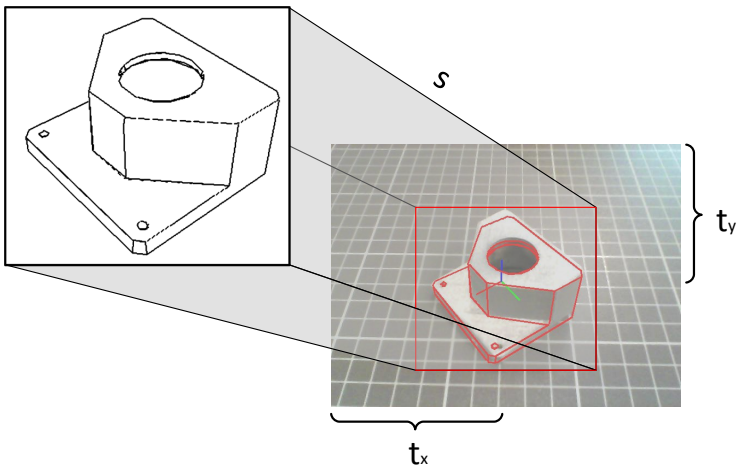


Figure 4.8: Affine transform between trained pose, $\mathbf{T}_t$ (left) and hypothesis pose, $\mathbf{T}_m$ (right). (t_x, t_y) correspond to the image plane translation and s to the scale difference.

and translation between the scene location $\mathbf{T}_m$ and trained location $\mathbf{T}_t$. To solve this, we propose to work with predefined 3D model points transformed with $\mathbf{T}_m$ and estimate their affine transformation (see Fig. 4.8).

During the chamfer matching evaluation (Eq. 4.4), we order lines by their length and start the summation from the largest one. During the summation, if a template score exceeds a threshold, it is automatically discarded, in order to reduce further computations. Finally, candidates with an evaluated cost below the threshold are inserted in a list with a limited number of best candidates.

Candidate Refinement

The previous step extracts a list of the best hypothesis of the object locations in the image. Since discretizing the rotation around the 3D circle normal axis generates small errors, a refinement step is essential to obtain a more accurate solution. In addition, the accuracy of the candidates is not high enough to pass a robust verification technique. Thus, each candidate must be refined with an efficient optimization approach.

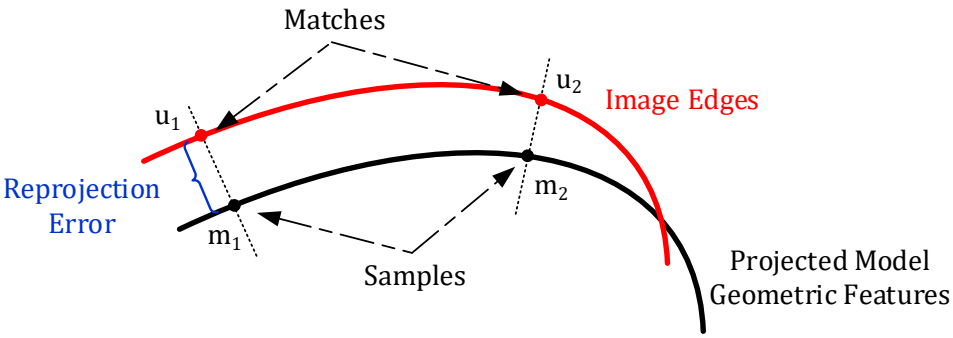


Figure 4.9: Candidate refinement minimizing the reprojection error between model samples and image edge matches.

We propose a method similar to the one proposed by Drummond and Cipolla (2002) where model features are matched to image edges minimizing the reprojection error. We project the visible edges and circle arcs of the model into the image using the estimated pose and select a set of 2D samples $\mathbf{m}$. For each sample $\mathbf{m}_i$, a local search is performed along the edge or arc normal direction. Then, if an image edge pixel $\mathbf{u}_i$ is detected in the vicinity of the sample, a positive correspondence is set. The current object position T is estimated by minimizing the distance between samples and image edge matches given by

$$\begin{aligned} \mathsf{T}^* &= \underset{\mathsf{T}}{\operatorname{argmin}} \sum_{i}^{n} \| \mathbf{m}_i - \mathbf{u}_i \|_2 \\ &= \underset{\mathsf{T}}{\operatorname{argmin}} \sum_{i}^{n} \| \mathsf{K} \cdot \pi(\mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}) - \mathbf{u}_i \|_2, \end{aligned} \tag{4.5}$$

where $\mathbf{x}_i$ represents a sample 3D point from the model visible edges and circle arcs. These are selected to achieve an evenly distributed point-cloud.

Fig. 4.9 shows an schematic representation of the matching step and the reprojection error formulation. In addition, we use multiple matches per sample to deal with cluttered backgrounds. Finally, the object pose is optimized using an iterative Levenberg-Marquardt algorithm, and all those multiple correspondences are managed with a Tukey loss function in order to remove the influence of outliers (see Sec. 3.5).

Figure 4.10: Recognition example of a real industrial object (see Fig. 4.1). Note how the projection of the model edges (green) perfectly aligns with the object in the image.

Candidate Verification

Some of the candidates may represent false positive object locations. For this reason, we need a verification phase to distinguish outliers and choose the best candidate solution. We propose a verification step that detects situations where the candidate is superimposed onto a cluster of edges of other objects. This can result in a low distance error even if those edges do not actually form part of the target object. The approach is based on the similarity between image gradient directions and model geometry feature normals.

Let $\mathbf{m}_i$ be a model point projected on the image plane I, the similarity is computed measuring the difference between the image gradient direction θ_I and the projected model feature normal θ_m. The points $\mathbf{m}_i$ are discrete points selected from visible model geometry features for the current candidate location. In the case of circular arcs, the 2D projection normal is computed from its corresponding

elliptical arc in the image. Thus, we define a similarity metric as:

$$\tau = \frac{1}{n} \sum_{i=1}^{n} |cos(\theta_l - \theta_m)|. \tag{4.6}$$

Ideally, a perfect match has the same image gradient directions as model feature normals which gives a similarity of one. We have seen in our experiments that to obtain a robust performance the score threshold must vary depending on the object shape. Working with complex objects demands less restrictive values. However, in general a threshold of 0.8 works relatively well with almost all tested models. In applications with heavily occluded objects, the value should be lowered. Finally, the verified candidate with the lowest reprojection error is taken as the right object location (see Fig. 4.10).

4.2.3 Online Object Tracking

Once the object localization is estimated, we need to follow the object position in time. To achieve this, we use the same technique as in the refinement step in Section 4.2.2. However, in this case, we use the pose of the object from the previous frame to start iterating. In this way, we refine the position of the object from frame-to-frame assuming that the relative motion between consecutive frames is low. This assumption is usually fulfilled in practice. Anyway, if the technician performs a fast camera movement and the tracking is lost, the 3D object recognition module is activated to recover the object location.

Another possible tracking solution could be to use the object localization as an initialization of the proposed visual SLAM approach of Chapter 5. Apart from obtaining a consistent 3D reconstruction it also works as a robust tracking method. However, this would only be possible for static scenes since the proposed visual SLAM approach does not handle dynamic environments.

4.3 ARgitu: Augmented Reality Guidance in Industry

The method is integrated in a complete framework, called ARgitu, that enables the creation of interactive guides for the assistance in maintenance of complex machinery. The framework is composed of a set of software libraries and two main application: an author tool that creates a database with information of the steps to perform, including several kinds of multimedia information (text, video, virtual reality animations and augmented reality) and a guiding tool that presents this information using a mobile device (see Fig. 4.2).

Both tools have been designed to provide an easy to use interface based on a simple navigation paradigm. The user can access the process information of a machine and navigate through each step of a maintenance operation. Depending on the type of the step, the user is presented with an interface that enables the creation of new contents (for the author tool) or its visualization (using the guiding tool). Currently, the contents are text (HTML including text and images), video (MP4 files), virtual and augmented reality visualizations.

Virtual reality is used to present the user with 3D animations of the activities to be performed. Depending on the situation, the user can navigate freely in the scene or the point of view is fixed in a region of interest. There, the user is presented with one or several animations that display the steps to accomplish a task including the tools required. Fig. 4.11 shows the creation of a virtual animation to assist during the maintenance of a robot arm.

Augmented reality is used to present the user with information in its actual context. ARgitu enables several approaches to AR, including using markers and images as anchors. However the most powerful approach is the use of a real component of a machine as target. In this case, the method described in Section 4.2 is used for the detection and tracking of the object. The author tool contains a simple way to import a geometrical element and use it as anchor for AR. Once the anchor's geometry is defined the user can use it as reference frame to add elements and create animations of components related to the target such as the tool required to complete a step. The author tool completes automatically the training phase of the detection and tracking algorithm. The information is stored with other assets of the application. When a user uses the guiding application all the relevant information is loaded from disk and the detection algorithm will search for the location of the object of interest. Once found, the system triggers the animation system that present the geometry and animations associated to that task with their position and orientation correctly aligned with the detected

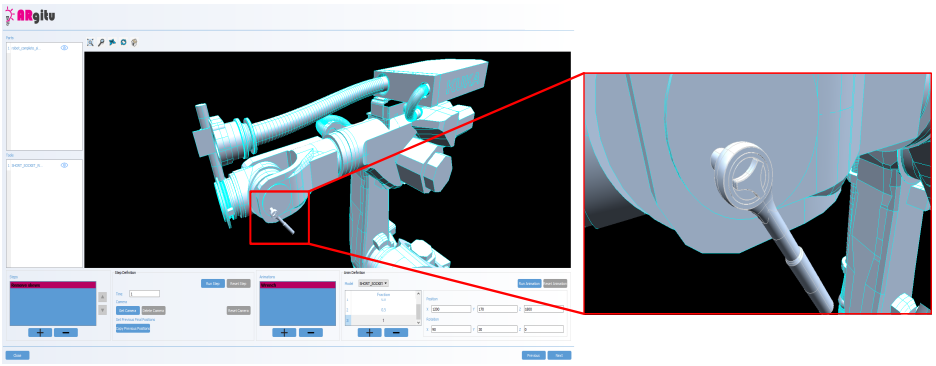


Figure 4.11: ARgitu: using the author application a wrench is virtually animated to assist during a maintenace task of a robot arm. The same virtual annotation is used in the AR example task of Fig. 4.2.

position of the target part (see Fig. 4.12). Fig. 4.13 shows an example of the experiments carried out to test the proposed tracking framework in real industrial conditions.

The virtual animations for tasks presented using virtual or augmented reality are created using the same interface in the author tool. The only difference is related to the location of the virtual elements that in the case of augmented reality are located in a position relative to the target object. Thus, the application enables presenting context-aware information about the task, such as the actions to be performed with a tool.

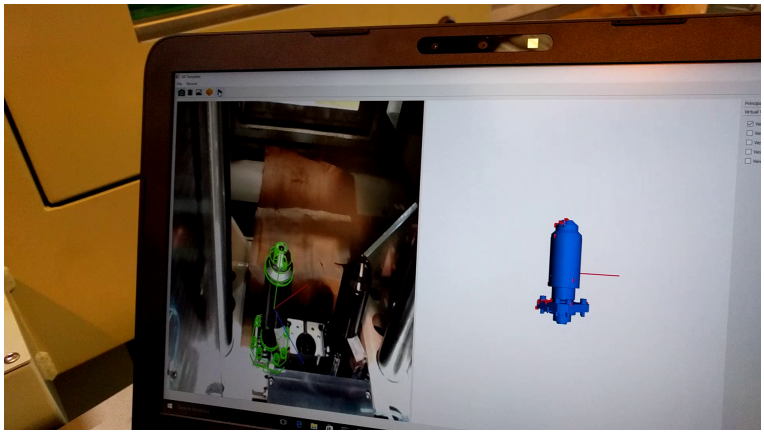
The creation of VR and AR based guides requires importing the model geometry. For example, as described above, the proposed object detection algorithm requires a surface representation of the target object. The framework supports using a number of 3D object formats. One of the formats supported is STEP (ISO 10303), this enables an extremely simple way to import data from CAD systems, since most of them are capable of exporting data in this format. Thus, it facilitates the use of the tool in small and medium size companies by removing the need of modeling the geometry externally.

(a)

(b)

Figure 4.12: Augmented reality taks: (a) the guiding application shows virtual annotations anchored to an object to assist the technician during a maintenance task. (b) the tracking allows moving the object while guiding annotations remain attached to the real object.

(a)



(b)

Figure 4.13: Evaluation of the proposed tracking framework in real industrial conditions: (a) experimental setup with a consumer laptop and a webcam. (b) close view of the tracking result where the model visible geometry (blue) is projected into the image (green).

4.4 Experiments And Discussion

As stated above, one of the most critical components of an AR pipeline is the object detection algorithm. Once the target component is detected several approaches can be used to track the camera in the scene. For this reason, this section presents a validation of our object detection approach. We first estimate the most appropriate system parameters, and then, we compare our approach to the most recent chamfer matching method, called Fast Directional Chamfer Matching (Liu et al., 2010), and to the two-dimensional version of the Iterative Closest Point (ICP-2D) (Besl and McKay, 1992), which is one of the most popular methods for registering a set of points.

Our dataset is composed of 240 640 $\times$ 480 images that contain different objects[1] placed in arbitrary 3D positions and configurations. These configurations include random camera locations showing a single object, grouped with different objects, cluttered by a regular grid and occluded by other objects. Each image is provided with the ground truth position of the objects, obtained manually beforehand. Fig. 4.14 shows several examples of the recognition systems under different configurations.

All experiments were executed using a standard PC with an Intel Core i7-860 CPU at 2.8GHz and 6 GB of RAM. All compared algorithms have been implemented by ourselves using C++ and OpenCV 3 without any strong optimization. When possible, they share the same codebase and the same parameters, enabling objective performance and timing comparisons.

4.4.1 Parameter Study

We evaluate the object detection rate varying several parameters. We use all images in the database to obtain the most robust parametrization for a general case. The parameters tested are the number of rotations around the normal, the chamfer distance threshold and the number of maximum candidates. During the experiments, we modify one of these parameters and fix the remaining two, with a value of 36 rotations around the axis (10 degrees), a maximum number of 20 best candidates and a chamfer distance threshold of 20 pixels. Fig. 4.15 presents the results for both the detection rate and the computational time.

[1] Some of the objects used in the experiments can be found in the RoCKIn@Work competitions, `http://rockinrobotchallenge.eu`

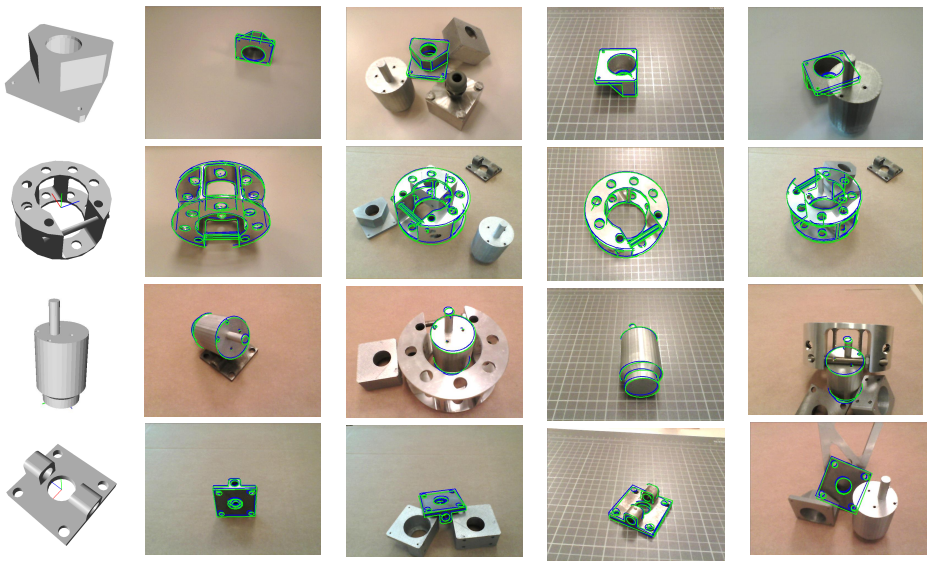


Figure 4.14: Detection of different objects over diverse configurations. (1-row Bearing Box, 2-row Horsehoe, 3-row Motor, 4-row Plate) The detections are shown by superposing the projection of the visible 3D model geometry for each camera location in green. The ground truth is presented in blue.

Evaluation of the number of rotations: We first evaluate the detection rate of each object respect to the number of rotations performed around the circle normal axis (see Section 4.2.2). For a low number of rotations, the method is unstable and achieves low detection rates. This is expected, since non-symmetric characteristics of the object are not taken into account. Indeed, the Motor model (see Fig. 4.14) is almost not affected since it has a revolution symmetry. As expected, the total execution time increases linearly with the number of rotations. Establishing the number of rotations to 36 is a good trade-off between speed and detection robustness.

Evaluation of the chamfer distance threshold: As expected, the detection rate for low thresholds (0-5 pixels) is close to zero. Increasing the threshold allows evaluating more candidates and the detection rate grows until it stabilizes for a value of around 10 pixels. Above this value, the computation cost remains constant. We fix the threshold to 10 pixels since it is the value that stabilizes the detection rate performance.

Evaluation of the number of maximum candidates: The detection rate increases with the number of candidates but stabilizes around a value of 20. This allows to evaluate more location hypothesis and, thus, estimate better solutions. However, the main drawback is the increment of the computation time, that increases linearly with the number of candidates. We conclude that setting the maximum number of candidates to 20 achieves a good balance.

Overall we also observe that the computational time increases with the number of circles in the model. The Horseshoe is by far the most computationally demanding object due to its large amount of holes and circular arcs in its surface.

4.4.2 Evaluation

With the parameters presented in the previous section, we compare our approach to Fast Directional Chamfer Matching (FDCM) (Liu et al., 2010), and to a two-dimensional version of Iterative Closest Point (ICP-2D) (Besl and McKay, 1992). We choose FDCM as the reference for chamfer matching techniques, which are mainly used for bin-picking vision based systems. Besides, we find ICP-2D as the state-of-the-art registering approach. FDCM estimates the object location minimizing both the edge proximity and their orientation similarity. ICP-2D aims to minimize iteratively the distance between a set of points by a nearest neighbor search.

For FDCM, we use our own implementation with the parameters proposed by the authors. Similar to our method, it is used as a candidate estimation technique in our recognition system. Since we do not have any prior information about the model location, we train the model for all possible camera locations and set the scale search from 0.6 to 1 times the template size with a step of 0.1. The detection system is based on scanning the image using a sliding window.

For ICP-2D, we also use our own implementation. In this case, the ICP-2D technique is used as a variant to our method. Instead of using edge templates for evaluating location hypothesis, we use the ICP-2D algorithm for registering them. Both the image point samples and the model samples are selected to achieve a homogeneous point-cloud. The image samples are computed from image edges and model samples from the projection of visible 3D model geometry. Since ICP-2D corresponds to an iterative minimization algorithm, we skip the refinement step.

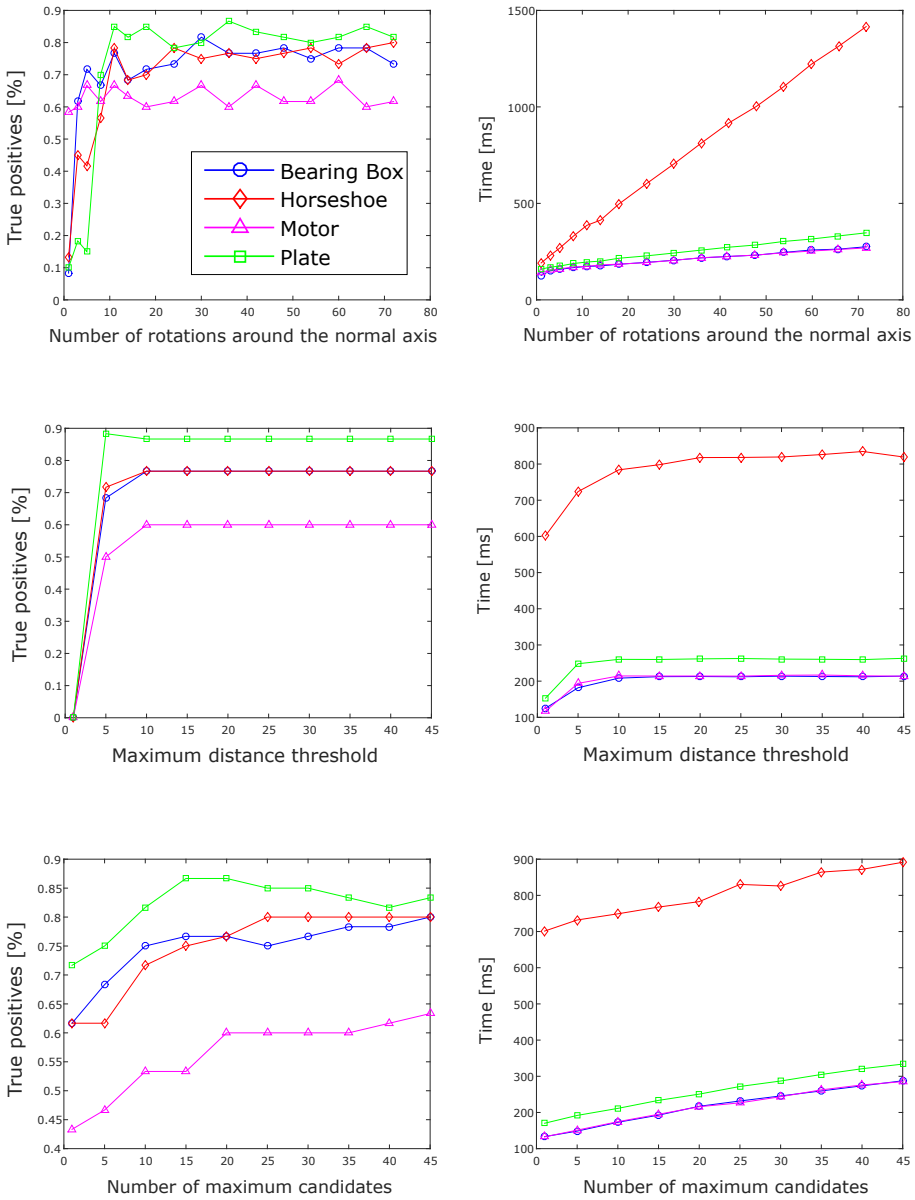


Figure 4.15: Detection rate (left) and computational time (right) for different parameter combinations. First row when changing the number of rotations around the normal axis, second row when changing the maximum chamfer distance threshold, and third row when changing the number of maximum candidates.

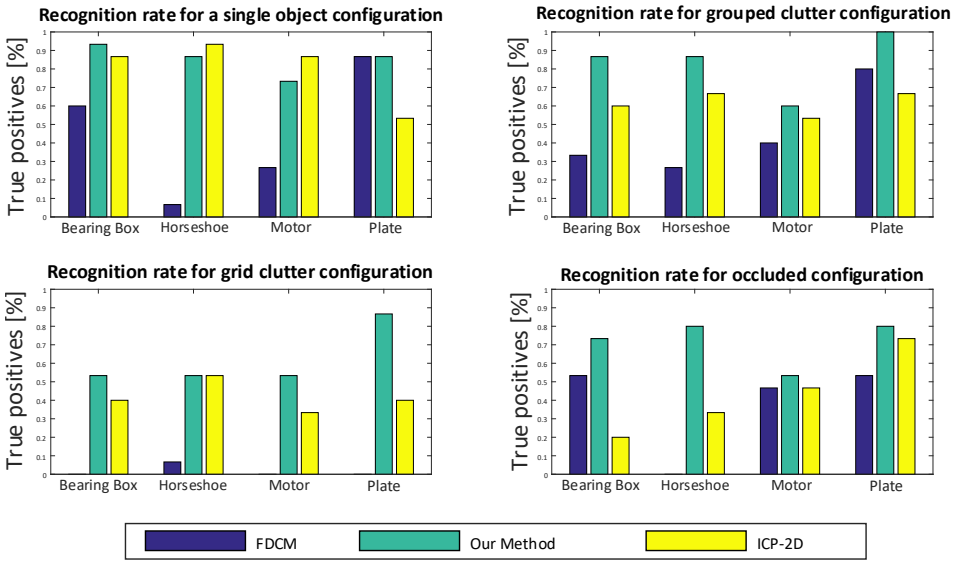


Figure 4.16: Recognition rate comparison for each object. The results are shown in four different configurations: a single object, group clutter, grid clutter and occlusions.

For this experiment, we separate the database in four different categories. Each category represents particular model configurations. In addition, we analyse each model separately to distinguish the method performance in each case. The four categories correspond to the configurations of: a single model, grouped objects, models with a regular grid background and occluded objects (see Fig. 4.14). Finally, we evaluate the average computation time of each stage in the recognition system.

Detection rate. The detection rate scores of the different methods is presented in Fig. 4.16. For a single object configuration, our method achieves better performance than FDCM for all models. Only ICP-2D behaves better with cylindrical models. In the case of FDCM, the detection rate is drastically lowered with very reflective objects, such as the Horseshoe. Only in the case of the Plate FDCM achieves the same recognition rate.

When the models are grouped with other objects our method achieves higher recognition rates, even if, in general, the scores are lower (see second column in Fig. 4.16). However, in the case of FDCM the number of true positives increases

for the Horseshoe and the Motor models. The ICP-2D technique starts to fail since the number of points in the neighbor increases, causing the failure of the optimization.

The robustness of our method to cluttered backgrounds is due to ellipses. The use of image ellipses allows the method to skip many wrong camera locations by searching locally around them. This can be observed with the regular grid clutter (see third column in Fig. 4.16). It is an extreme configuration where the background is filled with small squares and man-made object projections fit well in many configuration due to their nature. In this case, only our conic based approach is able to estimate the camera location. For all the objects, our method performs better or similar to the ICP-2D variation. This is the expected behavior since FDCM takes only edges into account without any robust feature classification.

To evaluate our method against occlusions, we have tested overlapping the target objects with different degrees of occlusion (see fourth column in Fig. 4.16). As expected, the number of correct detections decreases with the occlusion level. However, our method outperforms FDCM. For more complex objects, such as the Horseshoe, FDCM fails, whereas our method achieves good results. This is an advantage since man-made systems are usually assembled from many components, creating severe occlusions. For the ICP-2D-based variant, the decrease is more pronounced due to the increment in the number of neighbor points, but working with conics still allows to estimate the camera location in many cases.

After the detection rate validation, it is clear that the usage of conics together with edge based approaches outperforms other alternatives. We manage to obtain higher detection performance for all tested environments. The improvement is noticeably larger with challenging situations such as occlusions and cluttered backgrounds where the robustness of conics allows discarding many false positives that edge based approaches would take into account.

Processing time. Although our method and the ICP-2D variant work similarly in terms of robustness in some cases, our method outperforms ICP-2D in terms of speed by several orders of magnitude. The mean computation times for all the methods are shown in Table 4.1.

Our approach is about four times faster than FDCM. Fig. 4.17 shows a break down of the execution time for both methods. Our method clearly outperforms

Table 4.1: Mean computation time for all the dataset. It includes the ellipse detection, integral images computation, candidate search, refinement and verification steps.

Algorithm	FDCM	Our Method	ICP-2D
Time (ms)	1642	405	31432

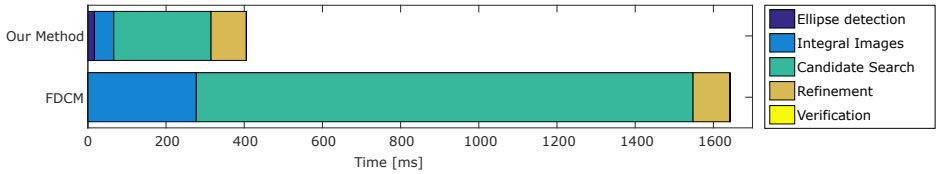


Figure 4.17: Comparison of the mean computation time for all the dataset between FDCM and Our Method. Our approach performs four times faster and the searching is speeded-up by a factor of five.

the search step in term of speed. The usage of conics allows us to discretize the image plane and, thus, significantly reduce the computational time in contrast to a brute force strategy using a sliding window. We reduce the searching by about five times under unconstrained conditions. In addition, our method uses a simpler chamfer distance technique, which uses only the location of edges, and allows executing integral images faster. Moreover, our method introduces the ellipse detection computation time but it is insignificant compared with the other steps. Both methods use similar time for the refinement and verification steps, since we set the same number of maximum candidates.

4.4.3 Examples

We present in Fig. 4.14 some examples of the response of the proposed recognition method using the dataset models. All of them are untextured models with different shape and materials. It shows the robustness of the proposed solutions against challenging situations, such as cluttered backgrounds and partial occlusions. Note how our method is able to handle reflections of mechanical pieces satisfactorily.

In addition, we present in Fig. 4.18 and 4.19 some examples of the proposed method working in a mobile device under real industrial conditions.

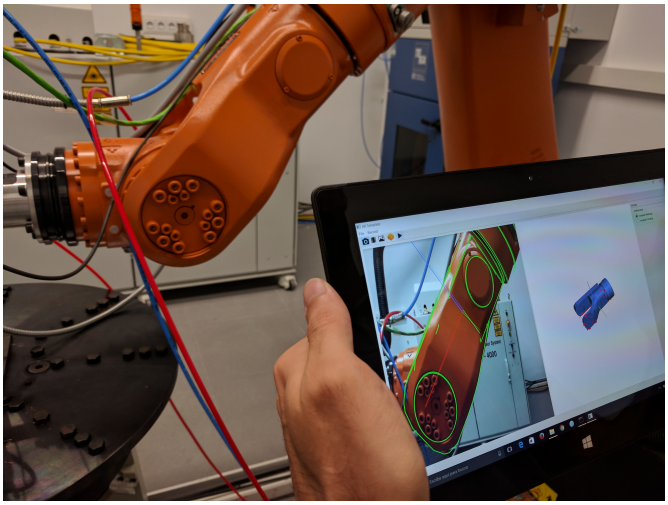


Figure 4.18: Recognition example of an industrial robot arm with a mobile device. The model visible geometry is projected into the image in green.

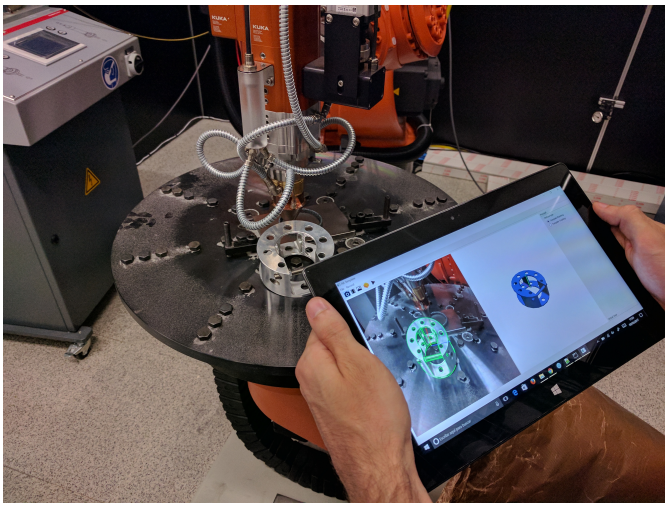


Figure 4.19: Recognition example of the Horseshoe model in an industrial environment with a mobile device. The model visible geometry is projected into the image in green.

4.4.4 Discussion

The proposed method requires the model to contain circles in its surface and those to be visible in the image. Although the usage of conics is one of its main strengths, it is also one of its main limitations. The lack of ellipses in the image makes the proposed solution fail and this occurs due to two main reasons: when the model position does not allow perceiving ellipses in the image or when the ellipse detector fails. Thus, the proposed method relies on the ellipse detector robustness which is usually degraded with severe occlusions. In this cases, only small elliptical arcs are visible. Nonetheless, it does not require texture information and can handle untextured models. Additionally, this thesis is oriented to industrial environments where the presence of untextured objects with revolution elements, such as holes or cylinders, is quite common.

The following chapter presents a robust visual SLAM approach that does not rely on feature detectors but on raw image intensity values. As a result, it can work in low textured environments without requiring specific features. Although it has been particularly developed for autonomous robot navigation, it could also be used as a tracking approach for AR applications in the industry due to its robustness in those conditions.

Chapter 5

Direct Sparse Mapping

Recovering the 3D geometry – camera location and scene structure – from a moving camera is still an open challenge in computer vision and robotics. In general, we can identify two different situations during robot navigation: the *exploration* that refers to the phase where the robot is traversing through an unknown environment and the *revisiting* where the robot is returning to an already known location. Ideally, the goal is to obtain a new map during the exploration that is perfectly consistent with the old one when revisiting. However, drift during exploration is inevitable and, thus, one should obtain the most accurate estimates to keep the drift as small as possible during exploration. At the same time, it is very important to reuse existing map information to correct the drift and maintain a consistent map during revisits. Otherwise, the system will be prone to duplicate map points and generate long-term motion drift and structure inconsistencies. Normally, the exploration task is tackled as a visual odometry problem and the revisiting task as a mapping problem (see Chapter 2).

During revisiting situations, we can additionally distinguish two cases. *Small scale* cases where the accumulated drift is small enough to be detected and corrected using the formulation of the model itself (i.e. reprojection error or photometric error). For example, this case is very usual in indoor applications, such as the cleaning robot that traverses the same rooms and corridors repeatedly. *Large scale* cases where the robot has traveled long enough distances that the accumulated drift cannot be handled with the model. In this cases, it is very common to use an appearance-based place recognition module which identifies if the current scene is similar to an already visited place. This situations are more likely to happen in long term trajectories such as self-driving cars. The latter is also known as the loop closure problem (Strasdat, 2012). This chapter tackles the problem of visual mapping with a direct formulation in

small scale cases. As we will validate in the experiments, a direct formulation is sufficient to handle many situations without requiring a place recognition module.

Direct approaches have proven to be an effective method for estimating scene geometry and camera motion in visual odometry (VO). Photometric bundle adjustment (PBA) minimizes the photometric error of map point observations over a subset of selected frames, known as keyframes. Normally, the number of keyframes in the PBA is limited to avoid large computations. We call active keyframes and active points to those keyframes and map points selected to be optimized in the PBA. A very common strategy to select active keyframes is to use a local sliding-window of most recent keyframes. Points are sampled across image pixels with locally high gradient module, such as edges and weak intensity variations. They are associated to only one keyframe where they are initialized. In the rest of keyframes, there is not an explicit and fixed data association, because the PBA recomputes the correspondences as a part of the optimization. Thus, direct methods do not rely on the repeatability of selected points and are able to operate in scenes with low texture but with contours.

Current PBA based methods are only able to do VO, which builds a temporary map to precisely estimate the camera pose. They use a sliding-window that selects close in time active keyframes, marginalizing map points that leave the field of view. The marginalization strategy reduces the computation complexity by removing old cameras and points while maintaining the system consistent to unobservable degrees of freedom, i.e. absolute pose and scale. As a consequence, if the camera revisits already mapped areas, the PBA cannot reuse marginalized map points and it is forced to duplicate them. This is a severe limitation: the system cannot benefit from the highly informative reobservations of map points, and this causes motion drift and structure inconsistencies.

In contrast, VSLAM methods build a persistent map of the scene, and continuously process map point reobservations. Instead of using a sliding-window and marginalization, they retain keyframes and map points with a fixed location in the model and select the active keyframes and map points according to covisibility criteria, i.e. they observe several map points in common. This results in a network of keyframes where the connectivity is based on whether they observe the same scene region, even if they are far in time. The fixation strategy maintains the system consistent to unobservable degrees of freedom and it enables the reuse of map points. Thus, VSLAM approaches

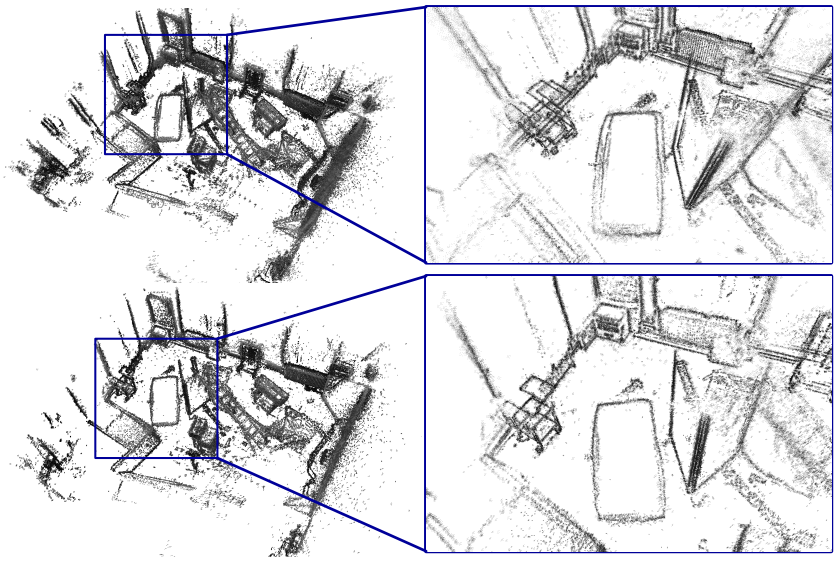


Figure 5.1: Estimated map by DSM with (bottom) and without (top) point reobservations in the V2_01_easy sequence of the EuRoC MAV dataset. DSM can produce consistent maps without duplicates.

can extract the rich information of map point reobservations reducing the drift in the estimates.

Transforming PBA based direct VO systems into VSLAM is not straightforward because there are several open challenges to solve:

1. When the camera revisits already mapped areas, the system has to select active keyframes that include map point reobservations. They are difficult to obtain because there are not point correspondences between keyframes. At the same time, we have to guarantee accurate map expansion during exploration. Therefore, we propose the Local Map Covisibility Window (LMCW) criteria to select active keyframes that observe the same scene region, even if they are not close in time, and the map point reobservations. It uses a combination of temporal and covisibility criteria to select the active keyframes.

2. The PBA optimization includes map points and keyframes distant in time and, hence, affected by the estimation drift. Normally, the photometric convergence radius is around 1-2 pixels due to image linearization and,

thus, a standard PBA cannot compensate the drift. We propose a multiscale PBA optimization to handle successfully these convergence difficulties. This strategy allows to exploit the rich geometrical information provided by point reobservations.

3. We have to ensure the robustness of the PBA against spurious observations. They result mainly from the widely separated active keyframes –in contrast with the close keyframes of VO– which render occlusions and scene reflections that violate the photo-consistency assumption. We incorporate a robust influence function based on the t-distribution into the PBA that neutralizes the adverse effect of the spurious observations.

This chapter presents the proposed direct VSLAM system, DSM (Direct Sparse Mapping). Up to our knowledge, this is the first fully direct monocular VSLAM method that is able not only to detect point reobservations but also to extract the rich information they provide (see Fig. 5.1). Sec. 5.1 provides an overview of some related works. Then Sec. 5.2, 5.3, 5.4 and 5.5 present all the steps that have been carried out to solve the problem of monocular VSLAM with a fully direct formulation. Finally, Sec. 5.6 describes the experiments that validate the performance of DSM in terms of both camera trajectory and reconstruction map accuracy. The latter is usually not reported in VO/VSLAM methods.

5.1 Related Work

The first real-time monocular VSLAM methods were indirect approaches, using FAST (Rosten and Drummond, 2006) and Harris (Harris and Stephens, 1988) corners associated across images in the form of 2D fixed correspondences. The 3D geometry was estimated minimizing the reprojection error. They rely on the repeatability of the corner detectors and required rich visual texture. Thanks to feature descriptors, they associate distant images. Davison et al. (2007) present MonoSLAM, which matches sparse keypoints and recovers the scene geometry in an EKF-based framework, later extended by Civera et al. (2008) to include a parametrization in inverse depth. Klein and Murray (2007) in PTAM propose for the first time to parallelize the tracking and mapping tasks, demonstrating the viability of using a BA scheme to maintain a persistent map in small workspaces. Klein and Murray (2008) extend PTAM to handle edgelets in the map and improve the robustness of the whole system to motion blur. Later, Strasdat et al. (2011) proposes a double window optimization to extend the potential of feature-based VSLAM to long-term applications. It combines a local BA with a global pose-graph optimization using covisibility constraints based on point matches.

Following these works, ORB-SLAM (Mur-Artal et al., 2015) presents the current reference solution among indirect VSLAM approaches. It is a full VSLAM approach that includes: a traditional BA with map reuse capabilities, loop closure correction and relocalization. Up to date, it is the most accurate monocular VSLAM method in many scenarios. The key aspects of its precision come from the management of map point reobservations in the BA using an appearance based covisibility graph. Later, Mur-Artal and Tardos (2016a,b) extend ORB-SLAM to stereo, RGB-D and visual-inertial systems. Similarly, DSM transfers the main ideas of feature-based VSLAM techniques to direct systems significantly increasing the accuracy of their estimates. As a direct approach DSM does not compute explicit point matches and, thus, cannot build an appearance based covisibility graph. Instead, DSM relies on geometric constraints to build covisibility connections between far in time keyframes. In addition, it works with a smaller window of covisible keyframes than ORB-SLAM to control computational limitations.

Recently, VO approaches have shown impressive performance. SVO (Forster et al., 2014) proposes an hybrid approach to build a semi-direct odometry system. They use direct techniques to track and triangulate points but they

ultimately optimize the reprojection error of those points in the background. Later, Forster et al. (2017) extend SVO to multi-camera systems and to track edgelets. OKVIS (Leutenegger et al., 2015) presents a feature-based Visual-Inertial Odometry (VIO) system that continuously optimizes the geometry of a local map marginalizing the rest. More recently, Engel et al. (2016a) made a breakthrough with their DSO, the first fully direct VO approach that jointly optimizes motion and structure formulating a PBA and including a photometric calibration into the model (Engel et al., 2016b). Inspired by OKVIS, DSO performs the optimization over a sliding-window, where old keyframes as well as points that leave the field of view of the camera are marginalized. It has shown impressive odometry performance and it is the reference among direct VO methods. However, as a pure VO approach DSO cannot reuse map points once they are marginalized which causes camera localization drift and map inconsistencies. Similar to other systems, DSO has been extended to stereo (Wang et al., 2017), omnidirectional (Matsuki et al., 2018), rolling shutter (Schubert et al., 2018) and visual-inertial (von Stumberg et al., 2018) systems. DSM uses the same photometric model of DSO and goes one step further to build the first fully direct VSLAM solution with a persistent map. The experiments report that using a VSLAM scheme achieves a significant accuracy increase of the camera trajectory when compared to the VO of DSO.

Many VO systems have been extended to cope with loop closures. Most propose to include a feature-based Bag of Binary Words (DBoW) to detect loop closures and estimate pose constraints between keyframes, following Galvez-López and Tardos (2012). Then, a pose-graph optimization finds a correction for the keyframe trayectory. VINS-mono (Qin et al., 2018) uses a similar front-end to OKVIS but includes additional BRIEF features to perform loop closure. LSD-SLAM (Engel et al., 2014) was the first direct monocular VO for large-scale environments. The method recovers semi-dense depth maps using small-baseline stereo comparisons and reduces accumulated drift with a pose-graph optimization. Loop closures are detected using FAB-MAP (Cummins and Newman, 2008), an appearance loop detection algorithm, which uses different features to those of the direct odometry. LSD-SLAM was also extended to many other visual systems such as stereo (Engel et al., 2015), omnidirectional (Caruso et al., 2015), visual-inertial (Usenko et al., 2016). LDSO (Gao et al., 2018) extends DSO with a conventional ORB-DBoW to detect loop closures and reduces the trajectory drift by pose-graph optimization.

All these methods have the next drawbacks:

1. They use a different objective function and points to those of the odometry.
2. Loop closure detection relies on feature repeatability, missing many corrections.
3. The error correction is distributed equally over keyframes, which may not be the optimal solution.
4. Although the trajectory is spatially corrected, existing information from map points is not reused and, thus, ignored during the optimization.

In contrast, full VSLAM systems like ORB-SLAM and DSM reuse the map information thanks to its persistent map. The reobservations are processed with their standard BA (either geometric or photometric), resulting in more accurate estimates. Thanks to the improvement in accuracy the need of loop closure detection and correction is postponed to trajectories longer than in their VO counterparts.

Moreover, DVO Kerl et al. (2013) proposes a probabilistic formulation for direct image alignment techniques. Inspired by Lange et al. (1989), they show the robustness of using a t-distribution to manage the influence of noise and outliers. Furthermore, Babu et al. (2016) demonstrate that the t-distribution represents well photometric errors while not geometric errors. We incorporate these ideas into the sparse photometric model together with a novel outlier management strategy. In this way, we make the non-linear PBA optimization robust to spurious point observations. They normally appear as a result of widely separated active keyframes and lack of explicit point matches.

5.2 Direct Mapping

The proposed VSLAM system consists of a tracking front-end (Sec. 5.4) and an optimization back-end (Sec. 5.3). The front-end is involved in tracking frames and points, and also provides the coarse initialization for the optimization. The back-end determines which keyframes form the local window (Sec. 5.3.1) and jointly optimizes all the active keyframes and map point parameters in the PBA (Sec. 5.3.2). Fig. 5.2 shows an overview of the system structure.

The main contributions of the proposed method are related with the back-end and, thus, we consider convenient to present it first. Later, we present the front-end, which is comparable to other direct approaches and, finally, the system bootstrapping. Similarly to most VSLAM systems (Engel et al., 2016a, Klein and Murray, 2007, Mur-Artal et al., 2015) the front-end and the back-end run in two parallel threads:

1. The tracking thread obtains the camera pose at frame rate (Sec. 5.4.1). It also decides when the map needs to grow by marking tracked frames as keyframes (Sec. 5.4.2).

2. The mapping thread processes all new frames to track points from active keyframes (Sec. 5.4.3). Besides, if the new frame is marked as a keyframe, the local window is recalculated, new points are activated and the PBA optimizes motion (keyframes) and structure (points) together using active keyframes. Finally, it maintains the model globally consistent, i.e. removes outliers, detects occlusions and avoids point duplications (Sec. 5.3.3).

The persistent map is composed of keyfames that are activated or deactivated according to covisibility criteria with the latest keyframe. The absolute pose of a keyframe i is represented by the transformation matrix $\mathbf{T}_i \in SE(3)$. For each keyframe, we select as candidate points those with a locally high gradient module and spread over the image. Each map point $\mathbf{p}$ is created in a keyframe and its pose is coded as its inverse depth. Thus, for each keyframe we store the raw image and the associated map points. We assume all images to be undistorted and use the pinhole model to project a point from 3D space to the image plane. Chapter 3 presents a more detailed explanation of the parameterization used for cameras and points.

The Local Map Covisibility Window (LMCW) (Sec. 5.3.1) selects which keyframes are active and form the local window. Once a keyframe is active,

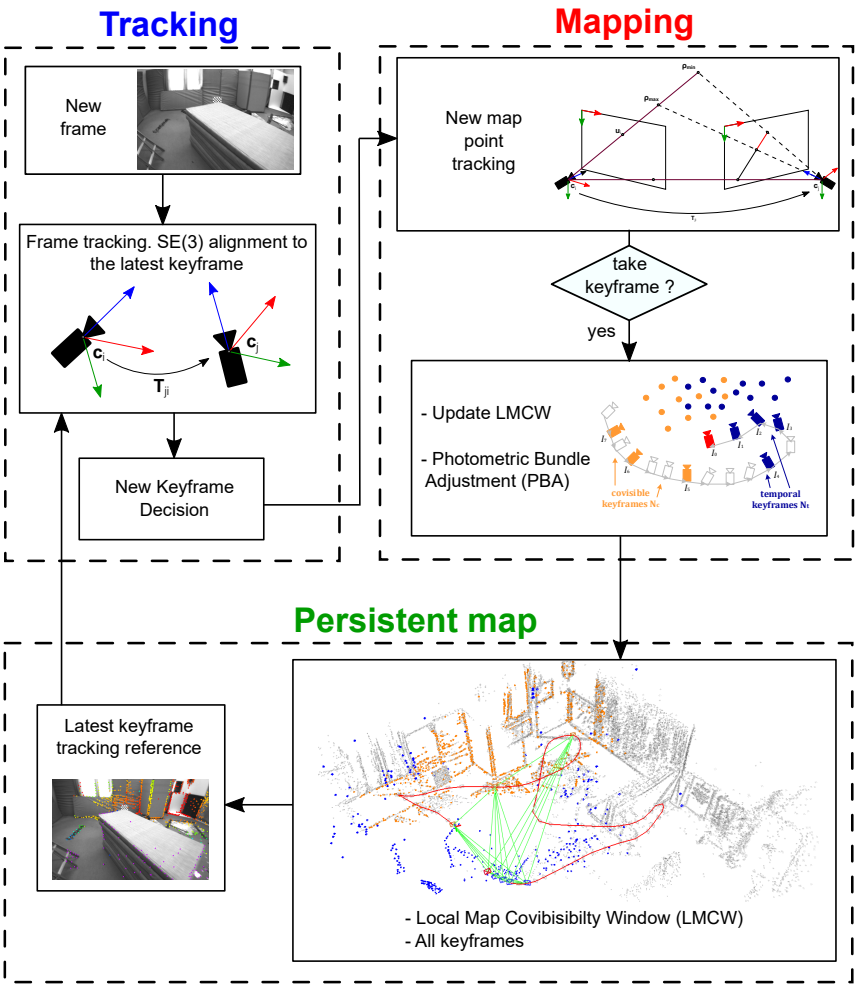


Figure 5.2: Overview over the complete DSM algorithm. Each new frame is tracked using image alignment techniques with respect to the latest keyframe. Then, each tracked frame is used to initialize new candidate points from the LMCW. If the tracked frame is spawned as keyframe, the LMCW is updated, new map points are activated and the PBA jointly optimizes all the model parameters (cameras and points). Finally, we maintain a persistent map to select active keyframes according to covisible criteria with the latest keyframe.

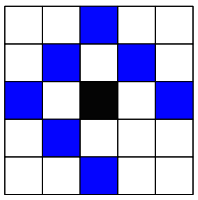


Figure 5.3: The pattern structure used to evaluate a single point. It is composed on $\mathcal{N}_p = 8$ spread pixels around the target point $\mathbf{p}$. Engel et al. (2016a) showed that it achieves a good balance between precision and computational time.

all its parameters (pose and affine light model) and associated points (inverse depth) are optimized by the PBA. Otherwise, they remain fixed to maintain the system consistent to unobservable degrees of freedom (gauge freedom). During optimization, we will use $\boldsymbol{\zeta} \in SE(3)^n \times \mathbb{R}^{2n+m}$ to represent the set of optimized parameters (n keyframes and m points) and $\delta\boldsymbol{\zeta} \in se(3)^n \times \mathbb{R}^{2n+m}$ to denote the increments. Moreover, we use the left-compositional convention for all optimization increments, i.e. $\boldsymbol{\zeta}^{(t+1)} = \delta\boldsymbol{\zeta}^{(t)} \boxplus \boldsymbol{\zeta}^{(t)}$ (see Sec. 3.2.3 and 3.5.6). This direct VSLAM framework enables to build a persistent map and reuse existing map information from old keyframes directly in the photometric bundle adjustment.

5.2.1 Photometric Model

The same photometric function proposed in (Engel et al., 2016a) is used in the whole system, i.e. geometry initialization (camera and point tracking), local windowed PBA and map reuse. For each point $\mathbf{p}$, we evaluate the sum of square intensity differences r_k over a small pattern $\mathcal{N}_p$ around it between the host I_i and target I_j images (see Fig. 5.3). We include an affine brightness transfer model to handle the camera automatic gain control and changes in scene illumination. The observation of a point $\mathbf{p}$ in the keyframe I_j is coded by:

$$E_p = \sum_{\mathbf{u}_k \in \mathcal{N}_p} w_k r_k^2 = \sum_{\mathbf{u}_k \in \mathcal{N}_p} w_k \left((I_i[\mathbf{u}_k] - b_i) - \frac{e^{a_i}}{e^{a_j}} (I_j[\mathbf{u'}_k] - b_j) \right)^2, \tag{5.1}$$

where $\mathbf{u}_k$ is each of the pixels in the pattern; $\mathbf{u'}_k$ is the projection of $\mathbf{u}_k$ in the target frame with its inverse depth ρ_k, given by $\mathbf{u'}_k = \pi(\mathbf{T}_{ji} \cdot \pi^{-1}(\mathbf{u}_k, \rho_k))$ with $\mathbf{T}_{ji} = \mathbf{T}_j^{-1}\mathbf{T}_i$; a_i, b_i, a_j, b_j the affine brightness functions for each frame; and

$w_k = w_{r_k} w_{g_k}$ a combination of the robust influence function w_{r_k} and a gradient dependent weight w_{g_k}:

$$w_{g_k} = \frac{c^2}{c^2 + \| \nabla I \|_2^2}, \tag{5.2}$$

which works as a heuristic covariance in the Maximum Likelihood (ML) estimation, reducing the influence of high gradient pixels due to noise. To sum up, the photometric cost function (Eq. 5.1) depends on geometric ($\mathbf{T}_i, \mathbf{T}_j, \rho$) and photometric parameters (a_i, b_i, a_j, b_j).

Note that since we are using the inverse depth, we need to use the linear 2D point mapping function presented in Sec. 3.4.2, which exploits all the benefits of the inverse depth parameterization. We use this formulation whenever we need to transform a 2D point from one image to another and during jacobian computation (see App. C).

5.3 Back-End

The back-end manages the persistent map, which includes all the keyframes and map points. It consists in the following tasks:

- Determine the LMCW (local window) using the pose information from the latest keyframe. This step selects the active keyframes that observe the same scene region even if they are not close in time. This is crucial to guarantee that map point reobservations are inserted into the PBA when revisiting already mapped areas.
- Activate new points from the LMCW and avoid point duplications with existing points in the persistent map.
- Jointly optimize for all the involved parameters using a multiscale photometric bundle adjustment. This comprises all the active keyframes in the LMCW and its corresponding active points. Consequently the PBA includes map points and keyframes distant in time and affected by the estimation drift.
- Decide which points are visible in which frames and maintain the model globally consistent. This is, identify which observations do not fulfill the photometric consistency and, consequently, decide when a map point should be removed. This implies to remove outliers and detect occlusions.

5.3.1 LMCW: Local Map Covisibility Window

This section presents the LMCW and the strategy to select its active keyframes and active map points. It is a combination of temporal and covisibility criteria with respect to the latest keyframe being created. Fig. 5.4 shows the LMCW selection strategy.

The first part is composed of N_t temporally connected keyframes that form a sliding-window like Engel et al. (2016a). This part is critical during exploration because it initializes new points (Sec. 5.4) and maintains the accuracy in odometry. Whenever a new keyframe is created, we insert it into the temporal part and remove another one. Thus, we maintain fixed size temporal keyframes. The strategy that selects the removed keyframe from the temporal part is summarized as:

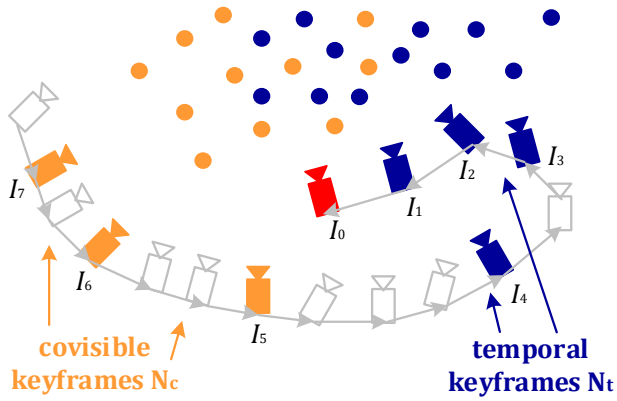


Figure 5.4: LMCW example with $N_w = 7$ and the latest keyframe. It is composed of $N_t = 4$ temporal (blue) and $N_c = 3$ covisible (orange) active keyframes. The red camera represents the latest keyframe being created.

1. Always keep the last two keyframes (I_1 and I_2). This ensures the odometry accuracy during challenging exploratory motions, such as rotations. It avoids premature fixation of keyframes location, guaranteeing that keyframes are well optimized before fixing them.

2. The remaining keyframes are evenly distributed in space. We drop the keyframe I_i that maximizes:

$$s(I_i) = \sqrt{d(I_0, I_i)} \quad \sum_{j=1}^{N_t} \left(d\left(I_i, I_j\right)\right)^{-1}, \tag{5.3}$$

   where $d(I_i, I_j)$ is the L_2 distance between keyframes I_i and I_j. This strategy favors observations rendering high parallax into the PBA, which increases the accuracy.

The second part is composed of N_c covisible keyframes. We aim to select keyframes covisible with those in the temporal part. Additionally, we seek to fill the latest keyframe I_0 with reobserved map points, favoring map points imaged in depleted areas (image areas where no other map points are imaged). Our strategy to achieve this goal is summarized as:

1. Compute the distance map to identify the depleted areas. All the map points from the temporal part are projected into the latest keyframe, then

the distance map registers, for every pixel, the L_2 distance to its closest map point projection.

2. Select a keyframe among the list of old keyframes, the one that maximizes the number of projected points in the depleted areas using the distance map. We discard points that form a viewing angle above a threshold to detect and remove potential occluded points as early as possible.

3. Update the distance map to identify the new depleted areas.

4. Iterate from (2) until N_c covisible keyframes are selected or no more suitable keyframes are found.

The covisible part incorporates already mapped areas in the LMCW before activating new map points. The proposed strategy avoids map point duplications ensuring the map consistency. The values of N_t and N_c are tuned experimentally in Sec. 5.6.

5.3.2 Photometric Bundle Adjustment (PBA)

Every time a new keyframe is created, all model parameters are optimized by minimizing the error from Eq. (5.1) over the LMCW of active keyframes $\mathcal{K}$. The total error is given by:

$$E = \sum_{I_i \in \mathcal{K}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in obs(\mathbf{p})} \sum_{\mathbf{u}_k \in \mathcal{N}_p} w_k r_k^2(\boldsymbol{\zeta}), \tag{5.4}$$

where $\mathcal{P}_i$ is the set of points in I_i and $obs(\mathbf{p})$ the set of observations for $\mathbf{p}$. Note that the LMCW reuses map point observations for which the initial solution is not inside the convergence radius and, thus, the PBA is not able to correct. We propose to use a coarse-to-fine optimization scheme over all active keyframes. In each level, we iterate until convergence and use the estimated geometry as an initialization for the next level. The same points are used across all levels and each level is treated independently, i.e. neither the influence function nor outlier decisions are propagated across the levels (Sec. 5.3.3). In this way, we are able to handle larger camera and point increments $\delta\boldsymbol{\zeta}$ with the photometric model.

We minimize Eq. (5.4) using the iteratively re-weighted Levenberg-Marquardt algorithm (see Sec. 3.5.3). From an initial estimate

$\boldsymbol{\zeta}^{(0)}$, each iteration t computes weights w_k and photometric errors r_k to estimate an increment $\delta\boldsymbol{\zeta}^{(t)}$ by solving for the minimum of a second order approximation of Eq. (5.4), with fixed weights:

$$\delta\boldsymbol{\zeta}^{(t)} = -\mathbf{H}^{-1}\mathbf{b}, \tag{5.5}$$

with $\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\text{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})$, $\mathbf{b} = \mathbf{J}^T\mathbf{W}\mathbf{r}$ and $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a diagonal matrix with the weights w_k, $\mathbf{r}$ is the error vector and $\mathbf{J} \in \mathbb{R}^{m\times d}$ is the Jacobian of the error vector with respect to a left-composed increment given by:

$$\mathbf{J}_k = \left.\frac{\partial r_k(\delta\boldsymbol{\zeta} \boxplus \boldsymbol{\zeta}^{(t)})}{\partial\delta\boldsymbol{\zeta}}\right|_{\delta\boldsymbol{\zeta}=0}. \tag{5.6}$$

The main difference of the PBA with the traditional geometric BA is that each residual r_k depends on two keyframes instead of one. This is due to the fact that each point is associated to a keyframe and relatively represented to it. However, this does not affect the sparsity structure of the problem and we take advantage of the Schur complement trick to solve the reduced problem (see Sec. 3.5.5). Besides, we use variable scaling techniques (see Sec. 3.5.5) to take into account the different magnitudes between the model parameters and accelerate the optimization. The gauge freedoms are controlled fixing all other keyframes that are covisible with the active ones.

The PBA is implemented using the Ceres optimization library (Agarwal et al.) with analytic derivatives. We provide the analytical expressions of the jacobians $\mathbf{J}_k$ in the Appendix C, which can be efficiently computed using the adjoint theory. Image gradients are computed using central pixel differences at integer values. For subpixel intensity and gradient evaluation, bilinear interpolation is applied.

5.3.3 Robust Non-linear PBA

The LMCW selects widely separated active keyframes according to geometric criteria without any consideration about the actual photo-consistency between the images of the map points in the selected keyframes. Hence, it is possible that some of the points do not render photo-consistent images, because they suffer, for example, from occlusions or scene reflections (see Fig. 5.5).

To make our PBA robust with respect to this lack of photo-consitency, we propose an outliers management strategy based on the photometric error

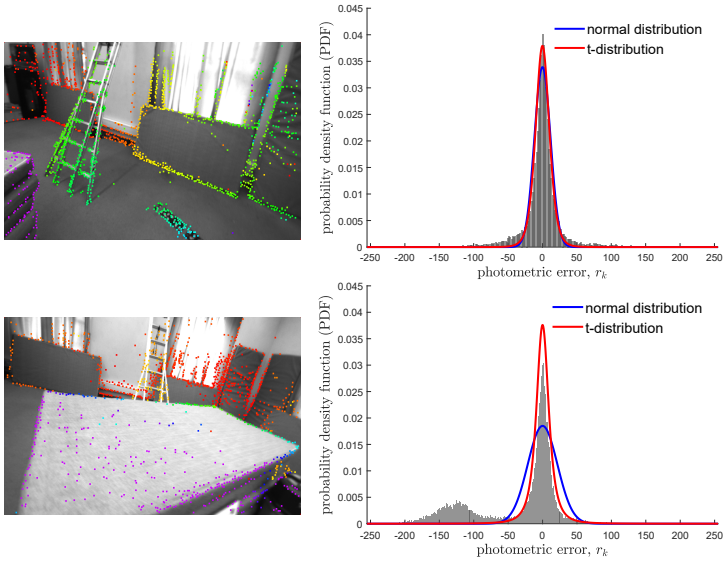


Figure 5.5: Probabilistic error modeling. The top row shows the case where most of the map points are photo-consistent, then both normal and t-distribution models fit well the photometric errors. The bottom row shows a challenging situation where covisible reobservations introduce many outliers due to occlusions, the t-distribution fits the observed errors better than the normal. On the left, the keyframe along with the point depth map after outlier removal.

distribution, from which we derive the appropriate weights for Eq. 5.4. According to the probabilistic approach, optimizing the Eq. 5.4 is equivalent to minimizing the negative log-likelihood of model parameters $\boldsymbol{\zeta}$ given independent and equally distributed errors r_k,

$$\boldsymbol{\zeta}^* = \underset{\boldsymbol{\zeta}}{\operatorname{argmin}} - \sum_{k}^{n} \log p(r_k \mid \boldsymbol{\zeta}). \tag{5.7}$$

The minimum of Eq. 5.7 is computed equating its derivatives to zero given by

$$-\sum_{k}^{n} \frac{\partial \log p(r_k \mid \boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}} = -\sum_{k}^{n} \left(\frac{\partial \log p(r_k)}{\partial r_k} \cdot \frac{\partial r_k}{\partial \boldsymbol{\zeta}} \right) = 0. \tag{5.8}$$

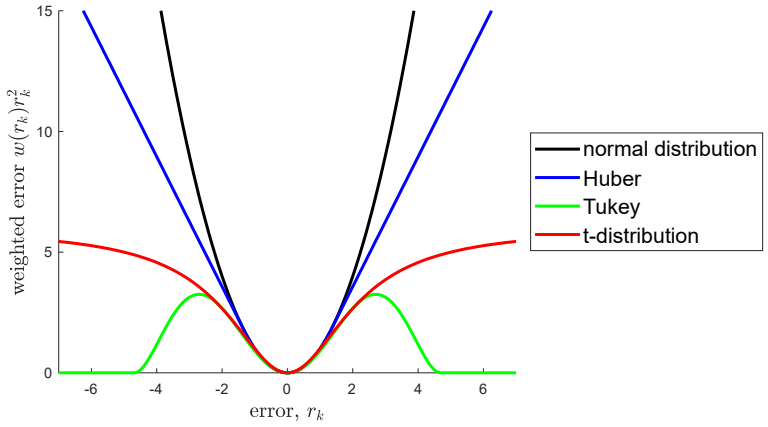


Figure 5.6: Comparison of different influence functions. Huber is plotted with $k = 1.345$ and Tukey with $k = 4.685$ assuming a Gaussian distribution $\mathcal{N}(0, 1)$. The t-distribution is plotted with $\nu = 5$ and $\sigma_t = 1$.

This is equivalent to minimizing the re-weighted least-squares Eq. 5.4:

$$\frac{\partial r_k}{\partial \boldsymbol{\zeta}} \cdot w(r_k) \cdot r_k = 0, \tag{5.9}$$

with the weights defined as

$$w(r_k) = -\frac{\partial \log p(r_k)}{\partial r_k} \cdot \frac{1}{r_k}. \tag{5.10}$$

Therefore, the solution is directly affected by the photometric error distribution $p(r_k)$ (see Kerl et al. (2013) for further details). Fig 5.6 shows the influence of different weight distributions in a least squares optimization. Next we describe them with more detail.

Gaussian distribution If errors are assumed to be normally distributed around zero $\mathcal{N}(0, \sigma_n^2)$, the model of error distribution is $p(r_k) \propto \exp(r_k^2/\sigma_n^2)$. This model leads to a constant distribution of weights which is a standard least squares minimization. Thus, it treats all points equally and outliers cannot be neutralized:

$$w_n(r_k) = \frac{1}{\sigma_n^2}. \tag{5.11}$$

Student's t-distribution Recently, Kerl et al. (2013) has analyzed the distribution of dense photometric errors for RGB-D odometry. It showed that the t-distribution explains dense photometric errors better than a normal distribution, providing a suitable weight function:

$$w_t(r_k) = \frac{\nu + 1}{\nu + (\frac{r_k}{\sigma_t})^2}, \quad \text{when } \mu = 0. \tag{5.12}$$

We have experimentally studied the sparse photometric errors and we conclude that the t-distribution also explains the sparse model properly (Fig. 5.5). In contrast to the normal distribution, the t-distribution quickly drops the weights as errors move to the tails, assigning a lower weight to outliers. Besides, instead of fixing the value of the degrees of freedom $\nu = 5$ as in Kerl et al. (2013), we study the behavior of the model when ν is fitted together with the scale σ_t (see Sec. 5.6). To fit the t-distribution, we minimize the negative log-likelihood of the probability density function with respect to ν and σ_t using the gradient free iterative Nelder-Mead method (Lagarias, Jeffrey et al., 1998). Besides, we filter out the gross outliers before fitting the t-distribution. We approximate the scale value $\hat{\sigma}$ using the Median Absolute Deviation (MAD) as $\hat{\sigma} = 1.4826$ MAD and reject errors that $r_k > 3\hat{\sigma}$.

M-estimators Whether the distribution of errors is hard to know or it is assumed to be normally distributed, using M-estimators is a popular solution. We have previously presented the two most popular alternatives, Huber and Tukey, in Sec. 3.5.4. The Huber estimator gives linear influence to the outlier, whereas the Tukey estimator removes the influence of the outliers by setting the weight to zero (see Fig. 5.6). Between these two alternatives, Huber is the most used one since it does not totally remove high error measurements but it decreases their influence, which is crucial for reobservation processing. Note in Fig. 5.6 how the t-distribution achieves a balance between Huber and Tukey. It is more aggressive with outliers than Huber but, in contrast to Tukey, it does not explicitly remove them.

Implementation of the probabilistic model into the PBA

We have studied the error distribution in each keyframe and concluded that there are differences between them. These variations might come from motion blur, occlusions or noise (see Fig. 5.5). Hence, we fit the error distribution for

each keyframe separately using all the observations from active points in that keyframe. This allows to adapt the PBA to different situations, e.g. certain error values might be considered as an outlier in a regular keyframe but inlier in a challenging one due to motion blur.

Computing the error distribution and, thus, the weight distribution each iteration changes the objective function (Eq. 5.7) and the performance of the optimization might degrade. We propose to compute the error distribution only at the beginning of each pyramid level and to maintain it fixed during all the optimization steps. At the end of the PBA, the error distribution is recomputed again using the photometric errors obtained from the best geometry solution $\boldsymbol{\zeta}^*$.

Outlier management

It is crucial to detect and remove outlier observations as soon as possible to maintain the PBA stability. To achieve this, we will exploit the information from each observation, which includes measurements from eight different pixels. We propose to build a mask for each point and mark each pixel measurement r_k as inlier or outlier. To consider a pixel measurement as inlier, the photometric error has to be lower than the 95% percentile of the error distribution of the target keyframe. For challenging keyframes the threshold will be higher, being more permissive, whereas for regular ones it will be lower, being more restrictive. When the current local PBA is finished, we count the number of inlier pixels in the mask. Whenever an observation contains a number of outlier pixels larger than a 30%, the observation is marked as an outlier and removed from the list of observations of the point. Besides, during the optimization, if the number of outlier pixels is larger than a 60%, the observation is directly discarded from the current optimization step, i.e. $w(r) = 0$.

We also detect and remove outlier points from the map. We propose to control the number of observations in each point to decide if it is retained. To retain a new point, it must be observed in all the new keyframes after its creation. When it has been observed in three keyframes it is considered a mature point. Mature points are removed if the number of observations falls below three.

5.4 Front-End

The front-end is in charge of tracking each new input frame and point candidates from the LMCW. It also decides when the map needs to be expanded with a new keyframe. As a result, it provides the coarse initialization of all the new parameters involved in the PBA, i.e. camera pose, camera affine light and point inverse depth.

5.4.1 Frame Tracking

Each new frame is tracked against a local map, which is updated after every new keyframe decision. The local map is formed with active points from the LMCW referenced to the latest keyframe. The frame pose and its affine brightness transfer model are computed by minimizing Eq. 5.1 in which the map points and the latest keyframe remain fixed. The initial estimation is provided by a velocity model. We use a coarse-to-fine optimization, as proposed in the PBA, to handle initial guesses with large errors. We use the same robust influence function of Sec. 5.3.3 to reduce the impact of high photometric errors.

Regarding the implementation, we use the inverse compositional approach (Baker and Matthews, 2004) to avoid re-evaluating Jacobians each iteration and reduce the computational cost. This is achieved changing the roles of the latest keyframe and the new frame: as we are estimating the relative motion between two cameras, it does not matter if we estimate the new frame motion or the keyframe motion. Thus, we can estimate the motion of the keyframe with respect to the frame and update the frame pose parameters using the inverse of the computed motion. The jacobians are estimated with respect to the keyframe, which is kept fixed with respect to the points, and remain constant during all the optimization. As a result, the jacobians can be precomputed and are required to be re-evaluated only at the beginning of each pyramid level.

5.4.2 New Keyframe Decision

Whenever we move towards unexplored areas, the map is expanded with a new keyframe. We use three different criteria with respect to the latest keyframe to decide if the tracked frame becomes a keyframe:

1. The map point visibility ratio between the latest keyframe and the tracked frame, i.e. $s_u = N^{-1} \sum \min(p_z/p'_z, 1)$, where N is the total number of visible points in the latest keyframe, p_z the point inverse depth in the latest keyframe and p'_z the point inverse depth in the tracked frame. The score is formulated to create more keyframes if the camera moves closer.

2. The tracked frame parallax with respect to the latest keyframe, defined as the ratio between the frame translation $\mathbf{t}$ and the mean inverse depth of the tracking local map $\bar{\rho}$: $s_t = \| \mathbf{t}\bar{\rho} \|_2$.

3. The illumination change, measured as the relative brightness transfer function between the tracked frame and the latest keyframe, i.e. $s_a = |a_k - a_i|$.

A heuristic score based on the weighted combination of these criteria determines if the tracked frame is selected as a new keyframe: $w_u s_u + w_t s_t + w_a s_a > 1$.

5.4.3 New Map Point Tracking

During exploration, the system requires to create new map points. Each keyframe contains a list of candidate points that are initialized and activated if so decided. We initialize the inverse depth of these candidate points using consecutive new tracked frames. To do so, we search along the epipolar line to find the correspondence with minimum photometric error (Eq. 5.1). Only distinctive points with low uncertainty will be activated and inserted into the PBA.

We follow a scheme similar to the one proposed in Engel et al. (2013). The inverse depth of each candidate point $\mathbf{u}_i$ is modelled by a Gaussian probability distribution $\mathcal{N}(\rho_i, \sigma^2_{\rho i})$, which is updated with every new tracked frame. We perform an exhaustive search along the epipolar line to find the best matching pair. If we already have an inverse depth hypothesis, the disparity search range is limited to $\rho_{max} = \rho_i + 2\sigma_{\rho i}$ and $\rho_{min} = \rho_i - 2\sigma_{\rho i}$. Otherwise, we search along the full disparity range. Fig. 5.7 illustrates the constrained search procedure.

The uncertainty of the disparity is estimated measuring the angle difference between the epipolar line direction and the point gradient direction. This is, a point with a gradient direction similar to the epipolar line will have high uncertainty and vice-versa. For example, if a point belongs to an horizontal edge and we are searching along the same edge, we will obtain many similar

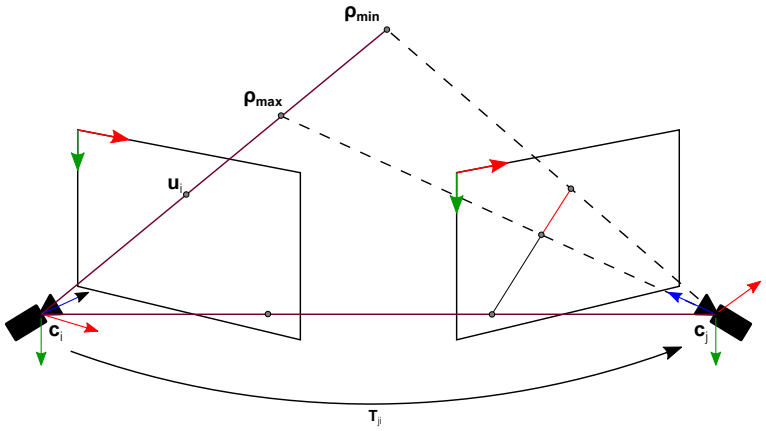


Figure 5.7: Epipolar constrained search of a candidate point given a prior inverse depth distribution.

matches without a clear winner. The uncertainty of the disparity is given by

$$\sigma_d^2 = \frac{\sigma_l^2}{(\mathbf{g} \cdot \mathbf{l})^2}, \tag{5.13}$$

where $\mathbf{g}$ is the point gradient, $\mathbf{l}$ the epipolar line direction and σ_l the uncertainty of the epipolar line computation. The value of the epipolar line uncertainty has been experimentally set to $\sigma_l = 0.5$. Additionally, we skip searches with high disparity uncertainty as we know that they would not provide reliable measurements.

During the epipolar search, we estimate the inverse depth for each corresponding pair (see Sec. 3.4) and map each pixel in the pattern $\mathcal{N}_p$ to the target frame using the estimated geometry. We also keep track of all the evaluated pairs and retain the best match only if the ratio of the residual with the second best match is bigger than 2. In this case, we obtain a distinctive point match $\mathbf{u}_j$, we propagate the uncertainty of the disparity σ_d to the uncertainty in the inverse depth σ_ρ. To do so, we obtain the inverse depth at the matched point and at disparity uncertainty extremes, and form the new inverse depth hypothesis $\mathcal{N}(\rho_j, \sigma_{\rho j}^2)$ as

$$\begin{aligned} \rho_j &= \mathcal{T}_\rho(\mathbf{u}_j), \\ \sigma_{\rho j} &= \max(|\mathcal{T}_\rho(\mathbf{u}_j + \sigma_d) - \rho_j|, |\mathcal{T}_\rho(\mathbf{u}_j - \sigma_d) - \rho_j|), \end{aligned} \tag{5.14}$$

where $\mathcal{T}_\rho(\mathbf{u}^*)$ is the inverse depth of the triangulated point for the corresponding pair $(\mathbf{u}_i, \mathbf{u}^*)$. In addition, we only keep the hypothesis of the match with the largest baseline as it provides the most accurate estimate.

Note that this delayed strategy requires several correspondences to obtain a good initialization as we are working with small baselines that render low parallax. To guarantee that we have enough initialized candidates to activate, we maintain candidate points from a keyframe until this is dropped from the temporal part of the LMCW. We only activate points that belong to image areas depleted from points (Sec. 5.3.1). Thus, when revisiting already mapped scene regions, only a few new points will be activated, as we will reuse existing map points.

5.5 Initialization

The estimation of the 3D geometry using a monocular system is a chicken-and-egg problem where the structure is required to estimate the motion and vice-versa (see Sec. 3.4). Thus, it is required to decouple the structure and motion problems. This section presents an automatic initialization algorithm to obtain the relative pose between two frames and triangulate an initial set of map points. The proposed method is similar to other indirect solutions (Mur-Artal et al., 2015, Schönberger and Frahm, 2016) but it uses the same points of the photometric model without requiring any specific feature detection and matching strategies. This allows to bootstrap the system in many difficult situations such as untextured scenes and motion blur. In addition, the initialization is independent of the scene structure (planar or not) and does not require the intervention of the user.

The automatic initialization algorithm follows these steps:

1. Set the first frame as the reference keyframe and select point candidates in the image as proposed in Sec. 5.2.

2. For each new input frame, track the position of point candidates in the new image using a pyramidal implementation of the Lucas-Kanade feature tracker (Bouguet, 1999), also known as optical flow. Fig. 5.8 shows some examples of this step. If not enough point matches are found, start again from step 1.

3. Using the previous matches, compute two geometrical models: a homography matrix assuming a planar scene and an essential matrix assuming a non-planar scene. We use a RANSAC scheme to remove the influence of outliers and obtain a more robust estimate. For each model count the number of inliers (N_H and N_E) using the reprojection error.

4. Select the most appropriate model using the heuristic inlier ratio given by

$$r = \frac{N_H}{N_E}. \tag{5.15}$$

   If the ratio is larger than $r > 0.8$ select the planar case (homography matrix), otherwise select the non-planar case (essential matrix).

5. Recover the motion from the selected geometrical model. To do so, we estimate all the possible solutions (due to projective ambiguities) and

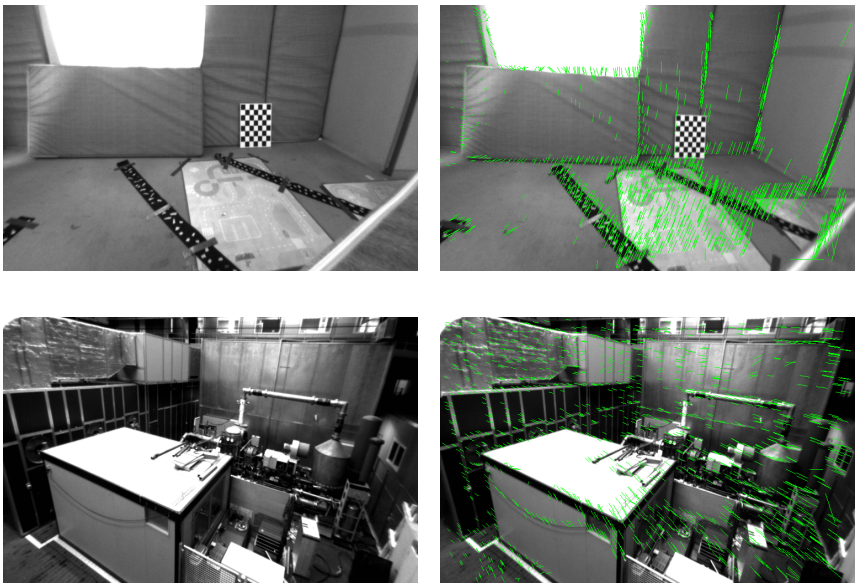


Figure 5.8: Examples of optical flow tracking of candidate points during system initialization. Green lines represent the motion of the candidate points in the image.

select the motion with the highest number of points in front of both cameras with a minimum number of points seen with parallax. If we find a clear winner, initialize the inverse depth of the point candidates in the first keyframe as proposed in Sec. 5.4.3. Otherwise, do not initialize points and continue from step 2.

6. Select the latest frame as a new keyframe and perform PBA to refine the geometry of the initial pair of keyframes and map points.

5.6 Experiments And Discussion

The experimental validation of the proposed system uses the publicly available EuRoC MAV dataset (Burri et al., 2015). This dataset presents two main advantages: first, it has three scenarios, two rooms (V1, V2) and a machine hall (MH), with very challenging motions and changes in illumination; second, it includes the 3D reconstruction ground-truth, which we also evaluate. We study the benefits of the VSLAM scheme of DSM with a version, DSM-SW (sliding-window), which only uses temporally connected keyframes as in Engel et al. (2016a). We compare our approach against state-of-the-art algorithms such as ORB-SLAM (Mur-Artal et al., 2015), DSO (Engel et al., 2016a) and LDSO (Gao et al., 2018). We evaluate the RMS Absolute Trajectory Error (ATE) and the Point to Surface Error (PSE). The ATE is computed using the keyframe trajectory for each sequence after Sim(3) alignment with the ground-truth. The PSE is estimated measuring the distance of the reconstructed model to the ground-truth surface after the trajectory alignment. The results are shown using normalized cumulative error plots, which provide the percentage of runs/points with an error below a certain threshold. These graphics provide both information about the accuracy and robustness of the evaluated method. All experiments are executed using a standard PC with an Intel Core i7-7700K CPU and 32 GB of RAM.

5.6.1 Parameter analysis and tuning

This section presents an experimental analysis of the main parameters and options defining the DSM performance. We also propose the tuning for the final system. To cover more cases, we run different experiments for the left and the right cameras of the stereo rig, and both in the forward and in the backward direction. We run each experiment 5 times. In total, we have made 220 experiments.

Coarse-to-fine PBA

We evaluate the effect of changing the number of pyramid levels N_p during the PBA. Fig. 5.9 shows the results for DSM-SW and DSM. Without the coarse-to-fine scheme DSM-SW performs better than DSM. Here, DSM is not able to benefit from point reobservations due to the accumulated drift. However, DSM is able to reuse map points for higher number of pyramid levels and

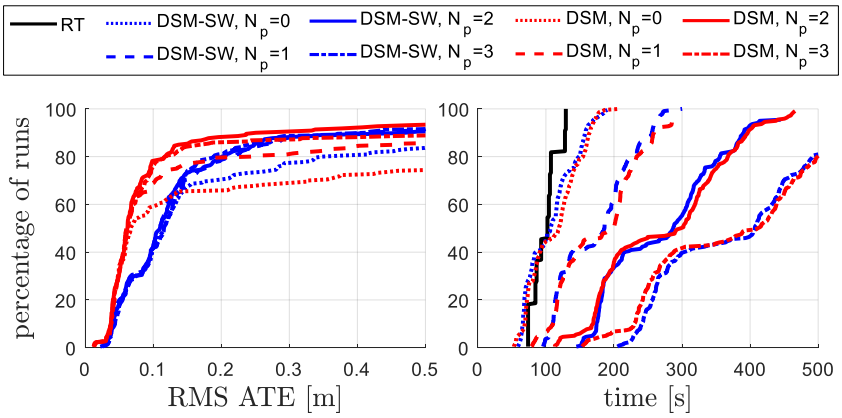


Figure 5.9: Number of pyramid levels N_p. RMS ATE (left) and processing times (right) compared with the RT (real-time) for different N_p.

it clearly achieves better accuracy. While a coarse-to-fine strategy certainly increases the accuracy of DSM, there is significantly less improvement for DSM-SW. This is the expected behavior since DSM requires larger convergence radius to process reobservations while DSM-SW does not (see Sec. 5.3.2). Note how DSM is able to process approximately the 80% of runs with a RMS ATE below 0.1m while DSM-SW only gets 40% of runs. Moreover, we see that using $N_p = 1$ with a sliding-window increases the performance. We also observe that increasing the number of levels after $N_p = 2$ for DSM does not increase accuracy but increases the runtime significantly.

Including reobservations in the PBA has little effect on the processing time. In contrast, the number of pyramids approximately increases the runtime by 50% for each level. Thus, we use $N_p = 2$ as default which achieves the best balance between efficiency and accuracy.

Number of PBA iterations

We also study the influence of the number of iterations in each pyramid level during the PBA. Fig. 5.10 presents the accuracy and processing time results for DSM. We observe that increasing the number of iterations after 3 has little impact in the accuracy. Apparently, it makes no sense to iterate until convergence each time if we are going to incrementally introduce new measurements to the

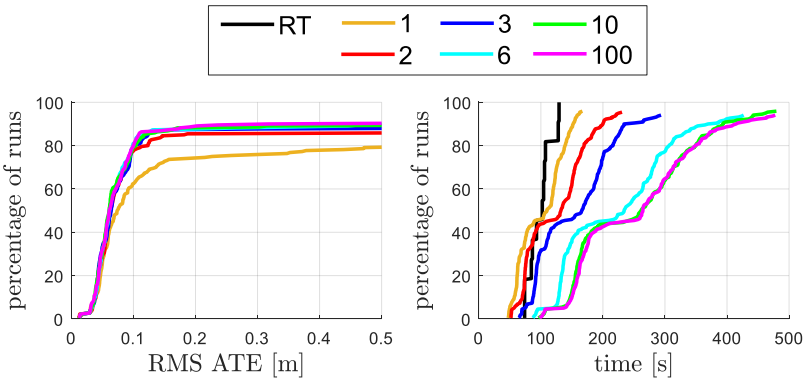


Figure 5.10: Number of PBA iterations. RMS ATE (left) and processing times (right) compared with the RT (real-time) for different number of iterations.

system that continuously change the local minimum. Moreover, the computational time increases linearly with the number of iterations. In general, we see that the PBA does not perform more than $6-7$ iterations. Regarding the map accuracy, we have observed a similar behaviour to the localization with relatively low impact in the precision after 3 iterations.

We believe that the number of iterations should be selected according to the application specifications. In the case, we want to find the maximum precision without time requirements it should be increased, otherwise 3 iterations provides sufficient accuracy with a computational performance closer to real-time. For the rest of the experiments, we will not limit the number of iterations to evaluate the maximum accuracy performance of the method.

Robust Influence Function

We study the effect of the selected model of weight distribution. Fig. 5.11 shows the results for the t-distrution and Huber models. In contrast to Kerl et al. (2013), we evaluate the influence of the model when the degrees of freedom ν are estimated together with the scale σ. For Huber, we study when the constant is fixed to $\lambda = 9$ and when it is dynamically changed with the MAD value. Interestingly, there is not significant difference between using fixed or dynamic values on both distribution models. However, the t-distribution performs better in challenging situations providing higher robustness than Huber. This comes

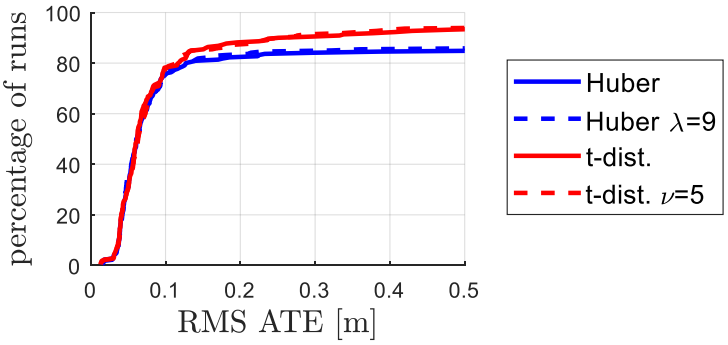


Figure 5.11: Robust influence function. Comparison of the RMS ATE between a Gaussian based M-estimator (Huber) and the t-distribution.

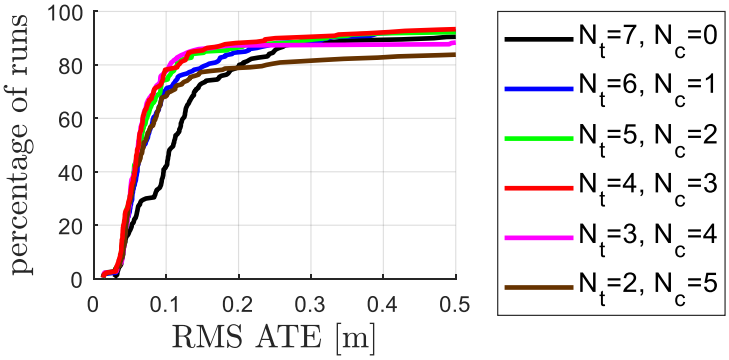


Figure 5.12: LMCW $N_w = N_t + N_c$. RMS ATE when changing the number of temporal N_t and covisible N_c keyframes.

from the fact that the t-distribution quickly drops the weights as errors move to the tails while the Huber model does not. We use the complete t-distribution model as default settings due to its flexibility handling challenging situations.

Number of covisible keyframes in the LMCW

We observe that increasing the number of covisible keyframes N_c increases the trajectory accuracy (Fig. 5.12.) With those covisible keyframes the PBA is able to handle point reobservations and to reduce the drift. However, the system requires temporally connected keyframes N_t to guarantee the odometry

robustness. Taking few temporal keyframes drastically reduces the accuracy. This is due to the fact that the temporal part ensures that new keyframes are well optimized and that enough new points are initialized during exploration. Thus, we use the combination of $N_t = 4$ and $N_c = 3$ as default settings, which achieves the best balance between precision and robustness.

5.6.2 Quantitative evaluation

This section presents a comparison of DSM against ORB-SLAM (Mur-Artal et al., 2015), DSO (Engel et al., 2016a) and LDSO (Gao et al., 2018). We report the results published in (Mur-Artal and Tardos, 2016b) for ORB-SLAM, in (Engel et al., 2016a) for DSO and we use the open-source implementation for LDSO. All the results are obtained using a sequential implementation without enforcing real-time operation using $N_w = 7$ active keyframes for all direct methods. We run on default settings all sequences both forward and backward, 10 times each, using left and right videos separately for a total of 440 runs.

Trajectory error

Table 5.1 reports the median errors for each forward sequence. Overall, we see that DSM-SW performs similarly to DSO. This is expected since both methods are based on the same sliding-window approach without a multiscale PBA. However, DSM-SW successfully executes all MH sequences, while DSO fails in MH_03_medium. This is probably due to the use of a more robust influence function in DSM-SW. DSM achieves higher accuracy in almost all sequences compared to the rest of direct approaches, DSO, LDSO and DSM-SW. DSO and LDSO only achieve slightly higher accuracy in a few sequences. In addition, ORB-SLAM obtains better results in V1 and V2, but DSM achieves the best performance for the MH sequences. Note that in contrast to ORB-SLAM, we do not incorporate any place recognition, pose-graph or relocalization modules. This shows the high precision of DSM due to point reobservations. In the sequence V1_03_difficult, DSM achieves an RMS ATE of only 7.1cm, which is by far the best performance among all the approaches tested. This sequence contains very rapid motions and illumination changes, which demonstrates the robustness of the proposed method. In addition, we successfully manage to complete all sequences and obtain an RMS ATE below 0.1m for all of them, except V2_03_difficult, where all of the compared approaches fail.

Table 5.1: RMS ATE [m] using forward videos for left (l) and right (r) sequences. (×) means failure and (-) no available data.

Seq.	ORB-SLAM	DSO	LDSO	DSM-SW	DSM
MH1_l	0.070	0.046	0.053	0.054	**0.038**
MH2_l	0.066	0.046	0.062	0.041	**0.036**
MH3_l	0.071	0.172	0.114	0.123	**0.053**
MH4_l	0.081	3.810	0.152	0.179	**0.060**
MH5_l	**0.060**	0.110	0.085	0.139	0.067
V11_l	**0.015**	0.089	0.099	0.099	0.094
V12_l	**0.020**	0.107	0.087	0.124	0.058
V13_l	×	0.903	0.536	0.888	**0.071**
V21_l	**0.015**	0.044	0.066	0.061	0.058
V22_l	**0.017**	0.132	0.078	0.123	0.058
V23_l	×	1.152	×	1.081	**0.669**
MH1_r	-	**0.037**	0.050	0.054	0.042
MH2_r	-	0.041	0.051	0.039	**0.037**
MH3_r	-	0.159	0.095	0.187	**0.049**
MH4_r	-	3.045	0.129	0.188	**0.059**
MH5_r	-	0.092	0.087	0.131	**0.064**
V11_r	-	0.047	0.662	0.031	**0.014**
V12_r	-	0.080	0.208	0.118	**0.048**
V13_r	-	1.270	0.642	1.313	**0.047**
V21_r	-	**0.027**	0.040	0.032	0.035
V22_r	-	0.059	0.068	0.314	**0.056**
V23_r	-	0.540	**0.171**	0.889	0.484

Mapping vs Pose-Graph

Comparing LDSO and DSM shows the differences in using a VO scheme with a pose-graph in contrast to a VSLAM scheme. Fig. 5.13 shows the RMS ATE for all the evaluated sequences for LDSO and DSM. Overall, we observe that DSM achieves better accuracy. We also see that reusing existing map points allows completing successfully a higher percentage of sequences. We build a persistent map and reuse map points to support the odometry estimation instead of permanently marginalizing all points that leave the local window. This can also be observed in Fig. 5.14. While DSM is able to process 80% of sequences

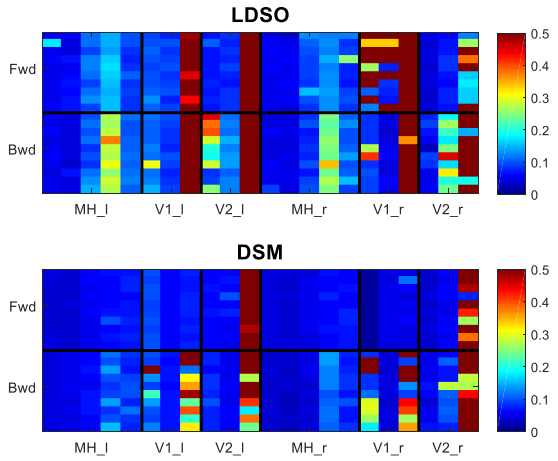


Figure 5.13: Full evaluation results. For each sequence (X-axis) we plot the RMS ATE [m] in each iteration (Y-axis), with a total of 440 runs.

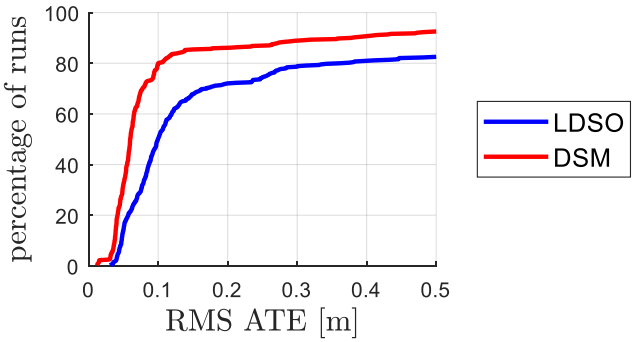


Figure 5.14: RMS ATE for LDSO and DSM.

with an RMS ATE below 0.1m, LDSO can only handle 50% of runs under this limit.

Moreover, we have observed that in some sequences LDSO misses many available loop closures due to a lack of feature matches. This makes the odometry drift until a larger correction loop is detected, causing a temporally inconsistent trajectory and structure estimations. Fig. 5.15 shows the evolution of the RMS ATE along the trajectory. It can be seen the effect of missing loop closures

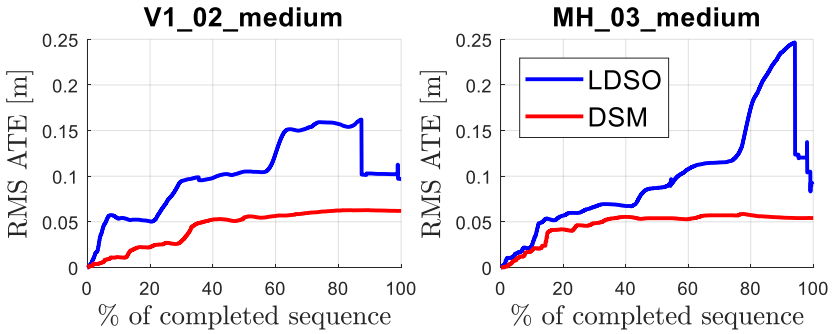


Figure 5.15: VSLAM vs VO + Pose-Graph. RMS ATE after processing each keyframe in the trayectory. It shows the time evolution of the error. While a feature-based pose-graph strategy may miss many loop closures, a VSLAM scheme continuously reuses existing information to provide more accurate and reliable estimates in time.

with a feature-based pose-graph strategy. In contrast, building a persistent map enables reusing existing map information continuously, which maintains the trajectory accuracy stable in time. Although the final RMS ATE is similar in both systems, the navigation estimation using a VSLAM approach is more accurate and, thus, more reliable. This clearly shows that using a VSLAM scheme provides better accuracy performance compared to a VO scheme with a pose-graph.

Map error

Fig. 5.16 shows the distance between the reconstructed points and the ground-truth surface. We compare all the sequences against LDSO except in V2_03_difficult where LDSO fails. Clearly, incorporating map point reobservations into the PBA increases not only the trajectory accuracy but the reconstruction precision too. Although the final trajectory RMS ATE is similar in some sequences, such as in V1_01_easy, the map is without a doubt more accurate in DSM. Besides, we have observed that LDSO creates ten times more points than DSM for these sequences, due to the fact that DSM reuses existing map points avoiding duplications.

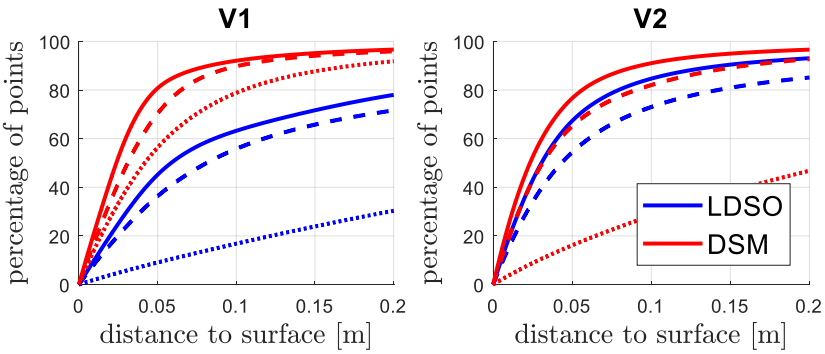


Figure 5.16: Map error. For each scene we show the accumulated PSE distribution using all the reconstructed 3D points for all runs. Solid lines (—) present easy sequences, dashed lines (- - -) medium and dotted lines (· · ·) difficult ones for each scene.

Table 5.2: Processing time and keyframe frecuency.

Operation	Median [ms]	Mean [ms]	St.D. [ms]
Frame & Point Tracking	7.44	7.45	0.31
Local PBA	888.77	908.53	121.10
Keyframe Period	396.28	397.22	177.51

Processing time

Table 5.2 reports the processing time required for each part of the method, as well as the used keyframe period time. In our current initial implementation, PBA is the bottleneck of the processing cost. We observe that it should be twice faster to obtain the required keyframe creation rate. It is possible to improve the runtime significantly using SIMD instructions to process each patch. Besides, many of the operations can be parallelized as they are independent for each point. We believe using these upgrades could make DSM run in real-time applications since the mapping thread is not required to run at frame rate but at keyframe rate.

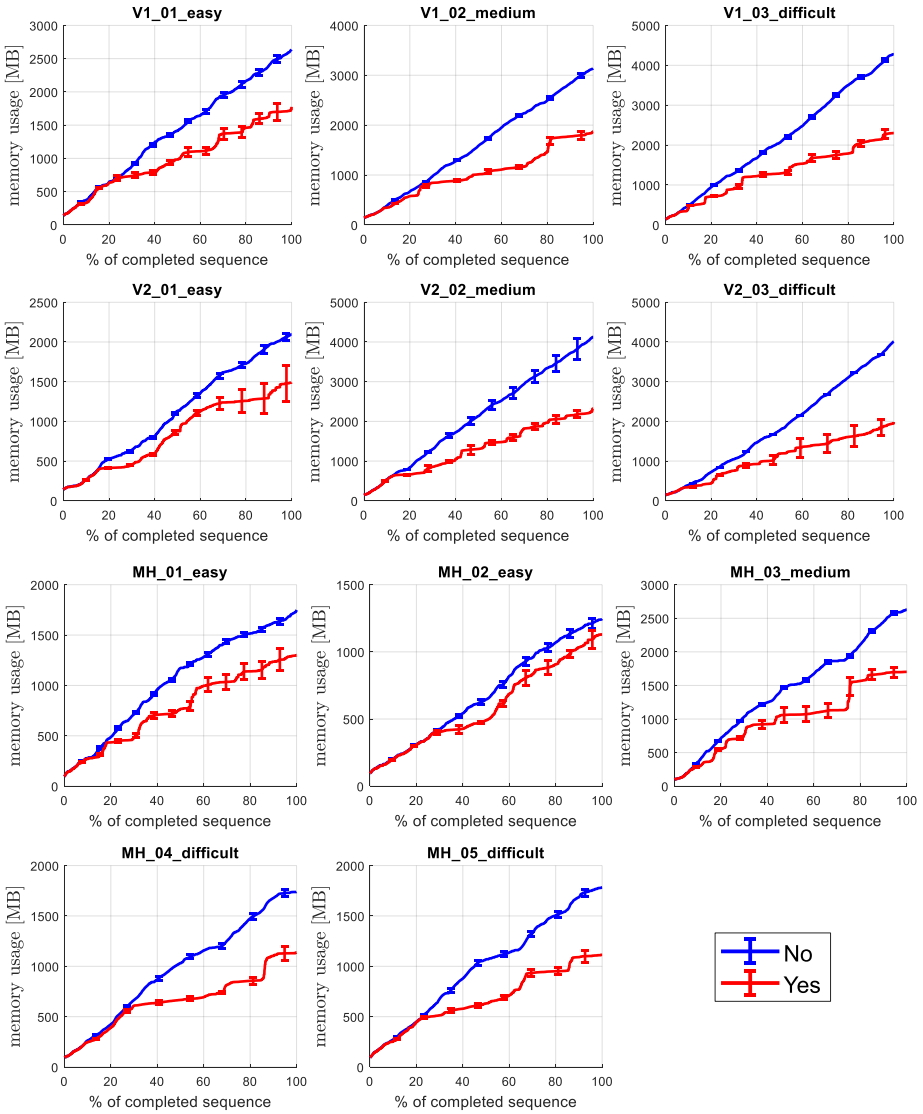


Figure 5.17: Memory usage comparison. In blue the memory usage for each sequence in normal conditions and in red when the memory pool is applied. Note how the memory requirement is reduced for all the sequences when we use the memory pool. In general, less than 1500 – 2000 MB is required, which is a reasonable number for current computers.

Memory requirements

An important concern regarding mapping approaches is the memory usage as they continuously store past information. To handle this situation, we have implemented a memory pool that allows reusing already reserved memory. This is, when a keyframe is deactivated, all its pyramidal image buffers are returned to the pool and reused by active keyframes. As a result, fixed keyframes store just a single image, the one provided by the camera.

Fig. 5.17 presents the memory requirements for all the sequences. As can be observed, the memory pool significantly reduces the memory usage for all the sequences. This is the expected behavior since we are reusing image buffers instead of reserving new memory blocks each time. However, we see that the memory usage continuously grows in time. This is due to the fact that we always create new keyframes even if we are in an already mapped area. It would be interesting to add map maintenance strategies such as removal of redundant keyframes (Mur-Artal et al., 2015) to ensure long-term operation efficiency.

5.6.3 Qualitative evaluation

Fig. 5.18 shows examples of estimated trajectories compared to the ground-truth. See how well the estimated trajectory fits the ground-truth, even in sequences where the drone is navigating in large loops as in sequences MH_04_difficult and MH_05_difficult.

Fig. 5.1 and Fig. 5.19 show some 3D maps obtained with DSM. In contrast to sliding-window based approaches, incorporating covisibility constraints avoids duplicating points and builds a consistent map. DSM estimates a precise camera trajectory and 3D reconstruction even in the most difficult sequences such as V1_03_difficult and MH_05_difficult. Note in Fig. 5.19 how there are many different objects that are easily recognizable, such as the chessboard, the ladder or the cupboard.

In addition, we also evaluate DSM in custom videos. Fig. 5.20 shows the 3D reconstruction of an office desk. It is able to recover the structure of untextured elements such as the contours of the screens, the keyboard keys, speakers and even cables. Fig. 5.21 presents some details of the same reconstruction where it is possible to easily recognize many different objects. Finally, 5.22 shows the estimated reconstruction under industrial conditions, including low texture and reflections. In this case, the boundaries of the inspection module and the

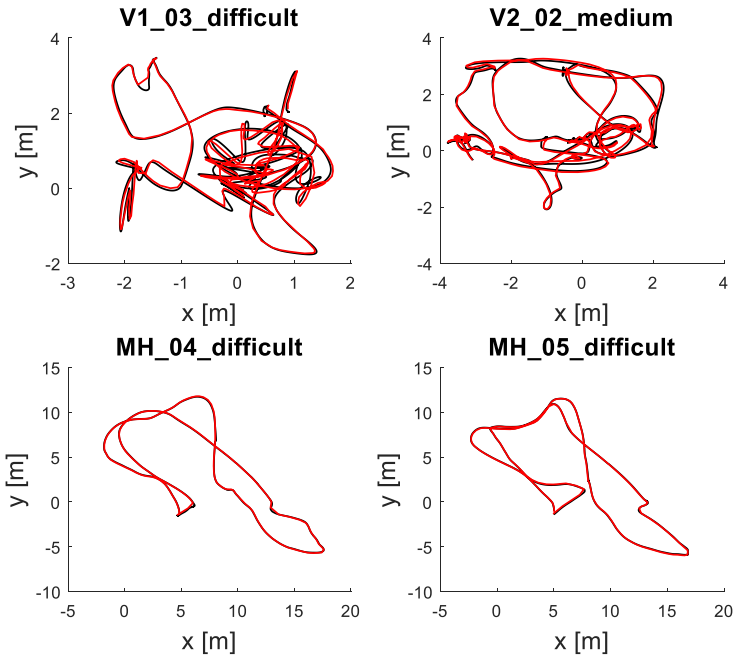


Figure 5.18: Trajectory examples by DSM (red) and ground-truth (black).

robot arm are easily identifiable which can be used to detect collisions between different mobile robotics in a production line.

5.6.4 Discussion

We have demonstrated the benefits of building a persistent map instead of just estimating the camera odometry with a temporary map. Both the accuracy of the trajectory and the reconstructed map improve by reusing map information in the photometric model. DSM manages to process scene reobservations and successfully completes 10 out of 11 sequences with an RMS ATE below 0.1m in the challenging EuRoC dataset without requiring any loop closure detection and correction.

During long-term sequences in the same environment DSM provides reliable estimates as long as point reobservations are successfully processed. As previously discussed it would be interesting to add map maintenance strategies,

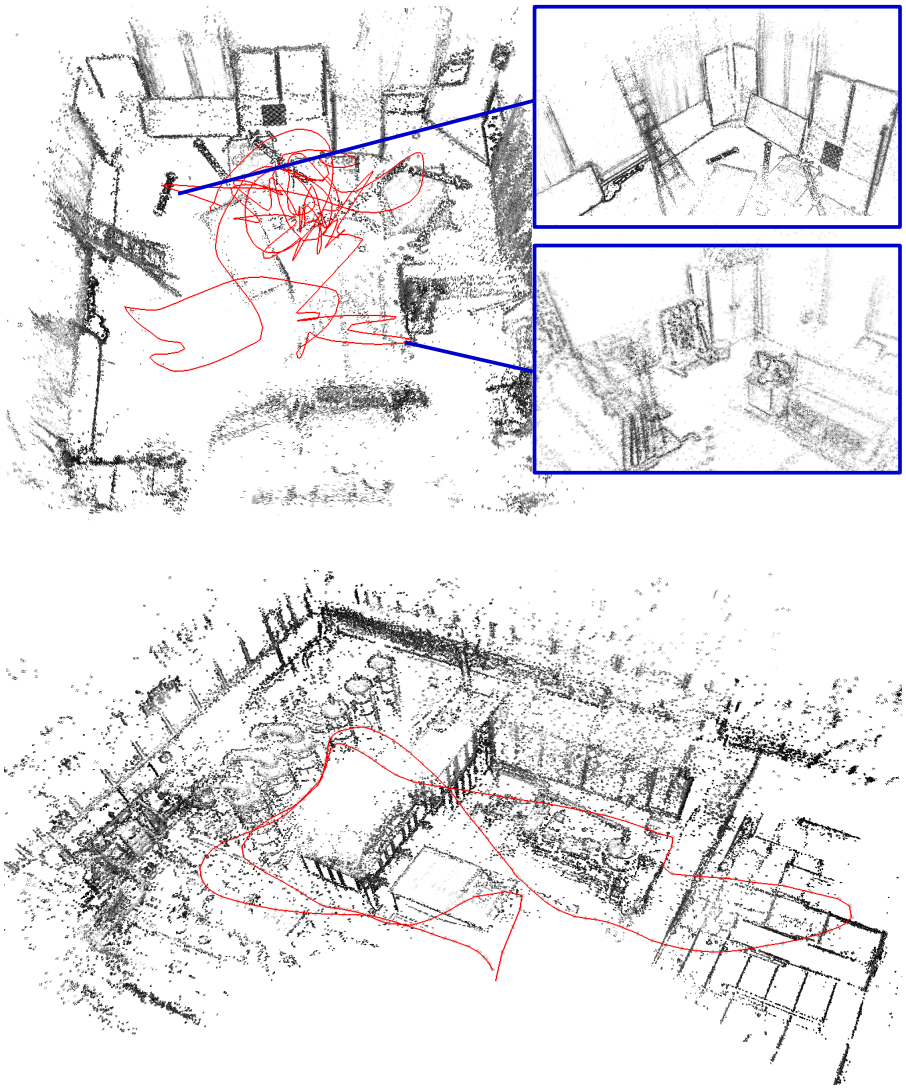


Figure 5.19: Qualitative examples. V1_03_difficult (top) and MH_05_difficult (bottom) sequences. The trajectory is displayed in red.

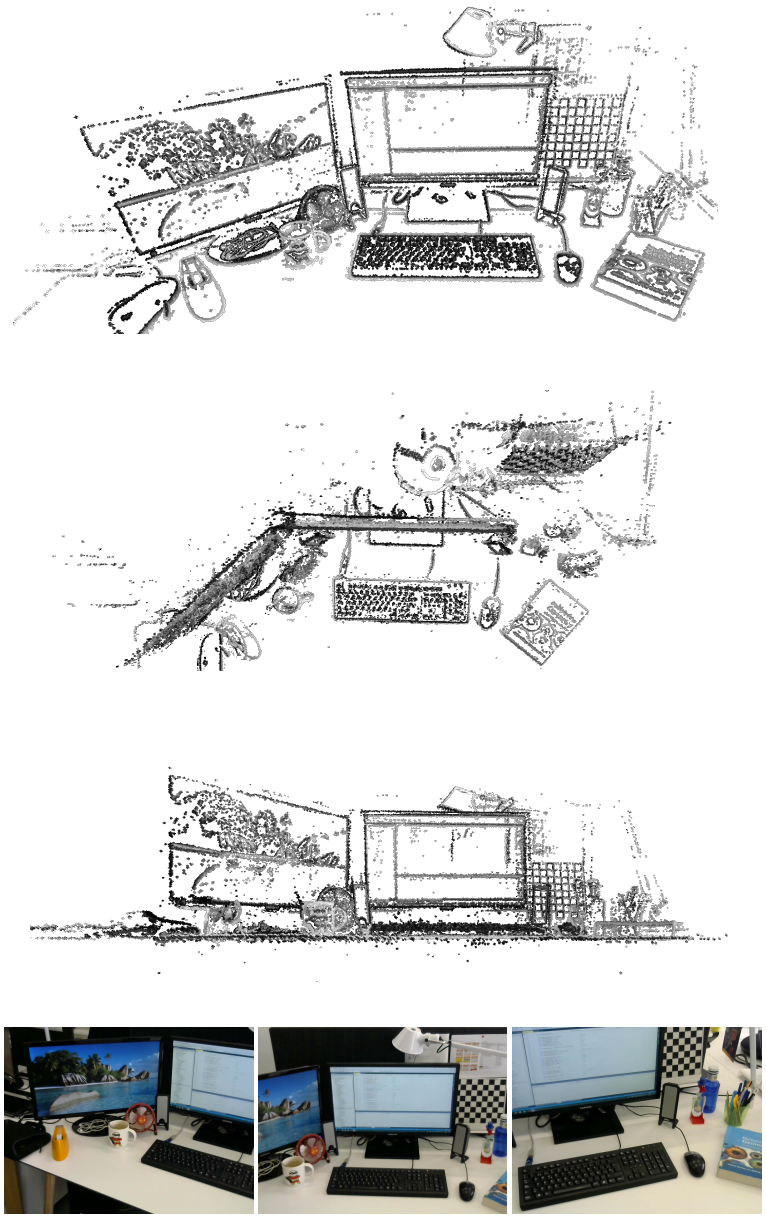


Figure 5.20: Example of the reconstruction of an office desk. Note how the objects are reconstructed with detail, such as the screens, the keyboard, the mouse and even cables.

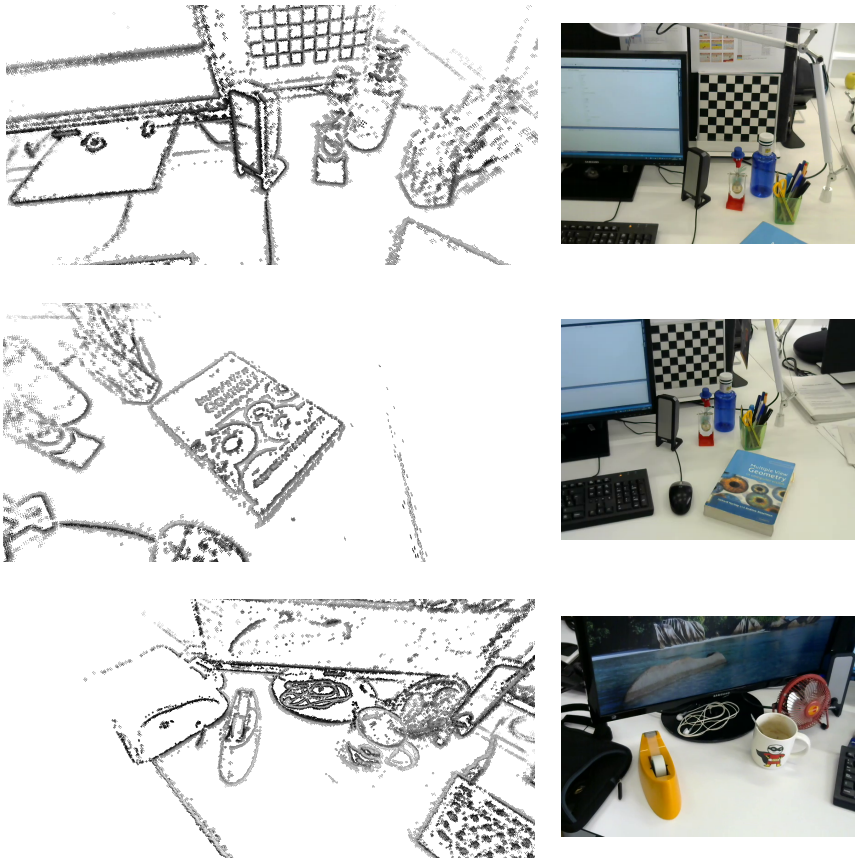


Figure 5.21: Example of reconstructed objects that are easily identifiable.

such as removal of redundant keyframes and points, for two reasons: (1) to maintain the memory requirements to the minimum; (2) to relax the covisibility graph of keyframes. The latter is very important as the number of fixed covisible keyframes increases, the graph becomes more rigid and the PBA hardly improves the estimate of active keyframes and points.

Even with a persistent map, it is not possible to handle all reobservations in all situations. In large trajectory scenarios, the accumulated drift makes it impossible to detect map point reobservations with geometric techniques alone. Sometimes map point reobservations do not even fall in the camera field of view due to the large drift, e.g. in a highway loop. In these cases, a place recognition module, which exploits the image appearance, would be useful to detect loop

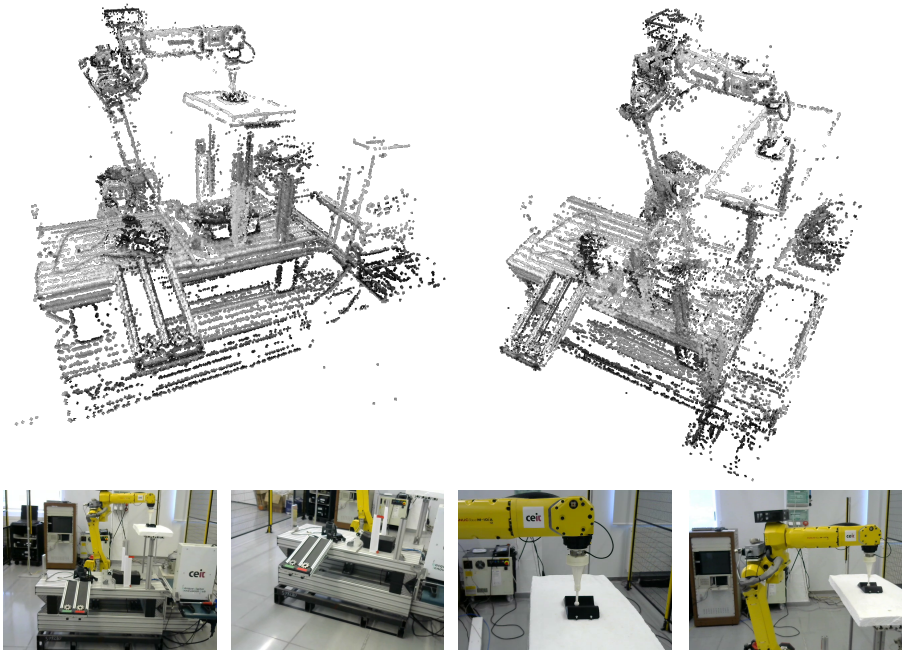


Figure 5.22: Example of the reconstruction of an industrial inspection module using a robot arm.

closures. Then, a pose-graph optimization will serve as an initialization for the PBA. Therefore, we believe that combining map reuse capabilities with a place recognition module, such as previously done with indirect techniques in (Strasdat et al., 2011) and (Mur-Artal et al., 2015), is the best alternative. In any case, we think that a pose-graph should only be used as a coarse initialization technique for the PBA, which is the optimization technique that actually exploits all the available geometric information in a VSLAM system.

Part IV

Conclusions & Future Research

Chapter 6

Conclusions and Future Work

This thesis has studied several perception techniques for augmented reality in industrial environments and mobile robotics. This chapter presents the main conclusions reached during the course of this work, as well as a number of future research lines in which this work can be extended.

6.1 Conclusions

In this thesis, we have presented two novel approaches for visual perception in two different situations. Chapter 4 has presented an object recognition and tracking method for robust pose estimation when dealing with untextured scenes of industrial environments. We have also integrated the proposed approach into a complete AR assistance tool, called ARgitu, for maintenance in industrial settings. Chapter 5 has presented a direct visual SLAM method, denoted as Direct Sparse Mapping (DSM), for estimating the camera motion and scene structure from a video stream. We have focused on developing a robust and accurate solution using direct techniques that provides reliable estimates when an autonomous mobile robot navigates continuously in the same environment.

The main contributions of each of the proposed approaches are summarized below.

Untextured object recognition for industrial environments

Markerless object recognition approaches make use of visual features in the image that are naturally in the scene. Industrial environments lack texture and

contain non-Lambertian surfaces that make texture-based approaches fail. We have proposed a new 3D object recognition method based on geometric features in order to detect and localize 3D models using a single image. More precisely, our method uses a combination of model circles and edge templates, to improve upon the weaknesses of each other.

The method improves the registration step of chamfer matching approaches using corresponding conics. Normally, the registration is performed scanning all the image by brute force. In contrast, the proposed correlation between model circles and image ellipses reduces significantly the computational cost. At the same time, it sidesteps the problem of estimating the camera location using a single circle. Instead of using just points and lines, we use the whole mode shape to solve the revolution symmetry ambiguity. Moreover, the experimental validation has demonstrated that the proposed approach is also more robust in challenging situations such as cluttered background and occlusions. This is due to the fact that image ellipses allow skipping many wrong camera location by searching just locally around them. This is an important property in industrial environments since machinery is composed of many different parts that are occluded between them.

Regarding computational time, the method can detect many kind of objects in a few hundred of milliseconds, achieving close to real-time performance in many cases and running four times faster that current state-of-the-art techniques in a standard PC. As a result, we obtain faster and more robust camera pose estimates, which is a determining factor for AR applications in order to give a correct visual feedback in a smooth manner.

In addition, we have developed an fully automatic training stage that extracts all the required geometric features from the 3D model: edges, circles and edge templates. Thus, it is possible to add new models easily in just a few minutes, which allows its direct application in the industry without requiring the participation of technicians.

Finally, the proposed method has been integrated in a complete pipeline for augmented reality applications. We present a framework to generate and present virtual and augmented information for the development of AR assistance tools in industrial settings. It includes all the tools required for the development and visualization of contents. Firstly an author tool enables users to create assistance guides for maintenance processes using different annotation types, such as text, video, virtual reality based animations and augmented reality. These guides are presented to the user using a simple to use guiding application. The framework

uses the proposed approach for recognition and localization of non-lambertian objects in industrial environments. Both applications leverage the capabilities of the proposed method, such as the simple training process, to detect and track objects in industrial environments in real-time.

Direct visual SLAM for mobile robotics

One of the core properties of a VSLAM approach is to detect when a mobile robot is traversing an already known location. To obtain this, it is required to correlate images of the same scene independently of the time and viewpoint at which the images were captured (covisibility). We have proposed DSM, a new direct VSLAM system. Up to our knowledge, DSM is the first fully direct monocular VSLAM approach that is able to detect and handle point reobservations when revisiting already mapped areas.

The system uses the same objective function and map points for all the tasks: tracking front-end, optimization back-end and map reuse. We build a persistent map by reusing map points from already visited scene regions. To obtain this, we have presented a new local window selection strategy using covisibility criteria, which enables to include map point reobservations into the photometric bundle adjustment. Instead of using feature matches as indirect approaches, the covisibility is obtained from geometric and photometric constraints. We have demonstrated that a coarse-to-fine strategy is required to process point reobservations with the photometric model due to its narrow convergence basin. The result is a system that builds a persistent map that can be reused, obtaining more accurate localization and structure estimates. In consequence, the need of loop closure detection and correction is postponed to trajectories longer than their visual odometry counterparts.

As a pure direct system, DSM does not rely on the repeatability of selected points that have been sampled across pixels with high gradient module. As a result, the system is able to work in challenging situations with a lack of textured surfaces and motion blur. At the same time, the system is able to recover the 3D geometry of contours in the scene and reuse map points avoiding duplications. The result is a more consistent, complete and dense reconstruction, which provides a rich description of the environment. This is a very important capacity for autonomous mobile robotics which allows them to understand better their surroundings in order to take the right decisions.

The exhaustive evaluation of the system in the challenging EuRoC MAV dataset has demonstrated that DSM is able to provide very accurate estimates of both the camera trajectory and map reconstruction. For the first time, we have also measured the precision of the map which is usually neglected. Our solution provides more accurate and robust localization and reconstruction estimates compared to the state-of-the-art direct VO implementations. While VO with pose-graph approaches miss many available loop closures, we continuously reuse existing map point information to provide stable estimates in time. To strengthen the proposed method, we have published the code as open-source.

6.2 Limitations and Future Research Lines

During the development of this work, we have identified a number of limitations and interesting future research lines. In the following, we present some of them to guide future studies.

1. **Object Recognition**

   The proposed method uses the 3D geometric properties of the model to recognize the target object in the image. However, it has been developed to work only with model circles and edge templates. As a consequence, it requires the target object to contain circles in its surface and relies on the performance of the ellipse detector.

   It would be interesting to incorporate a learning-based formulation to automatically extract the most useful geometric properties from each model during the training. One possible solution could be to realistically render the model in many different situations and positions (including illumination, material, texture, backgrounds, occlusions, etc.), and use Convolutional Neural Networks (CNN) to automatically learn the best model parameters for each case. Such approach would provide a coarse pose of the object that could be further refined using more traditional optimization methods. As well as the proposed approach, the result will contain an automatic training from the 3D model without the technician participation. In this way, we would obtain a more flexible algorithm that would allow to incorporate this technology in a wider range of practical applications.

   Finally, the proposed AR framework guides technicians during maintenance tasks, but it does not verify if the task has been properly executed. It would be appropriate to extend the framework with an additional module to determine the correctness of the task and provide the corresponding feedback to the worker.

2. **Visual SLAM**

   In the current implementation, the photometric bundle adjustment (PBA) is the bottleneck of the processing cost. A possible improvement could be to use SIMD instructions that permit processing the eight pixels in the patch in two steps. Moreover, there are many operation in the system

that are independent for each point and, thus, can be parallelized taking advantage of modern CPUs or even GPUs.

The system creates keyframes everytime the tracking requires new reference points, even if there are many keyframes that observe the same scene region. This is due to the fact that the tracking is performed against the latest keyframe and not against the persistent map. It would be interesting to check beforehand if there are keyframes in the map that could serve as reference for the tracking. Thus, the system would reuse keyframes and avoid creating unnecessary new ones. Another possible alternative could be to track directly against the persistent map using point observations from many different keyframes. The resulting system will be more efficient with respect to long-term operations.

Regarding with the previous limitation, it would be also interesting to add map maintenance strategies such as removal of redundant keyframes and points. The benefits of implementing this capacity will be twofold: reduce the memory footprint and reduce the stiffness of the covisibility graph. Consequently, the optimization graph will be more flexible and the PBA will have more capacity to improve the camera and point estimates. This is a very important aspect that should be taken into account, specially to achieve drift-free performance in long-term operations in the same environment.

Moreover, DSM uses geometry constraints to build the covisibility graph instead of place recognition. Thus, we cannot handle large scale loop closures nor relocalization, as we need the initial optimization seed must be near the solution. One possible alternative could be to make a hybrid solutions combining the strengths of indirect and direct approaches. However, a more interesting option could be to integrate learning-based features as an input for the direct VSLAM. Recently, von Stumberg et al. (2019) has proposed a network to train features with a large convergence basin for direct VSLAM. This extends the possibilities of developing place recognition modules using a direct formulation, which currently are dominated by indirect approaches.

It would also be interesting to extend DSM to another kind of cameras – omnidirectional or stereo – or even integrate it with other sensors such as inertial measurement units (IMU). We believe this could further boost the performance of DSM for real applications where, for example, the real scale of the scene is required.

Finally, there is still the open challenge of performing VSLAM in dynamic scenes. In fact, the real world is composed of moving objects and the assumption of large static environments is hard to be fulfilled. One possible solutions could be to recognize each of the objects so they could be tracked and reconstructed independently. At the same time, non-rigid VSLAM is also a promising technology, specially for medical applications, such as endoscopy, where human tissues come into play.

Part V

Appendices

Appendix A

Camera Models

The following sections present a review of several camera projection models providing the projection and unprojection functions when defined (see Sec. 3.3.1). We follow the works of (Devernay and Faugeras, 2001, Kannala and Brandt, 2006, Szeliski, 2010, Usenko et al., 2018).

A.1 Radial-Tangential Model

This is one of the most popular alternatives to model lense non-linear distortions and its implementation can be found in OpenCV library. It approximates the lense distortion using a polynomial of n degrees with two terms: the radial and tangential distortion. Higher degree coefficients approximate better lenses with large distortion but may become numerically unstable at the same time. The projection function with a polynomial of degree two is given by

$$\pi_r(\mathbf{x}) = \begin{bmatrix} \frac{x}{z}(1 + k_1 r^2 + k_2 r^4) + 2\rho_1 \frac{xy}{z^2} + \rho_2(r^2 + 2\frac{x^2}{z^2}) \\ \frac{y}{z}(1 + k_1 r^2 + k_2 r^4) + \rho_1(r^2 + 2\frac{y^2}{z^2}) + 2\rho_2 \frac{xy}{z^2} \end{bmatrix}, \tag{A.1}$$

$$r = \sqrt{\frac{x^2}{z^2} + \frac{y^2}{z^2}}, \tag{A.2}$$

where k_1, k_2 are the radial distortion coefficients and ρ_1, ρ_2 the tangential distortion coefficients. The projection function is defined for $\Theta = \{\mathbf{x} \in \mathbb{R}^3 \mid z > 0\}$ since it is implemented as a distortion function for pinhole-based projected points.

This model does not have a closed-form inverse. The unprojection function requires to find the root of the polynomial to recover r. It can be iteratively

obtained using the Newton's method. Alternatively, one can directly obtain the values of x/z and y/z by solving the system of equations with an iterative optimization, e.g. Gauss-Newton method. However, the latter requires to obtain the jacobians with respect to x/z and y/z which is more complex to implement.

The radial-tangenial model is well suited for cameras with low radial distortion, such as consumer cameras. In practice it has been demonstrated that the model is not suitable for wide-angle or fisheye cameras with large distortions and a field-of-view larger than a 120°.

A.2 FOV Model

The FOV model was specifically developed for fisheye cameras that are designed to include some kind of non-linear distortion. It assumes a radially symmetric distortion: the distance of an image point to the principal point is proportional to the angle between the corresponding 3D point, the optical center and the optical axis (see Fig. A.1). The corresponding projection function is given by

$$\pi_f(\mathbf{x}) = \begin{bmatrix} \frac{r_d}{r_u} x \\ \frac{r_d}{r_u} y \end{bmatrix}, \tag{A.3}$$

$$r_u = \sqrt{x^2 + y^2}, \tag{A.4}$$

$$r_d = \frac{\operatorname{atan2}(2 r_u \tan \frac{w}{2}, z)}{w}, \tag{A.5}$$

where $w \neq 0$ is the unique model parameter and represents the field-of-view of an ideal fisheye lens. The FOV model is defined for $\Theta = \{\mathbf{x} \in \mathbb{R}^3 \setminus [0,0,0]^T\}$.

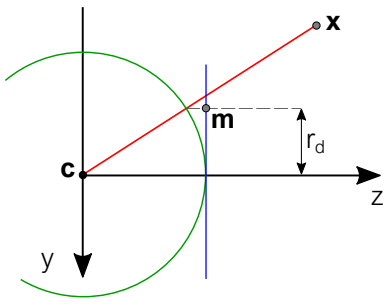


Figure A.1: Geometric representation of the FOV camera model.

The FOV model has a closed-form solution for the unprojection function defined as

$$\pi_f^{-1}(\mathbf{m}) = \begin{bmatrix} \frac{\tan(r_d w)}{2 r_d \tan\frac{w}{2}} m_u \\ \frac{\tan(r_d w)}{2 r_d \tan\frac{w}{2}} m_v \\ 1 \end{bmatrix}, \tag{A.6}$$

$$r_d = \sqrt{m_u^2 + m_v^2}. \tag{A.7}$$

A.3 Equidistant Model

The equidistant model is a generic camera model and has been demonstrated to fit well regular, wide-angle and fisheye lenses. It assumes that the distance from the optical center of the image to the projected point is proportional to a polynomial of the angle between the point and the principal axis (see Fig. A.2). The projection function with a polynomial of four parameters is given by

$$\pi_e(\mathbf{x}) = \begin{bmatrix} \frac{d(\theta)}{r} x \\ \frac{d(\theta)}{r} y \end{bmatrix}, \tag{A.8}$$

$$r = \sqrt{x^2 + y^2}, \tag{A.9}$$

$$\theta = \text{atan2}(r, z), \tag{A.10}$$

$$d(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9, \tag{A.11}$$

where k_1, k_2, k_3, k_4 are the model parameters. The function is defined for $\Theta = \{\mathbf{x} \in \mathbb{R}^3 \setminus [0,0,0]^T\}$.

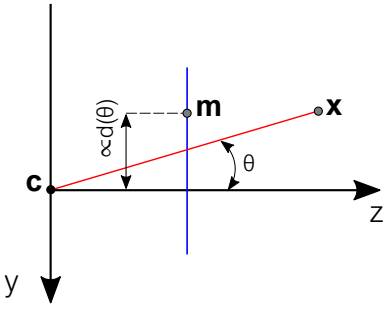


Figure A.2: Geometric representation of the Equidistant camera model.

The equidistant model does not have a closed-form inverse. The unprojection function requires an iterative optimization to solve the inverse of the polynomial, such as the ones proposed in the radial-tangential model. Moreover, this model can be found as a distortion model for the pinhole camera, e.g. the fisheye camera model in OpenCV. However, as explained before, implemented in this manner it has a singularity at $z = 0$, which makes it unsuitable for lenses with a field-of-view close to 180°.

A.4 Extended Unified Model

The extended unified camera model (EUCM) is a generalization of the unified camera model (UCM) that is widely used with catadioptric cameras. This model first projects the 3D point onto a symmetric ellipsoid around the z axis and then onto the image plane using the pinhole model shifted by $\frac{\alpha}{1-\alpha}$ (see Fig. A.3). A major advantage over the previous models is that the EUCM has a closed-form solution for projection and unprojection functions, and does not require computationally expensive trigonometric operations. The projection function is defined as

$$\pi_u(\mathbf{x}) = \begin{bmatrix} \frac{x}{\alpha d + (1-\alpha) z} \\ \frac{y}{\alpha d + (1-\alpha) z} \end{bmatrix}, \tag{A.12}$$

$$d = \sqrt{\beta (x^2 + y^2) + z^2}, \tag{A.13}$$

where $\alpha \in [0, 1]$, $\beta > 0$ are the model parameters. Note that for $\alpha = 0$ and $\beta = 1$ the model degrades to the pinhole model. The EUCM is defined for:

$$\Theta = \{\mathbf{x} \in \mathbb{R}^3 \mid z > -wd\}, \tag{A.14}$$

$$w = \begin{cases} \frac{\alpha}{1-\alpha} & \text{if } \alpha \leq 0.5, \\ \frac{1-\alpha}{\alpha} & \text{otherwise.} \end{cases} \tag{A.15}$$

The EUCM has a closed-form inverse function defined as

$$\pi_u^{-1}(\mathbf{m}) = \frac{1}{\sqrt{m_u^2 + m_v^2 + \lambda^2}} \begin{bmatrix} m_u \\ m_v \\ \lambda \end{bmatrix}, \tag{A.16}$$

$$\lambda = \frac{1 - \beta \alpha^2 r^2}{\alpha \sqrt{1 - (2\alpha - 1)\beta r^2} + (1 - \alpha)}, \tag{A.17}$$

$$r^2 = m_u^2 + m_v^2. \tag{A.18}$$

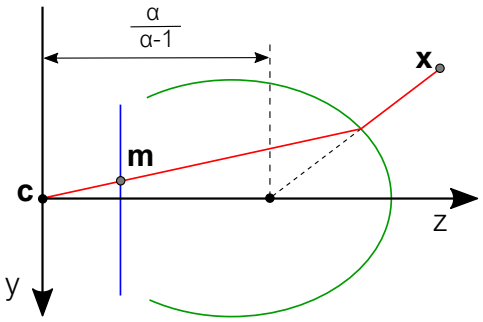


Figure A.3: Geometric representation of the EUCM.

A.5 Double Sphere Model

The double sphere (DS) model was developed to fit fisheye lenses. The DS model first projects the 3D point onto two consecutive spheres, which are shifted by ξ. Then, the point is projected onto the image plane using the pinhole model shifted by $\frac{\alpha}{1-\alpha}$ (see Fig. A.4). Similar to the EUCM, the DS model has a closed-form solution for projection and unprojection functions, and does not require computationally expensive trigonometric operations. The projection function is given by

$$\pi_d(\mathbf{x}) = \begin{bmatrix} \frac{x}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \\ \frac{y}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \end{bmatrix}, \tag{A.19}$$

$$d_1 = \sqrt{x^2 + y^2 + z^2}, \tag{A.20}$$

$$d_2 = \sqrt{x^2 + y^2 + (\xi d_1 + z)^2}, \tag{A.21}$$

where ξ, α are the model parameters. The projection function is defined for:

$$\Theta = \{\mathbf{x} \in \mathbb{R}^3 \mid z > -w_2 d_1\}, \tag{A.22}$$

$$w_2 = \frac{w_1 + \xi}{\sqrt{2 w_1 \xi + \xi^2 + 1}} \tag{A.23}$$

$$w_1 = \begin{cases} \frac{\alpha}{1-\alpha} & \text{if } \alpha \leq 0.5, \\ \frac{1-\alpha}{\alpha} & \text{otherwise.} \end{cases} \tag{A.24}$$

The unprojection function is defined as

$$\pi_f^{-1}(\mathbf{m}) = \frac{\lambda\xi + \sqrt{\lambda^2 + (1-\xi^2)r^2}}{\lambda^2 + r^2} \begin{bmatrix} m_u \\ m_v \\ \lambda \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix}, \tag{A.25}$$

$$\lambda = \frac{1 - \alpha^2 r^2}{\alpha\sqrt{1 - (2\alpha - 1)r^2} + (1 - \alpha)}, \tag{A.26}$$

$$r^2 = m_u^2 + m_v^2. \tag{A.27}$$

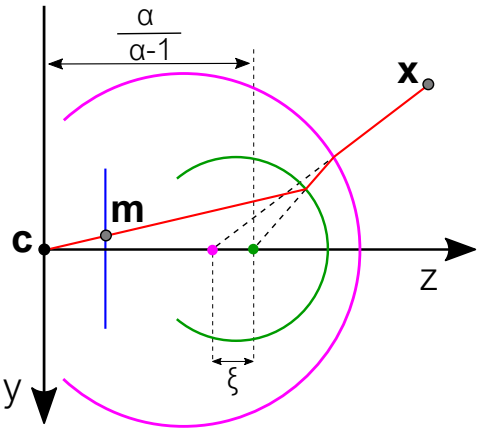


Figure A.4: Geometric representation of the DS camera model.

Appendix B

Conic Based Pose Estimation

This appendix presents how to estimate the pose of a camera using conics. We follow the method proposed in (De Ma, 1993), which shows that at least two conics are required to estimate the pose of a 3D object. First, we present the general geometric constraint that is obtained when projecting a 3D conic into an image. Then, we show how to obtain the camera pose in two cases: when the conic is an ellipse and a circle. We assume the image to be undistorted and use the pinhole camera projection model (see Sec. 3.3).

B.1 The Basic Geometric Constraint

Let $\mathbf{Q}$ be a conic that lies on a plane with a local coordinate system attached to it, in which the x-axis and y-axis lie on the plane and z-axis is normal. Any point $\mathbf{x}$ in the conic is projected into an image pixel $\mathbf{u}$ as

$$\mathbf{u} = \mathbf{K} \cdot \pi(\mathbf{R} \cdot \mathbf{x} + \mathbf{t}), \tag{B.1}$$

where $\mathbf{T} = [\mathbf{R} \mid \mathbf{t}]$ corresponds to the camera pose. Fig. B.1 shows an schematic representation of the problem.

At the same time, any projected point $\mathbf{u}$ is also part of a conic in the image and, thus, it can be represented in matrix form using the Eq. 3.9 as

$$\bar{\mathbf{u}}^T \cdot \mathbf{G} \cdot \bar{\mathbf{u}} = 0. \tag{B.2}$$

Without loss of generality, the pose of an ellipse lying in the object surface can be simplified using only the first two columns of the rotation matrix $\mathbf{R}$ and

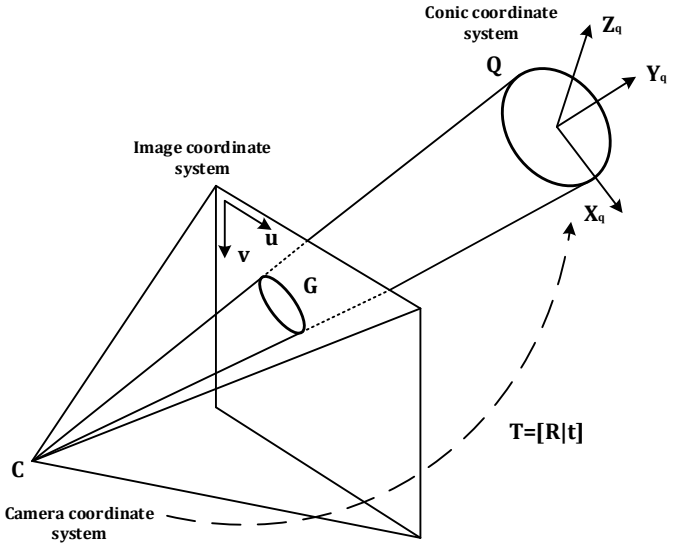


Figure B.1: 3D conic and camera representation in space. Any point in the local coordinates of a conic can be represented in camera coordinates using the transformation $\mathbf{T}$. The projection of a conic $\mathbf{Q}$ in the space is expressed as the matrix $\mathbf{G}$ in image pixels.

the translation vector $\mathbf{t}$ as $\mathbf{P} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{t})$. Using $\mathbf{P}$ and substituting the Eq. B.1 into the Eq. B.2 we get

$$\bar{\mathbf{u}}^T \cdot \mathbf{G} \cdot \bar{\mathbf{u}} = \mathbf{x}^T \cdot \mathbf{P}^T \cdot \mathbf{K}^T \cdot \mathbf{G} \cdot \mathbf{K} \cdot \mathbf{P} \cdot \mathbf{x} = 0, \tag{B.3}$$

which is the same conic representation as in Eq. 3.9. Both expressions are equal up to a scale factor, k:

$$\mathbf{P}^T \cdot \mathbf{K}^T \cdot \mathbf{G} \cdot \mathbf{K} \cdot \mathbf{P} = \mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P} = k \cdot \mathbf{Q}, \tag{B.4}$$

where $\mathbf{A} = \mathbf{K}^T \cdot \mathbf{G} \cdot \mathbf{K}$.

The Eq. B.4 is the basic geometric constraint of a conic and its projection. In the following sections, the pose estimation from corresponding conics is solved using this constraint. We study first the case of a single ellipse and then the case of a single circle.

B.2 Pose from a single ellipse

The problem is to determine the pose of a 3D object knowing the correspondence between an ellipse in the surface of the object and its projection in an image. As we will see, it is not possible to determine a unique pose from only one correspondence and more information is required, such as points (Costa and Shapiro, 2000), lines (Wang et al., 2008) or as in our proposed method, edge templates (see Chapter 4).

The translation vector can be expressed using the rotation coefficients as $\mathbf{t} = \mathbf{R} \cdot \mathbf{c}$ where $\mathbf{c} = (c_1, c_2, c_3)$. Therefore, the pose matrix $\mathbf{P}$ can be rewritten introducing the new vector $\mathbf{c}$ as

$$\mathbf{P} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}) = \mathbf{R} \cdot \mathbf{C} = \mathbf{R} \begin{bmatrix} 1 & 0 & c_1 \\ 0 & 1 & c_2 \\ 0 & 0 & c_3 \end{bmatrix}. \tag{B.5}$$

Introducing the new expression of the pose to the equation B.4 we get:

$$\mathbf{R}^T \cdot \mathbf{A} \cdot \mathbf{R} = k \cdot \mathbf{B}, \tag{B.6}$$

where $\mathbf{B} = (\mathbf{C}^T)^{-1} \cdot \mathbf{Q} \cdot \mathbf{C}^{-1}$. From the above expression, we observe that the matrices $\mathbf{A}$ and $\mathbf{B}$ are similar and, therefore, they have the same eigenvalues. Then, using the invariants of a 3×3 symmetric tensor we obtain three equations in terms of the ellipse parameters:

$$\det(k\mathbf{B}) = \lambda_1\lambda_2\lambda_3 = \frac{-k^3}{c_3^2 a^2 b^2}, \tag{B.7}$$

$$\begin{aligned} \frac{1}{2}[tr(k\mathbf{B})^2 - tr(k\mathbf{B}^2)] &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \\ &= \frac{k^2(d^2 - a^2 - b^2)}{c_3^2 a^2 b^2}, \end{aligned} \tag{B.8}$$

$$\begin{aligned} tr(k\mathbf{B}) &= \lambda_1 + \lambda_2 + \lambda_3 \\ &= \frac{k(d^2a^2 + c_1^2b^2 - c_1a^2 + c_3^2b^2 - a^2b^2)}{c_3^2 a^2 b^2}, \end{aligned} \tag{B.9}$$

where λ_1,λ_2 and λ_3 are the three eigenvalues of $\mathbf{A}$ and d is the absolute distance between the center of the ellipse and the origin of the camera coordinate system,

$d^2 = c_1^2 + c_2^2 + c_3^2$. Notice that it is a three equation system with four unknowns (c_1, c_2, c_3, k) so we require additional information to solve it. One possible solution is to know the absolute distance d and solve the system to obtain the values of (c_1, c_2, c_3, k). Now we can solve for $\mathbf{R}$ and $\mathbf{t}$.

Substituting the parameter values in the Eq. B.6, we rewrite the expression as

$$\mathbf{R}^T \cdot \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{U}^{-1} \cdot \mathbf{R} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{V}^{-1}, \tag{B.10}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the matrices whose columns contain the eigenvectors of $\mathbf{A}$ and $\mathbf{B}$ respectively, and $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{A}$.

Now denoting $\mathbf{W} = \mathbf{U}^{-1} \cdot \mathbf{R} \cdot \mathbf{V}$ and using the Eq. B.10, we conclude that $\mathbf{W}$ has eight possible solutions give by

$$\mathbf{W} = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}. \tag{B.11}$$

If we consider that $det(\mathbf{U}) = 1$ and $det(\mathbf{V}) = 1$, W has only four solutions since $det(\mathbf{W}) = 1$, in the case of a dextrorotation solution. However, in general we cannot guarantee that the determinants of the eigenmatrices are equal to one. Thus, we must evaluate all possible values of the $\mathbf{W}$ matrix (dextrorotation and levorotation cases) and discard the levorotation solutions ($det(\mathbf{R}) = -1$) . Finally, we can determine $\mathbf{R}$ and $\mathbf{t}$ with

$$\begin{aligned} \mathbf{R} &= \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^{-1}, \\ \mathbf{t} &= \mathbf{R} \cdot \mathbf{c}. \end{aligned} \tag{B.12}$$

There are 8 possible combinations for the signs of c_1, c_2 and c_3. A total of 4 solutions for $\mathbf{R}$ depending on the signs of the matrix $\mathbf{W}$. This yields to a total of 32 possible solutions. However, we know that the model is in front of the camera which reduces the number to 16. Finally, setting the z axis normal to the ellipse and the x axis aligned with the major semi-axis correspond to only 4 different poses (see Fig. B.2). Thus, as we have previously mentioned, in order to estimate a unique pose further information is required.

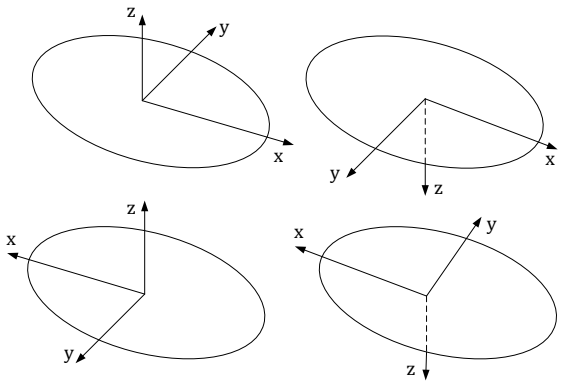


Figure B.2: Four equivalent solutions of a conic

B.3 Pose from a single circle

When the conic on the surface of the model corresponds to a circle the values of the major and minor semi-axes are equal, this is, $a = b = r$. Reasoning in the same way as the previous section, we obtain these three equations:

$$\det(k\mathbf{B}) = \lambda_1\lambda_2\lambda_3 = \frac{-k^3}{c_3^2 r^4}, \tag{B.13}$$

$$\begin{aligned}\frac{1}{2}[Tr(k\mathbf{B})^2 - Tr(k\mathbf{B}^2)] &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \\ &= \frac{k^2(d^2 - 2r^2)}{c_3^2 r^4},\end{aligned} \tag{B.14}$$

$$Tr(k\mathbf{B}) = \lambda_1 + \lambda_2 + \lambda_3 = \frac{k(d^2 + c_3^2 - r^2)}{c_3^2 r^4}. \tag{B.15}$$

In this case, we only have three unknowns d, c_3 and k and we can obtain a solution without any extra information. Therefore, c_1 and c_2 can have any value that verifies the equation of the absolute distance d. Once the unknowns are solved, we can obtain $\mathbf{R}$ and $\mathbf{t}$ as in the previous case. Furthermore, this method has also 4 possible poses (see Fig. B.2) and for each one the x-axis and y-axis axes are not uniquely determined (depending on the values of c_1 and c_2). This is due to the fact that circles do not contain major and minor axes, which creates an ambiguity.

Appendix C

Jacobians

This appendix presents the analytical derivatives of the photometric model used in Chapter 5. Many of the expressions are elaborated in a general manner so they can be used in many other computer vision and robotics applications. We first estimate the partial derivatives with respect to individual parts of the photometric equation: projection and geometric parameters. Then we use the chain rule to obtain the final jacobians for the photometric bundle adjustment.

The photometric residual is defined in the Eq. 5.2.1 as

$$r = (I_i[\mathbf{u}_i] - b_i) - \frac{e^{a_i}}{e^{a_j}}(I_j[\mathbf{u}_j] - b_j), \tag{C.1}$$

where I_i, I_j represent the images, $\mathbf{u}_i, \mathbf{u}_j$ the evaluated pixels in each corresponding image and a_i, b_i, a_j, b_j the brightness transfer function parameters. The transformation of the pixel $\mathbf{u}_i$ with a known inverse depth ρ_i from the image I_i to the image I_j is given by the Eq. 3.52 as

$$\mathbf{u}_j = \mathbf{K} \cdot \pi(\mathbf{x}'_j) \tag{C.2}$$

with

$$\mathbf{x}'_j = \rho_i \cdot \mathbf{x}_j = \mathbf{R} \cdot \mathbf{K}^{-1}\mathbf{u}_i + \rho_i \cdot \mathbf{t}, \tag{C.3}$$

that represents the scaled point transformation with the linear inverse depth mapping function of Sec. 3.4.2.

C.1 Projection Function

The jacobian of the projection function (Eq. C.2) on a pinhole camera (Eq. 3.38 and 3.42),

$$\mathbf{u} = \mathbf{K}\pi(\mathbf{x}') = \begin{bmatrix} f_u \frac{x'}{z'} + c_u \\ f_v \frac{y'}{z'} + c_v \\ 1 \end{bmatrix}, \tag{C.4}$$

is given by

$$\frac{\partial \mathbf{u}}{\partial \mathbf{x}'} = \frac{1}{z'} \cdot \begin{bmatrix} f_u & 0 & -f_u \frac{x'}{z'} \\ 0 & f_v & -f_v \frac{y'}{z'} \end{bmatrix}. \tag{C.5}$$

C.2 Camera Pose

Following the Eq. C.3, we can rewrite the scaled transformation of a point as

$$\bar{\mathbf{x}}'_j = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{K}^{-1}\mathbf{u}_i \\ \rho_i \end{bmatrix} = \mathbf{T}_{ji} \cdot \begin{bmatrix} \mathbf{K}^{-1}\mathbf{u}_i \\ \rho_i \end{bmatrix} = \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}'_i, \tag{C.6}$$

which depends on the world position of each of the cameras $\mathbf{T}_{ji} = \mathbf{T}_j^{-1}\mathbf{T}_i$.

As discussed in Sec. 3.2.3, the jacobian with respect to a pose is calculated using a small increment $\boldsymbol{\xi}$. The partial derivatives of $\mathbf{x}'_j$ with respect to $\mathbf{T}_i$ around zero are calculated employing the Eq. 3.36. In this case, we are working with the scaled case so the final expression is also scaled as

$$\begin{aligned}
\left.\frac{\partial \mathbf{x}'_j}{\partial \boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=0} &= \left.\frac{\partial(\rho_i \cdot \mathbf{x}_j)}{\partial \boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=0} \\
&= \rho_i \cdot \left.\frac{\partial(\mathbf{T}_j^{-1} \cdot exp(\boldsymbol{\xi}_i) \cdot \mathbf{T}_i \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=0} \\
&= \rho_i \cdot \left.\frac{\partial(exp(Adj_{T_j^{-1}} \cdot \boldsymbol{\xi}_i) \cdot \mathbf{T}_j^{-1} \cdot \mathbf{T}_i \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=0} \\
&= \rho_i \cdot \left.\frac{\partial(exp(\boldsymbol{\xi}'_i) \cdot \mathbf{T}_{ji} \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}'_i}\right|_{\boldsymbol{\xi}'_i=0} \left.\frac{\partial exp(Adj_{T_j^{-1}} \cdot \boldsymbol{\xi}_i)}{\partial \boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=0} \\
&= \rho_i \cdot \begin{bmatrix} \mathbf{I}_3 & -\hat{\mathbf{x}}_j \end{bmatrix} \cdot Adj_{T_j^{-1}}.
\end{aligned} \tag{C.7}$$

Similarly, the partial derivatives with respect to $\mathbf{T}_j$ around zero are given by

$$\begin{aligned}
\left.\frac{\partial \mathbf{x}'_j}{\partial \boldsymbol{\xi}_j}\right|_{\boldsymbol{\xi}_j=0} &= \rho_i \cdot \left.\frac{\partial((exp(\boldsymbol{\xi}_j)\cdot \mathbf{T}_j)^{-1}\cdot \mathbf{T}_i \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}_j}\right|_{\boldsymbol{\xi}_j=0} \\
&= \rho_i \cdot \left.\frac{\partial(\mathbf{T}_j^{-1}\cdot exp(-\boldsymbol{\xi}_j)\cdot \mathbf{T}_i \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}_j}\right|_{\boldsymbol{\xi}_j=0} \\
&= \rho_i \cdot \left.\frac{\partial(exp(-Adj_{T_j^{-1}}\cdot \boldsymbol{\xi}_j)\cdot \mathbf{T}_j^{-1}\cdot \mathbf{T}_i \cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}_j}\right|_{\boldsymbol{\xi}_j=0} \\
&= \rho_i \cdot \left.\frac{\partial(exp(\boldsymbol{\xi}'_j)\cdot \mathbf{T}_{ji}\cdot \bar{\mathbf{x}}_i)}{\partial \boldsymbol{\xi}'_j}\right|_{\boldsymbol{\xi}'_j=0} \left.\frac{\partial exp(-Adj_{T_j^{-1}}\cdot \boldsymbol{\xi}_j)}{\partial \boldsymbol{\xi}_j}\right|_{\boldsymbol{\xi}_j=0} \\
&= -\rho_i \cdot \begin{bmatrix} \mathbf{I}_3 & -\hat{\mathbf{x}}_j \end{bmatrix} \cdot Adj_{T_j^{-1}}.
\end{aligned} \tag{C.8}$$

Note that the partial derivatives with respect to one camera are the opposite to the other camera. Consequently, during numerical optimizations we only require to compute just one jacobian, greatly reducing the computational load.

C.3 Point Inverse Depth

The jacobian of the scaled point transformation with respect to the point inverse depth is given by

$$\frac{\partial \mathbf{x}'_j}{\partial \rho_i} = \frac{\partial(\mathbf{R}\cdot \mathbf{K}^{-1}\mathbf{u}_i + \rho_i \cdot \mathbf{t})}{\partial \rho_i} = \mathbf{t}. \tag{C.9}$$

C.4 Photometric Bundle Adjustment

This section presents all the required analytic derivatives to implement the photometric bundle adjustment of Chapter 5. To obtain them, we will use the expressions of the partial derivatives obtained in previous sections and use the chain rule to estimate the complete jacobians.

C.4.1 Geometric parameters

The jacobians of the photometric residual with respect to the geometric parameters are given by:

Host camera pose

$$\begin{aligned}\frac{\partial r}{\partial \boldsymbol{\xi}_i} &= -\frac{e^{a_i}}{e^{a_j}} \cdot \frac{\partial I_j}{\partial \mathbf{u}_j} \cdot \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}'_j} \cdot \frac{\partial \mathbf{x}'_j}{\partial \boldsymbol{\xi}_i} \\ &= -\frac{e^{a_i}}{e^{a_j}} \cdot \begin{bmatrix} g_x \\ g_y \end{bmatrix} \cdot \frac{\rho_i}{z'} \cdot \begin{bmatrix} f_u & 0 & -f_u \frac{x'}{z'} \\ 0 & f_v & -f_v \frac{y'}{z'} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_3 & -\hat{\mathbf{x}}_j \end{bmatrix} \cdot Adj_{T_j^{-1}} \\ &= -\frac{e^{a_i}}{e^{a_j}} \cdot \begin{bmatrix} g_x \\ g_y \end{bmatrix} \cdot \begin{bmatrix} f_u\rho_j & 0 & -f_u \frac{x'}{z'}\rho_j & -f_u \frac{x'y'}{z'^2} & f_u(1+\frac{x'^2}{z'^2}) & -f_u \frac{y'}{z'} \\ 0 & f_v\rho_j & -f_v \frac{y'}{z'}\rho_j & -f_v(1+\frac{y'^2}{z'^2}) & -f_v \frac{x'y'}{z'^2} & f_v \frac{x'}{z'} \end{bmatrix} \cdot Adj_{T_j^{-1}}\end{aligned} \tag{C.10}$$

Target camera pose

$$\begin{aligned}\frac{\partial r}{\partial \boldsymbol{\xi}_j} &= -\frac{e^{a_i}}{e^{a_j}} \cdot \frac{\partial I_j}{\partial \mathbf{u}_j} \cdot \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}'_j} \cdot \frac{\partial \mathbf{x}'_j}{\partial \boldsymbol{\xi}_j} \\ &= \frac{e^{a_i}}{e^{a_j}} \cdot \frac{\partial I_j}{\partial \mathbf{u}_j} \cdot \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}'_j} \cdot \frac{\partial \mathbf{x}'_j}{\partial \boldsymbol{\xi}_i} \\ &= -\frac{\partial r}{\partial \boldsymbol{\xi}_i}\end{aligned} \tag{C.11}$$

Point inverse depth

$$\begin{aligned}\frac{\partial r}{\partial \rho_i} &= -\frac{e^{a_i}}{e^{a_j}} \cdot \frac{\partial I_j}{\partial \mathbf{u}_j} \cdot \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}'_j} \cdot \frac{\partial \mathbf{x}'_j}{\partial \rho_i} \\ &= -\frac{e^{a_i}}{e^{a_j}} \cdot \begin{bmatrix} g_x \\ g_y \end{bmatrix} \cdot \frac{1}{z'} \cdot \begin{bmatrix} f_u & 0 & -f_u \frac{x'}{z'} \\ 0 & f_v & -f_v \frac{y'}{z'} \end{bmatrix} \cdot \mathbf{t} \\ &= -\frac{e^{a_i}}{e^{a_j}} \cdot \begin{bmatrix} g_x \\ g_y \end{bmatrix} \cdot \frac{1}{z'} \cdot \begin{bmatrix} f_u(t_x - \frac{x'}{z'}t_z) \\ f_v(t_y - \frac{y'}{z'}t_z) \end{bmatrix}\end{aligned} \tag{C.12}$$

C.4.2 Photometric parameters

The partial derivatives of the photometric residual (Eq. C.1) with respect to the brightness transfer function parameters are:

Host camera affine light

$$\begin{aligned} \frac{\partial r}{\partial a_i} &= -\frac{e^{a_i}}{e^{a_j}}(I_j[\mathbf{u}_j] - b_j) \\ \frac{\partial r}{\partial b_i} &= -1 \end{aligned} \tag{C.13}$$

Target camera affine light

$$\begin{aligned} \frac{\partial r}{\partial a_j} &= \frac{e^{a_i}}{e^{a_j}}(I_j[\mathbf{u}_j] - b_j) \\ \frac{\partial r}{\partial b_j} &= \frac{e^{a_i}}{e^{a_j}} \end{aligned} \tag{C.14}$$

Appendix D

Student's t-distribution

The Student's t-distribution is a continuous probability distribution. It is widely used in applied statistics and maximum likelihood estimation. It can be represented using 3 parameters: location μ, scale σ and degrees of freedom ν. The probability density function is given by

$$P(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}\sigma}\left(1 + \frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right)^{-\frac{\nu+1}{2}} \tag{D.1}$$

where $\Gamma(x) = (x-1)!$ is the gamma function.

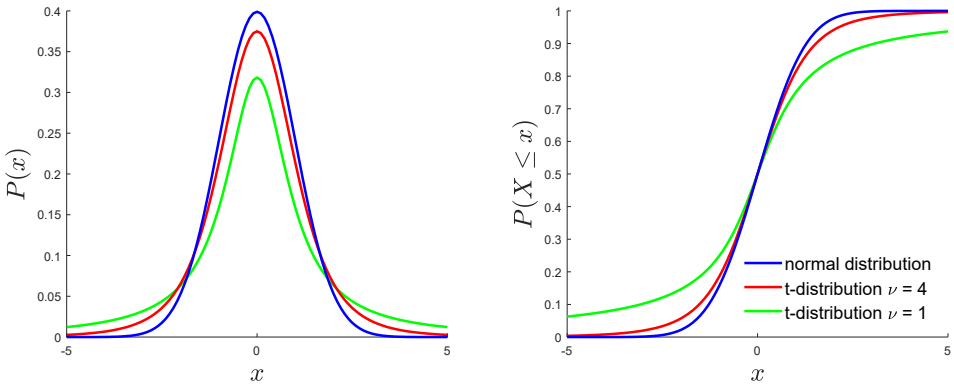


Figure D.1: Student's t-distribution. The probability density function on the left and the cumulative distribution function on the right for different values of degrees of freedom. The normal distribution is also represented for comparison.

The Student's t-distribution is symmetric and bell-shaped. It is similar to the normal distribution with heavier tails. This means that it models behaviours that tend to produce values far from the mean. Fig. D.1 shows the probability density and the cumulative distribution functions for different values of degrees of freedom. As can be observed, the Student's t-distribution becomes closer to the normal distribution as ν increases.

The parameters for a given distribution of values can be obtained by their maximum likelihood estimators (Liu and Rubin, 1995). Alternatively, one can minimize the negative log-likelihood of the probability density function (Eq. D.1) using a gradient free optimizer, such as the Nelder-Mead method (Gao and Han, 2012, Lagarias, Jeffrey et al., 1998, Singer and Singer, 2004). In both situations, the parameters are iteratively obtained starting from an initial guess. However, we found that minimizing the negative log-likelihood achieves better parameters estimates, specially when the degrees of freedom are jointly estimated.

Bibliography

Ababsa, F. and Mallem, M. Robust camera pose estimation combining 2D/3D points and lines tracking. *IEEE International Symposium on Industrial Electronics*, pages 774–779, 2008. doi: 10.1109/ISIE.2008.4676964.

Agarwal, S., Mierle, K., and Others. Ceres Solver.

Akinlar, C. and Topal, C. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011. ISSN 01678655. doi: 10.1016/j.patrec.2011.06.001.

Akinlar, C. and Topal, C. EDCircles: A real-time circle detector with a false detection control. *Pattern Recognition*, 46(3):725–740, 2013. ISSN 00313203. doi: 10.1016/j.patcog.2012.09.020.

Álvarez, H. and Borro, D. Junction assisted 3D pose retrieval of untextured 3D models in monocular images. *Computer Vision and Image Understanding*, 117 (10):1204–1214, 2013. ISSN 10773142. doi: 10.1016/j.cviu.2012.08.012.

Álvarez, H., Aguinaga, I., and Borro, D. Providing guidance for maintenance operations using automatic markerless augmented reality system. *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 181–190, 2011. doi: 10.1109/ISMAR.2011.6092385.

Ayad, M. S., Lee, J., Deguet, A., Burdette, E. C., and Prince, J. L. C-arm pose estimation using a set of coplanar ellipses in correspondence. In *2010 7th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, ISBI 2010 - Proceedings*, pages 1401–1404, 2010. ISBN 9781424441266. doi: 10.1109/ISBI.2010.5490260.

Babu, B. W., Kim, S., Yan, Z., and Ren, L. σ-DVO: Sensor Noise Model Meets Dense Visual Odometry. In *Proceedings of the 2016 IEEE International*

Symposium on Mixed and Augmented Reality, ISMAR 2016, pages 18–26, 2016. ISBN 9781509036417. doi: 10.1109/ISMAR.2016.11.

Baker, S. and Matthews, I. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000011205.11775.fd.

Bay, H., Ferrari, V., and Van Gool, L. Wide-baseline stereo matching with line segments. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:329–336, 2005. ISSN 10636919. doi: 10.1109/CVPR.2005.375.

Bay, H., Tuytelaars, T., and Gool, L. V. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, 2006.

Besl, P. J. and McKay, N. D. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. ISSN 01628828. doi: 10.1109/34.121791.

Bouguet, J.-y. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *In Practice*, 1(2):1–9, 1999. ISSN 0966842X. doi: 10.1016/j.tim.2005.08.009.

Bratanič, B., Pernuš, F., Likar, B., and Tomaževič, D. Real-time pose estimation of rigid objects in heavily cluttered environments. *Computer Vision and Image Understanding*, 141:38–51, 2015. ISSN 1090235X. doi: 10.1016/j.cviu.2015.09.002.

Brown, M., Windridge, D., and Guillemaut, J. Y. A generalisable framework for saliency-based line segment detection. *Pattern Recognition*, 48(12): 3993–4011, 2015. ISSN 00313203. doi: 10.1016/j.patcog.2015.06.015.

Buczko, M. and Willert, V. Flow-decoupled normalized reprojection error for visual odometry. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1161–1167, 2016. ISBN 9781509018895. doi: 10.1109/ITSC.2016.7795703.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M., and Siegwart, R. The EuRoC MAV Datasets. *Int. J. of Robotics Research*, 2015.

Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. ISSN 01628828. doi: 10.1109/TPAMI.1986.4767851.

Caruso, D., Engel, J., and Cremers, D. Large-scale direct SLAM for omnidirectional cameras. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2015-Decem, pages 141–148, 2015. ISBN 9781479999941. doi: 10.1109/IROS.2015.7353366.

Choi, C. and Christensen, H. I. 3D textureless object detection and tracking: An edge-based approach. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3877–3884, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6386065.

Civera, J., Davison, A. J., and Montiel, J. M. M. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008. ISSN 15523098. doi: 10.1109/TRO.2008.2003276.

Concha, A. and Civera, J. DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5686–5693, 2015. ISBN 9781479999934.

Costa, M. S. and Shapiro, L. G. 3D Object Recognition and Pose with Relational Indexing. *Computer Vision and Image Understanding*, 79(3):364–407, 2000. ISSN 10773142. doi: 10.1006/cviu.2000.0865.

Cummins, M. and Newman, P. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6): 647–665, 2008. ISSN 02783649. doi: 10.1177/0278364908090961.

Damen, D., Bunnun, P., Calway, A., and Mayol-cuevas, W. Real-time Learning and Detection of 3D Texture-less Objects: A Scalable Approach. *Procedings of the British Machine Vision Conference 2012*, pages 23.1–23.12, 2012. doi: 10.5244/C.26.23.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. MonoSlam: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):16, 2007. doi: 10.1109/TPAMI.2007.1049.

De Ma, S. Conics-based stereo, motion estimation, and pose determination. *International Journal of Computer Vision*, 10(1):7–25, 1993. ISSN 09205691. doi: 10.1007/BF01440844.

Debevec, P. E. and Malik, J. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, pages 3–8, 1997. ISBN 0897918967. doi: 10.1145/258734.258884.

Devernay, F. E. E. and Faugeras, O. Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. *Machine Vision and Applications*, 13(1):14–24, 2001. ISSN 0932-8092. doi: 10.1007/PL00013269>.

Douglas, D. H. and Peucker, T. K. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Classics in Cartography: Reflections on Influential Articles from Cartographica*, 10:15–28, 2011. ISSN 0317-7173. doi: 10.1002/9780470669488.ch2.

Drummond, T. and Cipolla, R. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7): 932–946, 2002. ISSN 01628828. doi: 10.1109/TPAMI.2002.1017620.

Ellis, T., Abbood, A., and Brillault, B. Ellipse detection and matching with uncertainty. *Image and Vision Computing*, 10(5):271–276, 1992. ISSN 02628856. doi: 10.1016/0262-8856(92)90041-Z.

Engel, J. *Large-Scale Direct SLAM and 3D Reconstruction in Real-time*. PhD thesis, The Technical University of Munich, 2016.

Engel, J., Sturm, J., and Cremers, D. Semi-dense visual odometry for a monocular camera. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456, 2013. ISSN 1550-5499. doi: 10.1109/ICCV.2013.183.

Engel, J., Sch, T., and Cremers, D. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, pages 834–849, 2014. ISBN 9783319106045. doi: 10.1007/978-3-319-10605-2_54.

Engel, J., Stückler, J., and Cremers, D. Large-scale direct SLAM for stereo cameras. In *IEEE International Conference on Intelligent Robots and Systems*, 2015. doi: 10.1109/IROS.2015.7353631.

Engel, J., Koltun, V., and Cremers, D. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016a. ISSN 0162-8828. doi: 10.1109/TPAMI.2017.2658577.

Engel, J., Usenko, V., and Cremers, D. A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*, 2016b.

Felzenszwalb, P. F. and Huttenlocher, D. P. Distance transforms of sampled functions. *Cornell Computing and Information Science Technical Report TR20041963*, 4:1–15, 2004. ISSN 1557-2862. doi: 10.4086/toc.2012.v008a019.

Fitzgibbon, A., Pilu, M., and Fisher, R. B. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5): 476–480, 1999. ISSN 01628828. doi: 10.1109/34.765658.

Forster, C., Pizzoli, M., and Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 15–22, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906584.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. ISSN 15523098. doi: 10.1109/TRO.2016.2623335.

Galvez-López, D. and Tardos, J. D. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5): 1188–1197, oct 2012. ISSN 1552-3098. doi: 10.1109/TRO.2012.2197158.

Gao, F. and Han, L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1): 259–277, 2012. ISSN 09266003. doi: 10.1007/s10589-010-9329-3.

Gao, X., Wang, R., Demmel, N., and Cremers, D. LDSO: Direct Sparse Odometry with Loop Closure. In *International Conference on Intelligent Robots and Systems*, 2018.

Garon, M. and Lalonde, J. F. Deep 6-DOF Tracking. *IEEE Transactions on Visualization and Computer Graphics*, 23(11):2410–2418, 2017. ISSN 10772626. doi: 10.1109/TVCG.2017.2734599.

Gomez-Ojeda, R., Moreno, F. A., Zuñiga-Noël, D., Scaramuzza, D., and Gonzalez-Jimenez, J. PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments. *IEEE Transactions on Robotics*, 35(3):734–746, 2019. ISSN 15523098. doi: 10.1109/TRO.2019.2899783.

Hanson, R., Falkenström, W., and Miettinen, M. Augmented reality as a means of conveying picking information in kit preparation for mixed-model assembly. *Computers & Industrial Engineering*, 113:570–575, nov 2017. ISSN 03608352. doi: 10.1016/j.cie.2017.09.048.

Harris, C. and Stephens, M. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988. ISSN 09639292. doi: 10.5244/C.2.23.

Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. ISBN 9780511811685.

Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. Dominant orientation templates for real-time detection of texture-less objects. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2257–2264, 2010. ISBN 9781424469840. doi: 10.1109/CVPR.2010.5539908.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012a. ISSN 01628828. doi: 10.1109/TPAMI.2011.206.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, volume 7724, pages 548–562, 2012b. ISBN 9783642373305. doi: 10.1007/978-3-642-37331-2_42.

Imperoli, M. and Pretto, A. D2co: Fast and robust registration of 3d textureless objects using the directional chamfer distance. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9163:316–328, 2015. ISSN 16113349. doi: 10.1007/978-3-319-20904-3_29.

Kannala, J. and Brandt, S. S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006. ISSN 01628828. doi: 10.1109/TPAMI.2006.153.

Kerl, C., Sturm, J., and Cremers, D. Robust odometry estimation for RGB-D cameras. In *Proceedings - IEEE International Conference on Robotics and*

Automation, pages 3748–3754, 2013. ISBN 9781467356411. doi: 10.1109/ICRA.2013.6631104.

Klein, G. and Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR - IEEE International Symposium on Mixed and Augmented Reality*, 2007. ISBN 9781424417506.

Klein, G. and Murray, D. Improving the agility of keyframe-based SLAM. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5303 LNCS(PART 2):802–815, 2008. ISSN 03029743. doi: 10.1007/978-3-540-88688-4-59.

K.Vairalkar, M. and S.U.Nimbhorkar. Edge Detection of Images Using Sobel Operator. *International Journal of Emerging Technology and Advanced Engineering*, 2(1):291–293, 2012.

Lagarias, Jeffrey, C., Reeds, James, A., Wright, Margaret, H., and Wright, Paul, E. Convergence properties of the nelder mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

Lamarca, J. and Montiel, J. M. Camera tracking for SLAM in deformable maps. In *European Conference on Computer Vision Workshop*, 2018.

Lange, K. L., Little, R. J., and Taylor, J. M. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408): 881–896, 1989. ISSN 1537274X. doi: 10.1080/01621459.1989.10478852.

Leizea, I., Mendizabal, A., Alvarez, H., Aguinaga, I., Borro, D., and Sanchez, E. Real-Time Visual Tracking of Deformable Objects in Robot-Assisted Surgery. *IEEE Computer Graphics and Applications*, 37(1):56–68, 2017. ISSN 02721716. doi: 10.1109/MCG.2015.96.

Lepetit, V., Moreno-Noguer, F., and Fua, P. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. ISSN 09205691. doi: 10.1007/s11263-008-0152-6.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334, 2015. ISSN 17413176. doi: 10.1177/0278364914554813.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013. ISSN 00221031. doi: 10.1016/j.jesp.2013.03.013.

Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *European Conference on Computer Vision (ECCV)*, 2018. ISBN 9783030012304. doi: 10.1007/978-3-030-01231-1_42.

Liu, C. and Rubin, D. B. ML estimation of the t distribution using EM and its extensions ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

Liu, M.-Y., Tuzel, O., Veeraraghavan, a., Taguchi, Y., Marks, T. K., and Chellappa, R. Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973, 2012. ISSN 0278-3649. doi: 10.1177/0278364911436018.

Liu, M. Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. Fast directional chamfer matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1696–1703, 2010. ISBN 9781424469840. doi: 10.1109/CVPR.2010.5539837.

Liu, Z. and Marlet, R. Virtual Line Descriptor and Semi-Local Graph Matching Method for Reliable Feature Correspondence. *Procedings of the British Machine Vision Conference 2012*, pages 16.1–16.11, 2012. doi: 10.5244/C.26.16.

Lowe, D. G. Distinctive image features from scale invariant keypoints. *Int'l Journal of Computer Vision*, 60:91–11020042, 2004. ISSN 0920-5691. doi: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Lu Wang, Neumann, U., and You, S. Wide-baseline image matching using Line Signatures. In *2009 IEEE 12th International Conference on Computer Vision*, number Iccv, pages 1311–1318, 2009. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459316.

Makris, S., Karagiannis, P., Koukas, S., and Matthaiakis, A.-S. Augmented reality system for operator support in human robot collaborative assembly. *CIRP Annals*, 65(1):61–64, 2016. ISSN 00078506. doi: 10.1016/j.cirp.2016.04.038.

Matsuki, H., von Stumberg, L., Usenko, V., Stückler, J., and Cremers, D. Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras. 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2855443.

Mendikute, A., Yagüe-Fabra, J. A., Zatarain, M., Bertelsen, Á., and Leizea, I. Self-Calibrated In-Process photogrammetry for large raw part measurement and alignment before machining. *Sensors*, 17(9), 2017. ISSN 14248220. doi: 10.3390/s17092066.

Micusik, B. and Wildenauer, H. Descriptor free visual indoor localization with line segments. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:3165–3173, 2015a. ISSN 10636919. doi: 10.1109/CVPR.2015.7298936.

Micusik, B. and Wildenauer, H. Structure from motion with line segments under relaxed endpoint constraints. *Proceedings - 2014 International Conference on 3D Vision, 3DV 2014*, pages 13–19, 2015b. doi: 10.1109/3DV.2014.17.

Mur-Artal, R. and Tardos, J. D. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. In *Computer Vision and Pattern Recognition*, 2016a. ISBN 1552-3098. doi: 10.1109/TRO.2012.2197158.

Mur-Artal, R. and Tardos, J. D. Visual-Inertial Monocular SLAM with Map Reuse. *IEEE Robotics and Automation Letters*, 2(2), 2016b. ISSN 2377-3766. doi: 10.1109/LRA.2017.2653359.

Mur-Artal, R., Montiel, J. M., and Tardos, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5): 1147–1163, 2015. ISSN 15523098. doi: 10.1109/TRO.2015.2463671.

Newcombe, R. A. *Dense Visual SLAM*. PhD thesis, Imperial College London, 2012.

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. DTAM: Dense tracking and mapping in real-time. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2320–2327, 2011. ISSN 1550-5499. doi: 10.1109/ICCV.2011.6126513.

Nistér, D. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004. ISSN 01628828. doi: 10.1109/TPAMI.2004.17.

Nocedal, J. and Stephe, J. W. *Numerical Optimization*. Springer-Verlag New York, 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5.

Pagani, A., Koehle, J., and Stricker, D. Circular Markers for camera pose estimation. *The 12th International Workshop on Image Analysis for Multimedia Interactive Services - WIAMIS 2011*, 2011.

Palmarini, R., Erkoyuncu, J. A., Roy, R., and Torabmostaedi, H. A systematic review of augmented reality applications in maintenance. *Robotics and Computer-Integrated Manufacturing*, 49(March 2017):215–228, 2018. ISSN 07365845. doi: 10.1016/j.rcim.2017.06.002.

Patraucean, V., Gurdjos, P., and Von Gioi, R. G. A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7573 LNCS(PART 2):572–585, 2012. ISSN 03029743. doi: 10.1007/978-3-642-33709-3_41.

Peng, X. Combine color and shape in real-time detection of texture-less objects. *Computer Vision and Image Understanding*, 135:31–48, 2015. ISSN 1090235X. doi: 10.1016/j.cviu.2015.02.010.

Pillai, S. and Leonard, J. Monocular SLAM Supported Object Recognition. In *Robotics: Science and Systems (RSS)*, pages 34–42, 2015. ISBN 9780992374716. doi: 10.15607/RSS.2015.XI.034.

Pizzoli, M., Forster, C., and Scaramuzza, D. REMODE: Probabilistic, monocular dense reconstruction in real time. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2609–2616, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907233.

Plantefève, R., Peterlik, I., Haouchine, N., and Cotin, S. Patient-Specific Biomechanical Modeling for Guidance During Minimally-Invasive Hepatic Surgery. *Annals of Biomedical Engineering*, 44(1):139–153, 2016. ISSN 15739686. doi: 10.1007/s10439-015-1419-z.

Platonov, J., Heibel, H., Meier, P., and Grollmann, B. A mobile markerless AR system for maintenance and repair. *Proceedings - ISMAR 2006: Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 105–108, 2007. doi: 10.1109/ISMAR.2006.297800.

Prasad, D. K., Leung, M. K. H., and Quek, C. ElliFit: An unconstrained, non-iterative, least squares based geometric Ellipse Fitting method. *Pattern Recognition*, 46(5):1449–1465, 2013. ISSN 00313203. doi: 10.1016/j.patcog.2012.11.007.

Qin, T., Li, P., and Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, pages 1–17, 2018. ISSN 15523098. doi: 10.1109/TRO.2018.2853729.

Rafael, G. V. G., Jérémie, J., Jean-Michel, M., and Gregory, R. LSD: a Line Segment Detector. *Image Processing On Line*, 2:35–55, 2012. ISSN 2105-1232. doi: 10.5201/ipol.2012.gjmr-lsd.

Ragni, M., Perini, M., Setti, A., and Bosetti, P. ARTool Zero: Programming trajectory of touching probes using augmented reality. *Computers and Industrial Engineering*, 124(July):462–473, 2018. ISSN 03608352. doi: 10.1016/j.cie.2018.07.026.

Rosin, P. L. Ellipse Fitting Using Orthogonal Hyperbolae and Stirling's Oval. *Graphical Models and Image Processing*, 60(3):209–213, 1998. ISSN 10773169. doi: 10.1006/gmip.1998.0471.

Rosten, E. and Drummond, T. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS: 430–443, 2006. ISSN 16113349. doi: 10.1007/11744023_34.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. ORB: An efficient alternative to SIFT or SURF. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011. ISSN 1550-5499. doi: 10.1109/ICCV.2011.6126544.

Schönberger, J. L. and Frahm, J.-M. Structure-from-Motion Revisited. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Schops, T., Enge, J., and Cremers, D. Semi-dense visual odometry for AR on a smartphone. In *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, pages 145–150, 2014. ISBN 9781479961849. doi: 10.1109/ISMAR.2014.6948420.

Schubert, D., Demmel, N., Usenko, V., Stückler, J., and Cremers, D. Direct Sparse Odometry with Rolling Shutter. 2018.

Shi, J. S. J. and Tomasi, C. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. ISSN 1063-6919. doi: 10.1109/CVPR.1994.323794.

Singer, S. and Singer, S. Efficient Implementation of the Nelder-Mead Search Algorithm. *Applied Numerical Analysis & Computational Mathematics*, 1(2): 524–534, 2004. ISSN 1611-8170. doi: 10.1002/anac.200410015.

Strasdat, H. *Local accuracy and global consistency for efficient visual SLAM*. PhD thesis, Imperial College London, 2012.

Strasdat, H., Montiel, J. M., and Davison, A. J. Real-time monocular SLAM: Why filter? In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2657–2664, 2010. ISBN 9781424450381. doi: 10.1109/ROBOT.2010.5509636.

Strasdat, H., Davison, A. J., Montiel, J. M., and Konolige, K. Double window optimisation for constant time visual SLAM. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2352–2359, 2011. ISBN 9781457711015. doi: 10.1109/ICCV.2011.6126517.

Stühmer, J., Gumhold, S., and Cremers, D. Real-Time Dense Geometry from a Handheld Camera BT - Pattern Recognition. *Pattern Recognition*, 6376 (Chapter 2):11–20, 2010. doi: 10.1007/978-3-642-15986-2_2.

Szeliski, R. *Computer Vision : Algorithms and Applications*, volume 5. Springer, 2010. ISBN 1848829345. doi: 10.1007/978-1-84882-935-0.

Tekin, B., Sinha, S. N., and Fua, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. ISBN 1-4244-0209-3. doi: 10.1109/CVPR.2018.00038.

Tombari, F., Franchi, A., and Di, L. BOLD features to detect texture-less objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1265–1272, 2013. ISBN 9781479928392. doi: 10.1109/ICCV.2013.160.

Topal, C. and Akinlar, C. Edge Drawing: A combined real-time edge and segment detector. *Journal of Visual Communication and Image Representation*, 23(6): 862–872, 2012. ISSN 10473203. doi: 10.1016/j.jvcir.2012.05.004.

Triggs, B., McLauchlan, P. F. P., Hartley, R. I., and Fitzgibbon, A. W. A. Bundle Adjustment. A Modern Synthesis. In *International Worshop on Vision Algorithms*, volume 34099, pages 298–372, 2000. ISBN 3540679731. doi: 10.1.1.37.6846.

Usabiaga, J., Erol, A., Bebis, G., Boyle, R., and Twombly, X. Global hand pose estimation by multiple camera ellipse tracking. *Machine Vision and Applications*, 21(1):1–15, 2009. ISSN 09328092. doi: 10.1007/s00138-008-0137-z.

Usenko, V., Engel, J., Stuckler, J., and Cremers, D. Reconstructing Street-Scenes in Real-Time from a Driving Car. In *Proceedings - 2015 International Conference on 3D Vision, 3DV 2015*, pages 607–614, 2015. ISBN 9781467383325. doi: 10.1109/3DV.2015.75.

Usenko, V., Engel, J., Stuckler, J., and Cremers, D. Direct visual-inertial odometry with stereo cameras. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 1885–1892, 2016. ISBN 9781467380263. doi: 10.1109/ICRA.2016.7487335.

Usenko, V., Demmel, N., and Cremers, D. The double sphere camera model. In *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pages 552–560, 2018. ISBN 9781538684252. doi: 10.1109/3DV.2018.00069.

Varadarajan, V. S. *Lie Groups, Lie Algebras, and Their Representations*, volume 102 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 1984. ISBN 978-1-4612-7016-4. doi: 10.1007/978-1-4612-1126-6.

Verhagen, B., Timofte, R., and Van Gool, L. Scale-invariant line descriptors for wide baseline matching. *2014 IEEE Winter Conference on Applications of Computer Vision, WACV 2014*, pages 493–500, 2014. doi: 10.1109/WACV.2014.6836061.

von Stumberg, L., Usenko, V., and Cremers, D. Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization. 2018. ISBN 9781538630808.

von Stumberg, L., Wenzel, P., Khan, Q., and Cremers, D. GN-Net: The Gauss-Newton Loss for Deep Direct SLAM. Technical report, 2019.

Wang, G., Wu, J., and Ji, Z. Single view based pose estimation from circle or parallel lines. *Pattern Recognition Letters*, 29(7):977–985, 2008. ISSN 01678655. doi: 10.1016/j.patrec.2008.01.017.

Wang, Y., Zhang, S., Yang, S., He, W., Bai, X., and Zeng, Y. A LINE-MOD-based markerless tracking approachfor AR applications. *International Journal of Advanced Manufacturing Technology*, 89(5-8):1699–1707, 2017. ISSN 14333015. doi: 10.1007/s00170-016-9180-5.

Wang, Y., Zhang, S., Wan, B., He, W., and Bai, X. Point cloud and visual feature-based tracking method for an augmented reality-aided mechanical assembly system. *International Journal of Advanced Manufacturing Technology*, 2018. ISSN 14333015. doi: 10.1007/s00170-018-2575-8.

Wang, Z., Liu, H., and Wu, F. HLD: A robust descriptor for line matching. *Proceedings - 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2009*, pages 128–133, 2009a. ISSN 00313203. doi: 10.1109/CADCG.2009.5246918.

Wang, Z., Liu, H., and Wu, F. HLD: A robust descriptor for line matching. *Proceedings - 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2009*, 42(5):128–133, 2009b. ISSN 00313203. doi: 10.1109/CADCG.2009.5246918.

Welch, G. and Bishop, G. An Introduction to the Kalman Filter. Technical report, 2006.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Robotics: Science and Systems (RSS)*, 2017. ISBN 978-0-9923747-4-7. doi: 10.15607/RSS.2018.XIV.019.

Zhang, L. and Koch, R. Line matching using appearance similarities and geometric constraints. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7476 LNCS(2009611008):236–245, 2012. ISSN 03029743. doi: 10.1007/978-3-642-32717-9_24.

Zhang, L. and Koch, R. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013. ISSN 10473203. doi: 10.1016/j.jvcir.2013.05.006.

Zhang, L. and Koch, R. Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation*, 25(5):904–915, 2014. ISSN 10959076. doi: 10.1016/j.jvcir.2014.02.013.

Zhang, L., Xu, C., Lee, K. M., and Koch, R. Robust and efficient pose estimation from line correspondences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

Bioinformatics), 7726 LNCS(PART 3):217–230, 2013. ISSN 03029743. doi: 10.1007/978-3-642-37431-9_17.

Zhang, X., Zhang, Z., Li, Y., Zhu, X., Yu, Q., and Ou, J. Robust camera pose estimation from unknown or known line correspondences. *Appl. Opt.*, 51(7): 936–948, 2012. ISSN 1539-4522. doi: 10.1364/AO.51.000936.

Zhao, C., Zhao, H., Lv, J., Sun, S., and Li, B. Multimodal image matching based on Multimodality Robust Line Segment Descriptor. *Neurocomputing*, 177: 290–303, 2016. ISSN 18728286. doi: 10.1016/j.neucom.2015.11.025.