Exploration of Blender to design a digital twin of multi camera metrology systems

Exploration of Blender to design a digital twin of multi camera metrology systems

by

Claire Pottier

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

March 2023

Abstract

After three industrial revolutions that transformed the agricultural and craft economy into one dominated by industry and machine manufacturing, a fourth industrial revolution has recently begun, plunging the world into an era of digitisation where new technologies such as the digital twin, the digital model, human-robot collaboration or the internet of things are being explored to create new models for agile factories that are reconfigurable, resilient, fast and knowledgeable.

In this new philosophy, called Industry 4.0, new needs in terms of worker safety, work supervision and quality control of goods have been developed, as well as new issues related to humans, economy, and technology development. Regarding these new paradigms, solutions based on multi-camera optical systems have been developed.

However, current physical available solutions are not necessarily optimised, cameras are installed according to geometry specificities such as being collinear, or through learning trial-and-error processes. Due to the recurrent nature of the optimisation problem, the development of virtual technologies generating new possibilities for virtualising the physical assets of the production chain, digital models or even potentially twins are being considered to solve this problem.

This research investigates the development of a digital model/twin of single and multi-camera optical systems to measure shapes and objects in an industrial environment. A literature survey was conducted to identify the solutions already developed, identifying their drawbacks to design an innovative system. Through this survey it was identified that the current modelling solutions are either mimicking the real-world in a simple way, or are designing the camera from its whole body to its radiometric map, which is time consuming, and maybe not lead to an accurate model.

An alternative at these designs was found in the 3D animation software, which use 3D virtual camera in its rendering process. Various pieces of 3D animation software exist, but Blender was chosen because it is open source, free, and offer divers tool such as ray-tracing and Python script.

The research presented maps the development of a digital model/twin of single and multi-camera optical systems in Blender through four steps: investigation whether Blender can be used as a digital base; designing a photorealistic digital model of the real-world; assessing the robustness of the model designed; and identification of positive and negative aspects of the modelling.

All experiments were completed in the real and virtual worlds to allow understanding and comparison of results.

The main conclusions of this research were the development of a digital representation of the real multi-camera network set up in Blender; the use of photorealism concepts; and the development of calibration routines for camera-based metrology systems modelled in Blender.

Acknowledgements

I would like to thank my supervisors: Professor Peter Kinnell, Dr Jon Petzing and Professor Neils Lohse at Loughborough University; and Dr Fariborz Eghtedari at the Manufacturing Technology Centre (MTC), in Coventry.

Thank you to Peter for his support and for offering me this amazing opportunity to come to the UK, and to work on this wonderful topic.

I sincerely thanks Fariborz for his help, support and ideas; as well as for the opportunity I had to present my work at one of the MTC workshops.

A big thanks to Jon who has given me incredible support and advice throughout this PhD journey, who has taught me a lot, who has pushed me to go beyond my limits, to believe in myself, and thanks to whom this project has become a reality - I am really grateful to have had the opportunity to work with you.

I would like to acknowledge Professor Jonathan Huntley and Dr Thomas Bamber, my internal reviewers during the first- and second-year progression reviews, respectively, as well as Dr Pedro Ferreira for his advices, and support.

Part of this PhD was carried out during the height of the Covid-19 pandemic, when the whole country spent months in lockdown. I would like to thank my friends Achim, Ezgi, Kathy, Marnessa, Myles, Qusay, Saad, Tianqi, Xiaohu, Zhi for their help and support, as well as all the good moments we had in the office, and outside.

Finally, with all my heart I thank my wonderful family. To my dearest parents, Jean-François and Nicole, as well as my grandmother Simone, thanks for always being there for me, for believing in me, and giving me your constant love, support and encouragement. This journey would not have been possible without you.

Publications and Output

Conferences:

- **C.Pottier**, J.Petzing, F.Eghtedari, N.Lohse, P.Kinnell (2021 November). Blender – A 3D animation software explored as a metrology tool. 3D Metrology Conference. Bilbao, Spain.
- **C.Pottier**, J.Petzing, F.Eghtedari, N.Lohse, P.Kinnell (2022 June). Multi-camera Large Volume for Manufacturing Applications. MTC.
  Coventry, UK.
- **C.Pottier**, J.Petzing, F.Eghtedari, N.Lohse, P.Kinnell (2022 September). Designing a photorealistic digital twin in Blender. IMEKO M4Dconf2022. Berlin, Germany.
- **C.Pottier**, J.Petzing, F.Eghtedari, N.Lohse, P.Kinnell (2022 November). Developing virtual metrology systems in Blender and assessing measurement performance compliance. 3D Metrology Conference. Aachen, Germany.

Journal paper:

- Journal paper in review – Measurement Science and Technology – IOP Science **C.Pottier**, J.Petzing, F.Eghtedari, N.Lohse, P.Kinnell. Developing digital twins of multi-camera metrology systems in Blender.
  Reference: C Pottier et al 2023 Meas. Sci. Technol. 34 075001 DOI 10.1088/1361-6501/acc59e

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial progress has been the driving force behind economic expansion. Optimizing and increasing factory manufacturing transformed and fuelled innovation, job creation and unprecedented material wealth over the last three centuries and has propelled engineers to anticipate the problems of tomorrow [1].

With these transformations, otherwise known as revolutions, production within agriculture and craft moved away from artisanship and towards domination by industry and machine manufacturing thanks to technological advancements and inventions [2]. Beginning the late 18th century, the first industrial revolution was kickstarted by the invention of the steam engine by James Watt, which enabled the development of new and faster types of transport and new means of communication [1].

Alongside these revolutions came corresponding concepts. In the late 19th century, the second industrial revolution ushered in the concept of the assembly line, and the principle of mass production. Alongside the evolution of manufacturing processes there occurred corresponding development in the sphere of worker safety, work supervision and quality control of goods [1].

Today in the United Kingdom, the health and safety of workers is nowadays ensured by laws such as the Working Time Regulations (1998) or the Regulatory Reform (Fire Safety) Order (2005). This evolution has taken over 200 years to arrive at the current laws and guidelines. This tradition began in 1802 with the Factory Act. Another breakthrough was the 1974 Work Act concerning Health and Safety which has set a standard for health and safety legislation across the world today [3].

Modern work management is currently optimised through strategic planning, customer relationship management, employee engagement surveys, benchmarking and balanced Scorecard (report that can be used by managers to keep track of the execution of activities by the staff within their control and to monitor the consequences arising from these actions) [4]. On the other hand, the quality control of goods is done through tests on samples collected at different steps of the manufacturing process. These tests help to isolate and identify the cause of a production problem and intuitively help implement the necessary corrective actions to prevent future incidents [5].

In the different processes described above, the work is often done manually and by regulation, so complex technologies are not needed. The only technologies used are the Internet, for quick searches and communication between each branch within ~~of~~ the network, and computers for numerical and statistical calculations. Yet, with the next revolution and development of Industry 4.0, some industrial concepts have evolved or were made redundant and replaced. For instance, within the production line, technologies are increasingly being integrated to empower manual work, provide more safety, speed up production and improve decision making [6].

Industry 4.0 can be defined as the latest industrial revolution. It is a philosophy of industry that responds to social, economic and political changes by increasing mechanisation and automation, digitalisation, networking and miniaturisation [7].

However, the willingness of factories to align themselves with this philosophy encounters human, economic and practical problems. Human problems stem from the pressures of digitalisation and automation. Technological progress makes people afraid of losing their jobs, but likewise they may not understand the changes in their workplace, and thus resist due to a sense of betrayal and lack of commitment to the evolutionary process, fitting well into the Luddite tradition [8]. Economic problems may be encountered due to the high cost of implementing, running and maintaining technologies such as automation and robots. Practical problems are a mixture of the two. People may not be properly trained in the use of increasingly complicated equipment, the technologies are sometimes not available, or available in a precursory state with important questions left without satisfying answers. For example, camera systems are used to monitor people or equipment and are easily available. However, questions about the optimal quantity of cameras and their positions in a factory to monitor people effectively lack satisfactory answers.

In the new industry model, the development of virtual technologies has introduced the notion of virtual space. Simulation tools and wireless networks create new possibilities to virtualize the physical assets of the production line in a computer system [9]. Importantly, the coexistence between the physical and virtual spaces has led to greater interactions between the virtual and the physical worlds than ever before, and in the process raising the question about the best ways of connecting physical and virtual spaces [10].

In this new industrial and economic context, the issues of worker safety, work management and quality control of goods are more prevalent than ever. However, these existing concerns are now compounded by the need to connect the virtual and physical industrial layers. To meet this demand new solutions are emerging with the rise of technologies driven by the new industrial model. Due to these different issues, non-intrusive technologies capable of tracking shapes (human and objects) such as optical systems have been promoted.

Optical systems have been widely used in manufacturing to track the motion of industrial problems. Their use has been facilitated by their accessibility, and simple implementation, as well as low cost and non-intrusiveness. However, different camera optical technologies have their own difficulties. These range from wearability and portability challenges of the systems [11], optimum placement of cameras in the physical environment, as well as accuracy problems in the detection [12], or optimisation problems [13].

The nature of problems faced by optical systems and the need to connect the virtual and physical layers of the industry has combined into the notion of developing digital twins and digital in response. By responding to industrial concerns head on, this development further encourages the digitisation of the factory space.

This digitalisation trend is defined as the automation of traditional manufacturing processes and systems, using modern ICT (Information and Communication Technology) technologies. This digitalisation process is applied at different levels, from software upgrades to the whole infrastructure and systems, to best revolutionise traditional production processes and activities [14].

The goal of this digitisation process is not simply to automate factories. Rather, it is to create agile, resilient, fast, and knowledgeable manufacturing companies that from the shop floor to the management can draw on resources as needed, while reducing risk and maximising [15-17].

Digitisation meets the industry's needs for worker safety, work management, product quality control. Digital twins and digital models have begun to be combined with the optical system to eliminate past problems related to optimisation, such as the optimum number of cameras necessary to cover a given volume, or the appropriate positioning of tracking markers within that volume [18-19].

However, the recent emergence of the digital twin concept (2010) and digitisation in factories have raised questions about the accuracy and reliability of the digital optical system representation. In addition, the concept and development of the digital twin has been called into question. The digital twin of the optical system can be designed in MATLAB from the radiometric map of the sensor to the camera body, or other solutions such as 3D animation software can be used. However, the use of 3D animation software, which will provide a ready-to-use virtual camera, raises the question of the degree of realism of the virtual model. These issues can be addressed through different/complete/varied metrology processes.

Metrology is the science of measurement and has been deeply intertwined with industry. It is an “old” science, which was known and used by ancient civilisations in China, India, Egypt and Mesopotamia [20]. Nowadays, this science is an integral tool used for basic measurement that can be taken for granted; measuring the weight vegetables. It likewise used in more complex tasks, such as measuring the position of mirrors on the James Webb space telescope.

Due to their recent advent in the new industrial model, metrology concepts face start-up issues. These consist of developing autonomous and reliable production lines, but also developing systems capable of analysing the causes of problems related to the evolution of parameters and machines in big data systems [21].

Currently, intergovernmental and international non-governmental organisations such as BIPM (International Bureau of Weights and Measures) [21], or IMEKO (International Measurement Confederation) [22], and national laboratories and institute such as NPL (National Physical Laboratory) [23], or PTB (Physikalisch-Technische Bundesanstalt) [24] are developing solutions to solve the challenges of tomorrow.

Nevertheless, this process has just started, and research is currently being carried out. Thus, metrology for Industry 4.0 is not yet fully here. This leaves companies alone with problems such as determining the accuracy of a digital twin which may turn them off using this new technology. The various gaps identified in this introduction lead to the Research Questions considered in this research.

1.1. Research questions

The background context of this research is the implementation of a multi-camera optical systems to measure shapes and track humans and objects in a factory environment. Due to the trend of digitalisation, and for the philosophy of Industry 4.0, the application of digital twin or digital model technologies has been considered to improve and aid the placement of optical systems.

Thus, the overall aim of this research is to develop a digital model/twin of single and multi-camera optical systems to measure shapes and track humans and objects in an industrial environment.

Regarding to the research aim - the development of a digital twin/model of single and multi-camera optical systems – research has already been published on that topic. For example, there are systems based on multi-layer neural networks and computer vision [25], which generate a simplified digital version of the model from images taken by real cameras; or systems based on a digital twin, as in cyber-physical systems, to enable optimisation of the planning and commissioning of human-based production processes [26]. However, they are time consuming to design due to the technologies and methods used, and might not be accurate.

Other alternatives to design a digital twin/model of a camera system is found in the 3D animation software, where virtual cameras are used as a key part of the rendering process. However, this type of software was originally designed for art and animation applications, but maybe an ideal platform for modelling the placement of cameras in a digital representation of the factory environment. Consequently, Research Question 1 considers; Can a camera-based metrology system be modelled in a 3D animation software?

Designing a digital twin/model of a real system raises the question of the complexity of the virtual equivalence. As shown by [25], and [26], the virtual system can be a simplified representation of the real-world system, or a near-perfect copy. So, Research Question 2 asks; to what extent should a virtual measurement model conform to the real-world equivalent?

Furthermore, as explained earlier, digital models/twins, like other technologies used in the development of Industry 4.0, are being used to empower manual labour, provide more safety, speed up production and improve decision-making. However, persistent issues about accuracy remains, leading to Research Question 3 - how can a metrology system based on a virtual camera and 3D animation software be made as traceable as a real-world system?

1.2. Research novelty

Depending on the industrial and metrological contexts, as well as the research questions, it is possible to give an initial idea of what the novelty of the research might be. As outlined below, three main points stand out:

- **The design of a camera-based metrology digital model/twin in a 3D animation software:**

  3D animation software is used in metrology applications as a tool to visualise point clouds [27], for simulations [28], or to generate images to train artificial intelligence [29]. However, 3D animation offers a wide set of tools such as modelling, ray-tracing, and Python scripts, which can be used to generate a virtual model of a real model. In addition, virtual cameras are already coded and implemented, and tracking tools are available for object measurement.

  The novelty would then be the utilisation by itself of a software to design a camera-based metrology digital model/twin.

- **Determination of the level of similarity between the real and virtual models:**

  By definition, a digital twin is a digital representation of an actual or intended physical product or system [30], and, in the context of this research, a digital model is considered as a virtual three-dimensional representation of an object that can be used for simulation and analysis [31]. According to the needs of Industry 4.0, the virtual model/twin should be similar to the real object, not a simplified representation.

  Generating a realistic model is necessary, as well as determining, through experiments, the optimal level of similarity required to achieve correct measurement results from the virtual camera-based metrology system.

  The novelty seems to design a virtual model like the real one, and to quantify the level of similarity through objective methods.

- **Calibration and reliability of the virtual-world system:**

  Metrology processes are in a continuous state of being developed to meet the needs of Industry 4.0. However, based on the analysis of processes and methods already developed to quantify the accuracy and precision of digital representations, as in [32], it is necessary and possible to develop methods to ensure the calibration and traceability of the virtual world camera-based metrology system to the relevant primary standard.

  The novelty could lie in the use of already existing methods, such as the Fast Fourier Transform, and applying them to the output and/or elements of the virtual system, in order to ensure the beginnings of traceability and calibration.

1.3. Thesis structure

Chapter Two reviews the literature and theory that underpins the research in this thesis. The literature review part of this chapter introduces the concept of Industry 4.0 and Smart Industry. As the work focuses on 3D animation software, cameras and the digital twin, camera components (such as the lens and shutter), the definition of the digital twin and an explanation of why 3D animation software was considered are covered, as well as a list of different potential software that could be used for research. The theoretical part of this chapter outlines important mathematical concepts required to model the camera and photogrammetry, such as reprojection errors and epipolar geometry, supported by the theory and history of camera calibration.

Chapter Three addresses the methodology. This chapter presents the methodologies used to perform the various experiments outlined in Chapters 4, 5, 6 and 7, with a list of the artefacts used, as well as a list of the algorithms developed for this work.

Chapter Four is about the exploration of how to create a digital model/twin in the virtual environment. The chapter presents the software chosen, and how to design a virtual camera in Blender (camera components, textures, and light management). The initial experiments to validate the first virtual reality model generated in the default software environment are also given, along with an application of the software. This application is about simulated different camera position around an object, to check the impact of the incidence angle on the measurement.

Chapter Five discusses the design of the photorealistic digital twin in the 3D animation world. A definition of the concept of photorealism is given before the results of the second experiment are presented. This second experiment focuses on the characteristics of the illumination environment, object-light interactions through texture surfaces, and the environment for generating a more photorealistic virtual model.

Chapter Six presents the robustness study of the photorealistic model designed in Chapter Five. In this chapter, the comparison between the real and virtual answers for different objects (simple and complex) is shown, and a conclusion about the validity of the model is addressed. In addition, the results of the study of new sources of errors are covered.

Chapter Seven explores the limits of the virtual system designed in Chapter Five through three different experiments. These experiments test the capacities of the virtual system to reproduce the same behaviours as the real system in the measurement of more complex objects, in a controlled room and in a more complex room. In addition, the ability of designing of more complex illumination environments was also explored to identify possible simulation problems.

Chapter Eight concludes on the research conducted in the thesis and reviews the research questions identified in Chapter 1. The novelty of the thesis, and the contributions to knowledge are also discussed in this chapter, as well as future work, and final remarks.

Chapter 2

Literature review

This research project is related to different fields because of its industrial and artistic characteristics. The objective of this literature review is to highlight the shortcomings of existing camera based measurement systems and the need to design a new virtual model of camera systems to meet the requirements of the factory of the future.

The development of a new industry model defines a new industrial age, using technologies to reshape the world of tomorrow and adapt to the new global market. In Industry 4.0, new paradigms are developed, and new challenges emerge. Optical systems are often cited technology to address these. Their state of the art is presented in Section 2.1.

This shows that optical systems are composed of a large variety of technologies such as; laser trackers, LIDAR, or camera-based systems. However, camera-based systems seem to be the predominate technology due to their non-intrusive nature and ease of use, they are used in a large range of applications such as in medicine for diagnostics, or in industry, for monitoring people, and tracking objects on an assembly line. Section 2.2 will then explain the functioning of cameras, as well as the mathematical model used to calculate their parameters. In addition, due to the focus on camera-based systems, the notion of Photogrammetry and the mathematical theory linked to it is explored in Section 2.3.

As identified above, industries are using camera-based systems for various applications. However, without camera calibration, useful and coherent measurements cannot be taken. The procedure of calibration allows to identify possible sources of errors, as well as to determine the camera parameters, and the corrections needed to be applied to them. Calibration techniques are then examined in Section 2.4, concentrating on two mathematical methods used later in this research: the reprojection error to calibrate the system, and the bundle adjustment to refine the result.

However, camera-based systems also have drawbacks and are not necessarily optimal solutions. The use of the digital twin as a possible alternative solution for optimising camera networks is one way of overcoming these problems. Section 2.5 provides a definition of the digital twin and explores different 3D animation software that can be used as a basis for creating a virtual model of reality with the aim of improving performance and usability.

2.1. Monitoring systems

Today's manufacturing industries are constantly evolving and integrating new technologies to meet their persistent needs for production efficiency, flexibility, competitiveness and sustainability.

The world has entered the fourth industrial revolution, where Smart Manufacturing, also called Industry 4.0 or Intelligent Manufacturing, is the new factory model with the purpose of optimizing production and product transactions by using advanced information and manufacturing technologies. This concept of the factory is also considered as a new model based on intelligent science and technology which greatly upgrades the design, production, management, and integration of the whole life cycle of a typical product [33 - 36].

With the development of Industry 4.0, the current factories have started to change and to called Smart Factory or Factory fo the Future. These new models are charaterised by implementation of new technologies such as 3D printing, artificial intelligence, digital twin, and sensors [37].

The use of these technologies is driven by the need to generate value and new opportunities for growth and productivity by reframing the competitive landscape with intelligent products and new service models. Smart manufactories are then characteristic by four factors, which meet these needs of growth and generation of value [37 -38]:

- Data nalysis for optimal decision making.
  The utilisation of embbedded sneors and interconnected machinery produce generate a lot of data which are used to investigate historical trends, identify patterns and make better decisions.
- IT-OT integration.
  Interconnectivity is the heart of the factory's network architecture. Real-time data generated by sensors, devices and machines on the factory floor can then be immediately used by other factory assets through this network.
- Custom manufacturing.
  Customized goods can be produced which meet individual customer's needs more cost-effectively.
- Supply chain.
  Transparency, and efficient supply chain are critical operations in the industry. This new industrial revolution allows a more efficient way to manage raw materials. In addition, product data can be share with suppliers, allowing a better shedule deliveries.

Three stages to transform factory into smart factory can then be drawn [37- 38]:

- Stage 1: Digital connectivy and sensors:
  Integration and automation of new technology to improve productivity, quality and efficiency and better manage risk. This is done at the company scale.
- Stage 2: Digital enginnering:
  Industry 4.0 technologies are used to enhance factory's product deisgn and supply chains, to develop smart products and cultivate upstream and downstram connected ecosystems. This is also done at the company scale.
- Stage 3: Digital operations:
  The company is fully digitalized, and is demarcating itself from rivals. This company can also develop whole new markets, services, and business models. Examples of this

dynamic are seen in fields involving bioinformatics, nanotechnology, and quantum technologies.

In this new type of industrial setting and infra-structure, remote monitoring based on optical systems, are and will be increasingly needed and used for; ensuring the safety of workers, monitoring different production technologies, production optimisation, and collision avoidance [39 - 42].

2.1.1. Optical Systems

Optical systems measure relative and absolute displacement and are often based on calibrated cameras and markers attached on work surfaces or a performer's body. Data are captured from a set of images generated by single or multiple cameras, where the 2D coordinates are extracted; triangulation is then performed in software to calculate the 3D position of the measured subject measured [43]. They are divided into three families: Passive markers, active markers, and the marker-less systems.

Marker systems

Passive marker systems are based on markers made of retro-reflective material which reflect the light generated near the camera lens [43]. For instance, StarTracker developed by Mo-Sys [44] uses retro-reflective stickers covering an entire scene, for tracking purposes. It is composed of a single camera and reports the position and orientation of the studio camera in real time to the rendering engine thanks to a "star map". The name "star" defines small, identical retro-reflective stickers, randomly placed in the scene. Due to the random placement of "stars" in the scene, the system does not have light restriction, and its tracking is absolute, nor does it have drift because it is always referencing itself to its "star map".

Another example of this technology is the NIKON K-CMM. This is based on active LED calibration which emit light tracked by special cameras. This is a flexible and common method used in the industry [45]. Portable, accurate and easy to use, NIKON K-CMM can perform indoor measurements, and is currently used for metrology applications [46].

Active marker systems use pulsed-LED to emit IR light [43]. A well-known example of this technology, is the VICON system. The VICON technology is based on passive LED calibration, i.e., the retroreflective markers are tracked by infrared cameras [45]. Being prominent in the optical system over the last 35 years, VICON provides tailored optical systems for a diverse range of applications [45].

NiKON and VICON systems are quite popular in the industrial, sport and biomechanics areas [47-48], it remains that their deployment and tracking volume are quickly limited by the localisation of the markers in the measurement environment. This is caused by the camera's field of view and associated occlusions, and the line of sight, as all solutions are based on markers [43].

Marker-less systems

Marker-less optical system has been developed to overcome the problems listed above. They are non-invasive systems, and they are not marker-based. In deciding not to use markers to detect and track

objects and people, marker-less technologies have had to find ways to collect images, and to represent the measurement volume, and what is in it.

To meet these challenges, various technologies have emerged, based on the emission of light in the visible or infrared light spectrum in form of the laser light, light patterns or modular light pulses [49]. Examples include laser trackers, time-of-flight lights detection and ranging (called LIDARs), and structured lights systems [49].

Laser trackers were developed in the 20th century by Lau et al [50, 51] at the National Institute of Standards and Technology (NIST) to facilitate robot metrology. They are portable coordinate measuring systems (CMS), and measure three-dimensional coordinate position of a target in a Cartesian system, distance between objects and along with two angles, providing the results in a spherical coordinate system [51, 52].

A laser tracker is an assembly of various mechanical and optical components. The distance is measured by the light path generated by the reflection of the laser beam on a retroflected marker. The system contains two angular encoders measuring the two angular axes of the trackers: the azimuth and the elevation (or zenith) axis [50].

This technology is widely used for the verification of large-scales pieces in the aerospace, spatial, or naval sectors, due to its high speed in data acquisition, wide measurement range in large-scale metrology, and accuracy [53]. However, a controlled environment is needed for its use, due to the laser, leading to use more in laboratories than in companies for safety reasons. Moreover, geometrical and optical misalignment errors exist, with errors linked to the encoders [49, 50].

Time-of-flight (TOF) LIDARS are based on infrared light, and are the well-developed technology for distance measurements in both single-spot depth ranging and 3D mapping. The distance corresponds to the delay between the emitted light by the system, and the reflected one by the object [54, 55].

Two types of TOF exist, the direct and indirect. Direct TOF employs a pulsed laser and a time measurement circuit. The association of both allows the direct measurement of the laser pulse reflected by the artefact being measured. In comparison, indirect TOF measures the phase difference between the emitted and reflected light [56].

This technology is used in several areas such as automotive applications [57], gesture recognition [58], and imaging-based optical metrology [59]. The advantages of this technology are its portability, its size, the possibility to use at any distances as long as enough reflected light returns to the sensor, high voxels density which provide multiple simultaneous range measurements, and its modulation frequency bandwidths, allowing, of instance, to reduce the noise by increasing the frequency deviation, or reducing the interference by the adjacent channels [60]. However, the accuracy is limited by multi-path interference error, phase wrapping of the modulation envelope from 2π to 0, multiple propagation paths from the light source of the pixel, and aliasing [61].

An alternative technology belonging to the structured lights systems family is demonstrated by Microsoft's Kinect widely used in biomechanical, industrial and medical area [62-63]. Composed of an infrared (IR) Emitter, an RGB-D sensor and an IR Depth sensor, this technology uses structured light principles, synchronized colour and depth images for tracking and detection applications [43].

The structured light is generated by the IR emitter, which creates a speckle pattern by projecting laser light that is reflect back to the IR depth sensor. The depth map of the scene is then constructed by

matching the pattern creating by the IR emitter, and the pattern detected by the IR depth sensor. When the light pattern hits an object in the scene, it is deformed. This information helps to generate the depth of map of the scene, and to measure the distances between the objects and the sensor [64].

The advantages of the Microsoft's Kinect are numerous, quick generation of 3D information in real time, a range of applications between 0.4 m and 7 m, cheap technology and portable. However, it is sensitive to external infrared sources such as sun light, and it is not suitable for outdoor applications. In addition, the precision decreases with increasing range according to a second order polynomial equation, and the detection of crystalline or highly reflective objects is not possible [65-66].

Two technologies were developed by Microsoft: Kinect 1 and Kinect 2. Table 2.1 gives an overview of these two technologies [67] and demonstrates that Kinect 2 is better than Kinect 1 due to the improved technologies used and the accuracy/precision.

Table 2.1: Overview of Kinect 1 and 2

Optical system	Sensor features	Depth and RGB map	Accuracy and Precision	Other
Kinect 1	- Acquisition rates of up 30 Hz for the depth and the RGB maps. - Depth sensor resolution: 640×480 pixels. - RGB sensor resolution: 320×240 pixels	- Depth data are obtained using structured light approach. - Infrared dot pattern is used for the RGB map. - Stereo triangulation is used to obtain 3D position.	- Accuracy: 10 mm - Precision: about 8 mm in the range of 1 m, and decreases rapidly following a second order polynomial	The fixed density of points limits accuracy when moving away from the camera, as the points become more scattered. This means that the boundaries of surfaces in the distance are often jagged.
Kinect 2	- Acquisition rates of up 30 Hz for the depth and the RGB maps. - Depth sensor resolution: 512 x 424 pixels. - RGB sensor resolution: 1920 × 1080 pixels	- Depth data are obtained using the time-of-flight. - Distance to points on the surface is measured by computing the phase-shift distance of modulated infrared light. - The intensity of the captured image is thus proportional to the distance of the points in 3D space.	- Accuracy: 10 mm - Precision: 8 mm in the range of under 1 m. - Precision: 25 mm in the range of 2 m.	The results coming from the TOF can suffer from various artifacts caused by the reflections of light signal from the scene geometry and the reflectance properties of observed materials.

As explained in this section, optical systems are divided into two groups: marker and marker-less optical systems. Marker technologies use two types of targets, active and passive. They are accurate (Table 2.2), easy to use, and portable, but they are limited due to their light of sight and restricted by the detection of the markers. Marker-less technologies have been developed to remove the markers restriction. Based on the light path and measuring the different between the light emitted and the light received, they are also portable and accurate (Table 2.2), but they can be used at a larger scale, in more applications.

However, most of these technologies cannot be used out of a controlled environment as in the case with laser trackers [49], or have multi-path interference error, and problems with the propagation of

the light. Nevertheless, an alternative solution based on single or multi-camera network systems, photogrammetry also seems to be a suitable candidate.

The key elements important to know about these technologies are summed up in Table 2.2. Noted that these technologies are commercially available.

Table 2.2: Commercially available optical systems

Optical system	Advantages	Disadvantages	Accuracy	Precision
NIKON K-CMM	- Flexible method. - Portable. - Easy to use.	- Deployment and tracking volume are quickly limited by the markers' locations. - Occlusions. - Line of sight.	0.1 mm [68]	Information not provided
Vicon	- Robust real-time tracking. - Accurate. - Easy to use - Portable.	- Deployment and tracking volume are quickly limited by the markers' locations. - Occlusions. - Line of sight.	25 µm [69]	19 µm [69]
Leica Laser tracker	- High speed in data acquisition. - Wide measurement range in large-scale metrology.	- Need a controlled environment. - Errors linked to the encoders. - Errors linked to geometrical and optical misalignment. - Light of sight.	~10 µm [70]	- 150 m -> measurement uncertainty of ± 10 µm. - 20 m -> ±0.3 µm/m. - 3 Km -> 0.6 mm. - 50 m -> 5 µm/m [73]
DJI LiDAR	- Portable - Use at any distances. - Height range of voxels providing multiple simultaneous range measurements. - Modulation frequency bandwidths.	- Multi-path interference error. - Phase wrapping. - Multiple propagation paths. - Aliasing. - Light of sight.	Between 0.04 m [71] and 0.88 mm depending of the distance and the wavelength used [72]	Up to 7 cm [72]
Microsoft's Kinect	- Quick generation of 3D information in real time. - A range of applications between 0.4 m and 7 m. - Cheap. - Portable.	- Sensitive to external infrared sources. - Not suitable for outdoor applications. - Detection of crystalline or highly reflective objects is not possible. - The precision decreases with range according to a second order polynomial equation. - Light of sight.	10 mm	- 8 mm in the range of under 1 m. - 25 mm in the range of 2 m [67].

2.2. Real camera definition

2.2.1.Real life camera

A camera is defined as a physical device or sensor used for recording visual images in the form of photographs, film, or video signals. It is generally composed of a lens, a shutter, and a sensor that captures and records visual information, as illustrated in Figure 2.1[74].

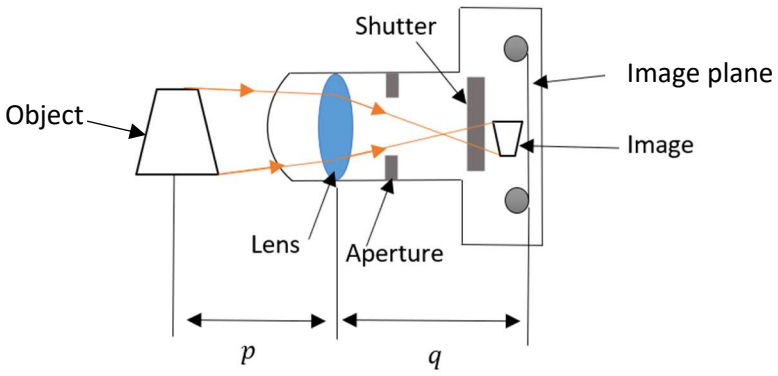


Figure 2.1: Camera's elements

2.2.2.Camera elements

A camera is defined by physical elements such as the lens, the focal length, or the shutter; but also, by no physical elements such as noise and lens distortion. These two elements are defined through the laws of optics, and the intrinsic physics of the camera component. As sources of error, they are part of the camera's signature.

The lens

The lens is a piece of glass or other transparent material with curved sides that concentrates or disperses light rays. Different types of lenses exist such as the thin lens, the concave lens, and the convex lens, with the lens being selected according to the focus length [75].

The focal length

The focal length, f, is the distance between the rear nodal point and the focal point of the lens, while the lens is focused to infinity. Its purpose is to take parallel rays of light and converge them into a single point of focus, F in Figure 2.2. The rays converging on F are recorded, either on film or, more commonly, with a digital sensor [76].

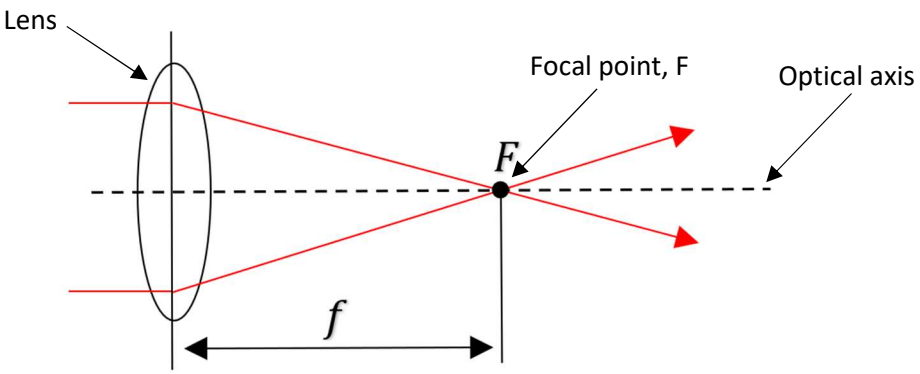


Figure 2.2: Focal length illustration

The aperture, depth of field and F-stop

The aperture is the opening in a lens through which light enters the camera. For a better understanding, the aperture and the pupil of the eye can be compared to each other, as they use the same mechanism. When a person moves from a dark to a bright environment, the iris dilates to bring in more light, and retracts to bring in less light, thus controlling the size of the pupil [77]. The aperture works in exactly the same way.

The F-stop or F-number represents the ratio of the lens focal length (f), to the diameter of the entrance pupil. An easy way to define it, is to look at it as a fraction of the aperture capacity. For instance, a F-number of $f/8$ mean that the aperture is open at $1/8$ of its total size [78].

The depth of field is one of the most important concepts in photography. It defines the distance between the closest and farthest objects in a photo and describes what parts of a photograph are in focus, i.e., which parts are clear/sharp, and what parts are out of focus, i.e., which parts are blurred, difficult to distinguish [77].

Depth of field and aperture are linked to each other by the F-stop number. When the aperture is large (the F-stop number is small), depth of field is limited, and vice versa.

The shutter speed

The shutter speed, also referred to keep as exposure time, is the length of time the camera shutter is open, exposing light onto the camera sensor. It is also defined as the time the camera spends taking a photograph [79].

The sensor

The sensor of the camera, called image sensor or imager, detects, and conveys information used to make an image, by converting the light waves into signals. Two types of camera sensors exist, CMOS (Complementary metal-oxide-semiconductor) and CCD (charge-coupled device).

Noise

Noise appears in an image as a grainy veil in a photograph, obscuring details and degrading the quality of the picture. Two different types exist: shot noise and digital noise. They are difficult to distinguish, and lead to the same result: pixels that are randomly too bright, too dark, or discoloured.

Shot noise, also known as photon noise, is randomness due to photons in the photographed scene, which are discreet and random. Digital noise, or electronic noise, is randomness caused by the camera sensor and internal electronics, which introduce imperfections within an image [80]. To measure the noise in the background of an image, signal-to-noise ratio can be used, as explained in Chapter 5, Section 5.1.2.

Lens distortion

The term "distortion" defines the symmetrical shift of pixels between the mathematical model of perspective projection and the physical model of image formation. In photography, distortion can be radial, tangential and/or both. Figure 2.3 shows the effect of distortion on a point P in 2D. In the case of radial distortion, this point moves radially with respect to its correct position (it moves upwards), whereas in the case of tangential distortion, the position of P is modified but remains on the same circle. Distortion can also come from various sources, such as imperfections in the lens [81].

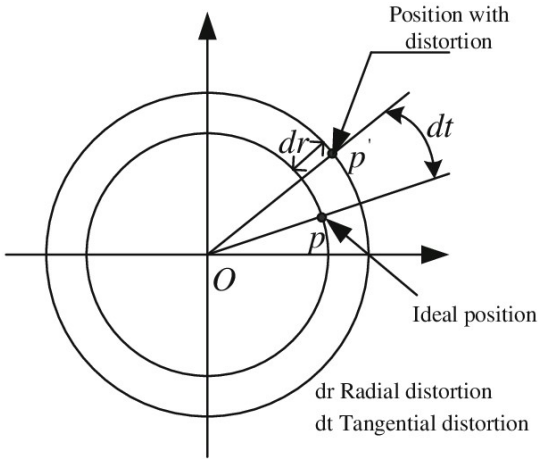


Figure 2.3: Representation of tangential and radial distortion

As illustrated above, radial distortion is caused by the defective radial curvature of the lens elements, i.e., the imperfect shape of the lens, while tangential distortion is caused by non-parallelism of the lens and the image plane.

The radial distortion is characterised by straight lines curved inwards or outwards in a barrel or a pincushion shape, and is expressed as (Equation 1):

$$\partial_{pr} = k_1\rho^3 + k_2\rho^5 + k_3\rho^7 + \cdots \qquad \text{Equation 1}$$

Where ρ is the radial distance from the principal point of the image plane, and $k_1, k_2, k_3, \ldots$ are the coefficients of radial distortion [82].

The general distortion equation, for a point with x and y coordinates is (Equation 2):

$$\begin{cases} x' = x(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1xy + p_2(r^2 + 2x^2) \\ y' = y(1 + k_1r^2 + k_2r^4 + k_3r^6) + p_1(r^2 + 2y^2) + 2p_2xy \end{cases} \qquad \text{Equation 2}$$

Where x' and y' are the coordinates of the point distorted; k_1, k_2, k_3 are the coefficients of radial distortion; p_1, p_2 are the coefficients of tangential distortion, and r is the radial distance [82].

Defining the key elements of the physical parameters, as well as the noise and the distortion, is a good start for designing a virtual camera, but not enough to understand how a camera works. Mathematical models are used to provide a description of a system using mathematical concepts and language. The functioning of a camera can be explained through diverse mathematical models such as the pinhole model and the panoramic model. The choice of the model depends on the type of camera used.

In this work, a raspberry Pi V2 camera is used. The reason behind this choice is explained in Chapter 3, Section 3.2. This camera works in a similar manner than a camera obscura and can be modelled with the pinhole model.

2.2.3.Pinhole model

A pinhole camera is a camera with a tiny aperture called a pinhole, rather than a lens. Light originating from the scene passes through the aperture and projects an inverted image onto an image plane, as illustrated in Figure 2.4.

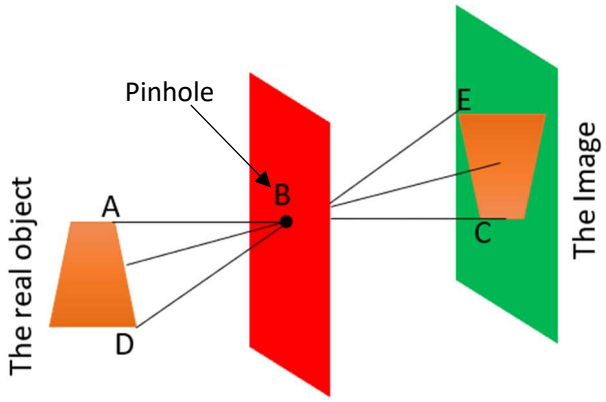


Figure 2.4: Image is formed

The image, defined by the points A and D, is inverted because the propagation of light is always in a straight line, so when rays of the object pass through the small hole, in B, they do not disperse and instead cross over to form the inverted image, so A is projected in C, and D in E [83]

The pinhole camera model is generally used to define a relationship between a 3D point and its corresponding 2D projection on the image plane. This geometric mapping formed is called perspective projection or triangulation. The point where all the rays intersect is called the optical centre or centre of the camera, and the point where the image plane intersects with the optical axis is called the principal point. From this configuration, intrinsic and extrinsic camera parameters can be calculated [83].

Intrinsic parameters

Intrinsic parameters describe the internal parameters of the camera and can be summarised in the intrinsic matrix K (Equation 3), without including the radial lens distortion:

$$K = \begin{pmatrix} f_x & s & O_x \\ 0 & f_y & O_y \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{Equation 3}$$

where f (f_x, f_y) is the focal length, s is the pixel skew and O (O_x, O_y) is the principal point.

These parameters describe a perspective projection, as illustrated with the geometry in Figure 2.5:

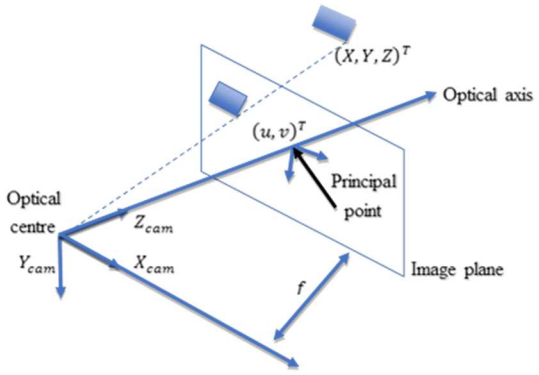


Figure 2.5: The ideal pinhole camera model

In Figure 2.5, the camera is represented by its optical centre. In addition, its optical axis is collinear with the Z_{cam} axis and the optical centre, is located at the origin of the camera reference frame. The 2D point (u, v) is the projection of the 3D world point $(X, Y, Z)^T$ onto the image plane and is written as:

$$u = \frac{Xf}{Z} \text{ and } v = \frac{Yf}{Z} \text{ with } f\text{: focal length}$$

By using the projective geometry framework, the projection of the 2D point in the 3D plane can be reformulated as Equation 4:

$$(\lambda_u, \lambda_v, \lambda)^T = (X_f, Y_f, Z)^T \qquad \textbf{Equation 4}$$

Converting to the matrix expression, Equation 4 becomes (Equation 5):

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \text{Equation 5}$$

With $\lambda = Z$ is the homogeneous scaling factor.

Equation 5 is true in world coordinates. However, the 3D points (reprojection of the 2D point in Figure 2.6) have to be expressed in the image coordinate frame.

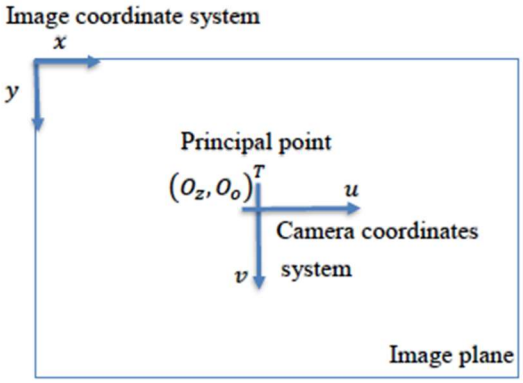


Figure 2.6: The image (x, y) and camera (u, v) coordinate system

Most imaging systems use the top left pixel of the image as the origin. However, in Figure 2.6, the frame origin corresponds to the principal point of the camera $(O_x, O_y)^T$. In an ideal camera, this point is the centre of the picture. However, in a normal camera, the position of this point changes according to the camera calibration.

Equation 6 results from the conversion of Equation 5 from world coordinates to image coordinates:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & s & O_x & 0 \\ 0 & f & O_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \text{Equation 6}$$

The parameter s, in Equation 6, correspond to the skew coefficients. This coefficient corresponds to the skewness of the pixels composing the image. A pixel is considered as skew, if its aspect ratio equal to 1:1. If the pixel is skey, $s = 0$; otherwise $s = f_x . \tan(\alpha)$, where α is the angle between the image axes.

In practice, the skew coefficient depends on the sensor manufacturer who typically defines the aspect ratio. Manufacturing and acquisition defects can result in pixel skew. These issues can be accounted for in the camera model using the parameters η and τ, which model the pixel aspect ratio and skew. The new expression with these two parameters is (Equation 7):

$$\lambda\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & \tau & O_x & 0 \\ 0 & \eta * f & O_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = [K\ O_3]\ P \qquad \text{Equation 7}$$

Where $P = (X, Y, Z, 1)^T$ are the3D points defined with homogeneous coordinates, and O_3 is the all zero-element vector

Extrinsic parameters

Extrinsic parameters give the external position and orientation of the camera in the 3D world. R is a 3 by 3 rotation matrix, giving the directions of the camera reference frame, in the world coordinate system [84, 85]. T is a 1 by 3 vector, expressing the camera translation, in the world coordinate system [86, 87].

2.3. Photogrammetry

The previous sections have given an overview of optical systems and how cameras work. The conclusion of these sections is that there are different measurement systems that can use markers (e.g., VICON and NIKON K-CMM), or that can be markerless by emitting light (laser trackers, TOF LIDAR, and Microsoft Kinect).

In addition, a brief explanation of each technology was given, identifying their advantages and disadvantages. At this stage, only the Microsoft Kinect appears to be a solution suited to the needs of tomorrow's manufacturing industry. However, another solution does exist, photogrammetry.

The aim of this section is therefore to present photogrammetry and compare it with the Microsoft Kinect, in order to determine which technology is more appropriate for this thesis.

2.3.1. Photogrammetry VS Microsoft Kinect

Photogrammetry is a method used for three-dimensional measurements in industrial and engineering works by recreating a 3D model of an object/room from a set of camera's images [88]. Employed as a measurement tool for, for instance, calibration, inspection, monitoring [89], it has never stopped evolving with the industry and the society.

Photogrammetry can be performed with one camera, two cameras, or with multi-cameras. Photogrammetry with two cameras is called Stereoscopy. Reproducing the stereovision, it allows for perceiving the third dimension of the scene, the depth.

A standard photogrammetric process contains three main phases (Figure 2.7):

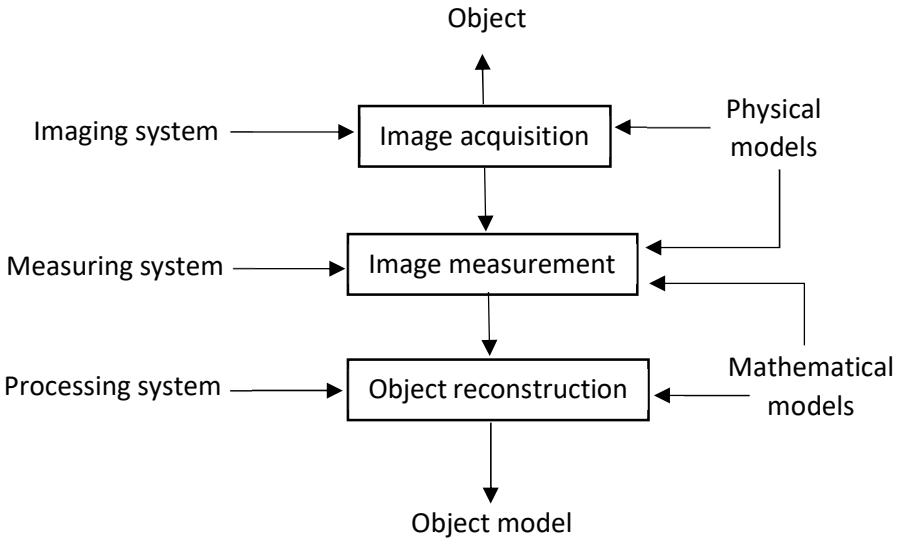


Figure 2.7: The photogrammetric process: from object to model [99].

Image acquisition is achieved with a Film or Digital camera. Image measurement is composed of image rectification, restitution and aerotriangulation (the process for determining the correct position and orientation of each image in a series of aerial images so they can be compiled into a map). Finally, examples of image reconstruction are orthophotos (geometrical correct), topographic maps and digital elevation model.

However, sources of error exist. They can be linked to the extraction of the point coordinates. An image is a 2D object, so only the x and y coordinates are available. The z coordinate is lost, removing the depth information. This information can be crucial to define the geometry and the boundary constraints limits for any optimisation problems. The errors come from the camera due to its pixel resolution, distortion, focal distance and noise. The radiometric map (colour map of the camera) can also generate errors. As the electromagnetic radiation recorded in the image is affected by the transmission medium (air, glass) and the light sensitivity of the sensor (film, electronic sensor), the colour map changes, which alters the colour and contrast of the image [88].

Photogrammetry is used in diverse areas. With the development of methods, needs and technologies, this method starts to be divided into different branches, and categorised in a multiplicity of ways:

- **Camera position and object distance** with close-range photogrammetry, where imaging distance is under 300 m.
- **Number of measurement images**, where Stereo photogrammetry uses dual image processing and stereoscopic measurement; and where multi-image photogrammetry uses n images (where $n > 2$, bundle adjustment).

- **Method of recording and processing** with analogue, analytical, digital photogrammetry, and line photogrammetry.
- **Availability of measurement results** namely offline, online and real-time photogrammetry.
- **Application or specialist area** such as engineering, industrial, biostereometric, multi-media, shape from stereo, and structure from motion [88].

These categories are merged into two main groups: Aerial photogrammetry, and Close range (or Terrestrial) photogrammetry.

Aerial photogrammetry is composed of three branches, which are analogue, analytical and digital photogrammetry. Analogue photogrammetry includes all methods and techniques to extract information from analogue photographs, based on mechanical and optical methods. Analytical photogrammetry is based on the reconstruction of camera's position as the previous one, but does not perform it mechanically. Digital photogrammetry is similar mathematically to the analogue one but use digital images [88].

Close range photogrammetry is based on the same theory as aerial photogrammetry but the pictures are taken at a short distance from the object of interest.

To perform photogrammetry, a set of cameras is needed as well as targets to be used as reference points, and to delimit the measurement volume. The choice of the camera is based on the application requirements, the camera cost and the camera performance. For the targets, it will depend on the applications and the type of camera optical system. Different types of targets exist. They can be retro-reflective spheres, circular plastic, spherical plastic, patterned targets or coded targets (like bar codes) [88].

Noted that retro-reflective targets are widely used in industrial applications, but they are sensitive to surface marking caused by for instance fingerprints or dust, and relative high manufacturing cost.

As previously explained, photogrammetry is a method recreating a 3D model of an object/room from a set of camera's images [88], whilst Microsoft Kinect is a device composed of an infrared (IR) Emitter, an RGB-D sensor, and an IR Depth sensor, performing tracking and detection applications.

The comparison between these two technologies is based on the comparison performed by Newhall [90], and is presented in Table 2.3.

Table 2.3: Comparison between the photogrammetry method and the Kinect V2 [90]

	Photogrammetry (Digital single-lens reflex (DSLR) Camera)	**Kinect V2**
Cost	Low to medium	Low
Portability	Good	Good, but requires USB 3 and external electricity supply
Object size	From small objects to entire rooms or building	Rather limited to objects
Indoor/Outdoor	Both	Indoor
Ease-of-use	Rather easy, needs care for photographs acquisition	Needs some practice
Processing time	Some hours all in all	Long, due to manual treatment
Accuracy	Very high, up to sub-millimetric (depends on the camera use)	About 10 mm

In Table 2.3, Microsoft's Kinect V2 performance is worse than the photogrammetry method with a DSLR camera. Its accuracy is about 10 mm, it is not easy to use, cannot be used outdoors and the detection is limited by the size of the object. On the contrary, the photogrammetric method seems to be limited only by the choice of the camera used, as well as uncertainty about ideal location of individual camera nodes.

In the factory of the future, monitoring systems have to be easy to use, cheap, flexible, and agile. Regarding results exposed in Table 2.3, photogrammetry seems to be the more suitable method to fulfil these requirements. Due to this conclusion, the mathematical models and concepts behind the photogrammetry principle are presented in the next sections for a better understanding of the results presented in Chapters 4, 5,6 and 7; as well as the sources of errors linked to it.

2.3.2. Triangulation

One of the outputs of photogrammetry is the 3D reconstruction. This result corresponds to the recreation of a 3D model of an object/room from a set of camera's images [88] based on triangulation, as shown in Figure 2.8.

By taking photographs from at least two different locations, so-called "lines of sight" can be developed from each camera to point on to the object. These lines of sight or rays are mathematically intersected to produce the 3-dimensional coordinates of the points of interest, as illustrated in Figure 2.9[91].

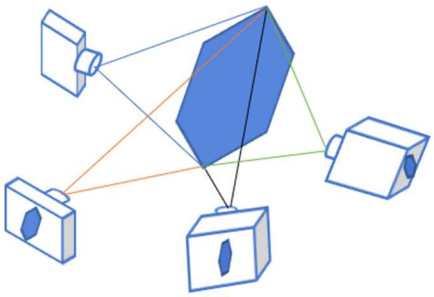


Figure 2.8: Illustration of terrestrial photogrammetry

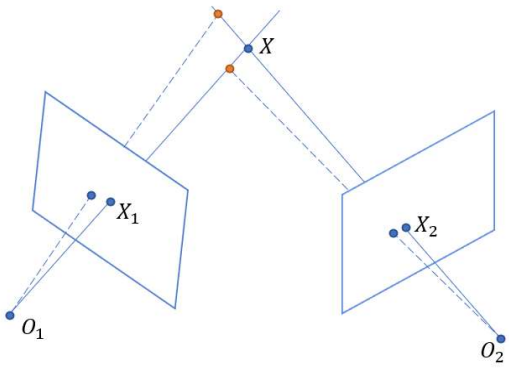


Figure 2.9: Illustration of the triangulation problem

The resulting triangulation problem is mathematically expressed as (Equation 8):

$$X = minimum\left(d^2(x_1,\ P_1X) + d^2(x_2, P_2X)\right) \qquad \text{Equation 8}$$

Where

d is the distance from x to X; x_1 and x_2 are the 2D representation of the point X in each camera's views; P_1 and P_2 are the reprojection matrices of camera 1 and 2; and X is the 3D point of the object in the scene.

And is solved by using the epipolar geometry, explained in the new Section.

2.3.3. Epipolar geometry

Epipolar geometry defines geometric relations between 3D points of the scene and their projections onto the 2D images created by the camera looking at the scene, as illustrated in Figure 2.10. Noted that the cameras are assumed to be pinhole cameras (see Section 2.2.3 about the pinhole model) [92].

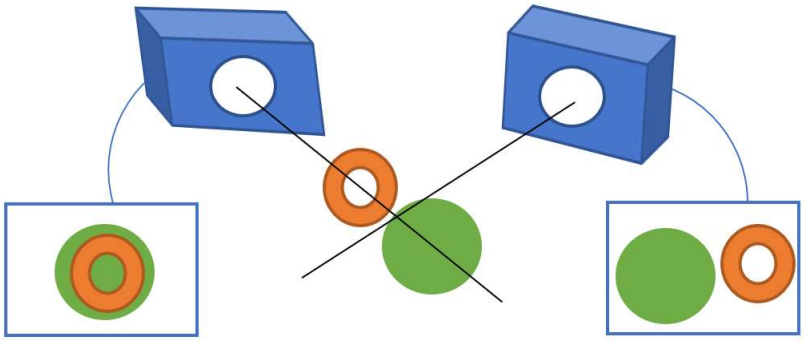


Figure 2.10: Principle of the epipolar geometry

One of the uses of epipolar geometry is to express the intrinsic ambiguity of the mapping from 3D to 2D. This ambiguity is defined by a loss of z-axis (depth) information due to the reconstruction of a 3D image with 2D information. The absence of this information results in a loss of perception of depth, texture, elevation and shape in the image, as shown in Figure 2.11.

Figure 2.11: Example of intrinsic ambiguity of the mapping from 3D to 2D

Furthermore, from a mathematical point of view, this results in a loss of geometric constraints, which leads to errors in solving regression problems due to an incorrect definition of the function space.

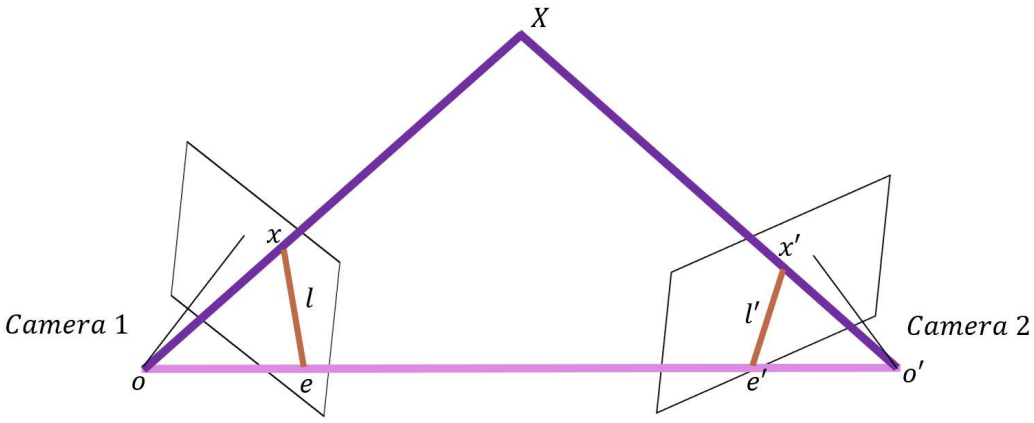


Figure 2.12: Epipolar geometry

In Figure 2.12, two cameras are represented, named camera 1 and 2, their optical centre is noted o and o'; and x and x' are the projection in the camera frame of point X, which belongs to the image frame and to the observable object.

The elements composing the epipolar geometry are:

- In pink, the baseline which is the line connecting the two camera centres.
- In violet, the epipolar plane which is the plane containing the baseline.
- In brown, the epipolar lines, denoted l and l', which are the intersections of the epipolar plan with the image planes (in white). A minimum of two epipolar lines are always represented due to the need for two cameras to define the epipolar geometry.
- Denoted as e and e', the epipoles are the intersections between the baseline and the image planes. They are the projection of the other camera centre in the current camera epipolar plane, i.e., e is the projection of o' in the camera 1 epipolar plane [93].

The epipolar constraint is the relationship between the epipolar lines and the projection of object points in the camera frame. Illustrated in Figure 2.13, this constraint can be translated as the obligation of the potential matches for p to lie on the corresponding epipolar line l', and same for p' with l.

This constraint is an important tool allowing simplification of calculations and to validate triangulation. Moreover, it is the starting point to define the collinearity equations as well as the stochastic model of the camera system.

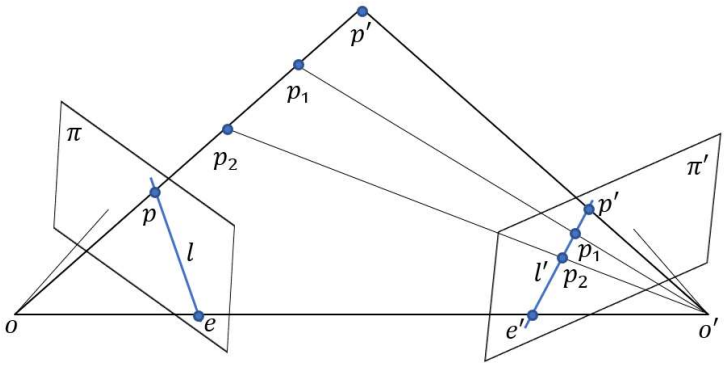


Figure 2.13: Illustration of the epipolar constraint

2.3.4.Collinearity equations (Functional model)

Collinearity equations reflect the nature of the observations, such as 2-D image coordinates or 3-D model coordinates, and the particular problem that one wishes to solve, such as space resection or relative orientation [94].

In an ideal photogrammetry system, shown in Figure 2.14, a pinhole camera, and two coordinate systems related to the image (x', y') and the object (x, y, z) are set up. The object point p is projected in the image coordinate system through the camera in p', and $p'$is called the corresponding image point of p.

The point o' is the origin point, i.e., it is the projection of the camera centre o in the image coordinate system, with the focal length f, between them on the z axis.

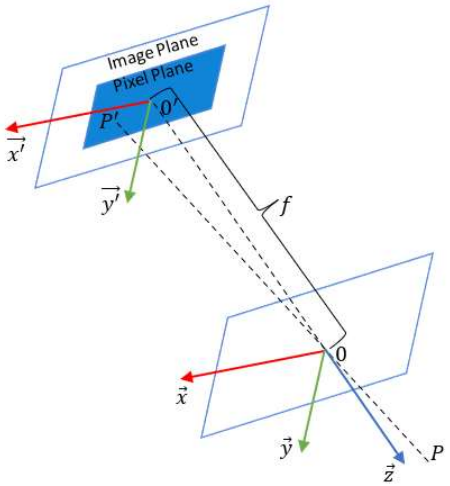


Figure 2.14: Photogrammetry system

If a simple version of Figure 2.14 (Figure 2.15) is taken as the study case, it leads to the result defined in Equation 9, expressing location along a line in space of each object point, their corresponding image point and the perspective centre of the camera lens.

$$a = k.A \qquad \text{Equation 9}$$

Where k is a scale factor; and a and A are the image space vector and object space vector respectively.

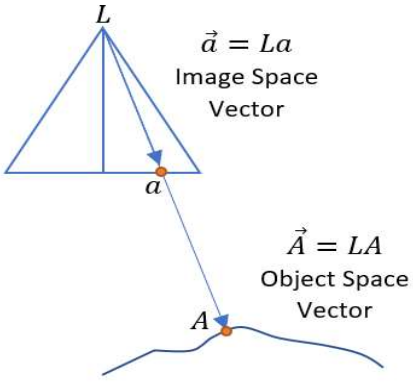


Figure 2.15: Simple version of Figure 2.14 where L is the optical centre of the camera, a, the image point and A, the object point

Equation 9 is valid only if the two vectors are expressed in the same coordinate system. However, a is expressed in the image coordinate system (Equation 10):

$$a = \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix} \qquad \text{Equation 10}$$

Where x_0, y_0, z_0 are the coordinates of the origin of the image coordinate system, and f, the focal length.

And A is expressed in the object coordinate system (Equation 11):

$$A = \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix} \qquad \text{Equation 11}$$

Where X_L, Y_L, Z_L are the coordinates of the camera perspective centre, noted L in Figure 2.15.

As a result, one of these two vectors have to be expressed in the other coordinate system. The transformation applied to the object space vector, A gives the collinearity model [94] (Equation 12):

$$\begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix} = \lambda . R^T \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix} \qquad \text{Equation 12}$$

Where λ is a scale factor, and R, the rotation matrix of the camera.

By substituting the focal length f from the third row into the first two rows of Equation 12, the collinearity model is rewritten as (Equation 13):

$$\begin{cases} x = x_0 - f.\frac{N_x}{D} \\ y = y_0 - f.\frac{N_y}{D} \end{cases} \qquad \text{Equation 13}$$

Where:

$$\begin{cases} N_x = r_{11}(X - X_L) + r_{21}(Y - Y_L) + r_{31}(Z - Z_L) \\ N_y = r_{12}(X - X_L) + r_{22}(Y - Y_L) + r_{32}(Z - Z_L) \\ D = r_{13}(X - X_L) + r_{23}(Y - Y_L) + r_{33}(Z - Z_L) \end{cases}$$

However, Equation 13 is non-linear and needs to be linearised with the Least-square adjustment, before being used to solve:

- Space resection problem:
  camera exterior orientation unknown, object points known, observed image coordinates given, usually implying a single image.
- Space intersection problem:
  camera exterior orientations known, object point unknown, observed image coordinates given, usually implying a single object point.
- Bundle block adjustment problem:
  simultaneous resection and intersection, multiple images, and multiple points [95].

However, Equation 13Equation 12 only defines the relationship between the point of an object and its projection in the view of a single camera. In photogrammetry, several points and several cameras are used. Thus, the coplanarity condition equation has to be used in addition to the collinearity equations.

2.3.5. Coplanarity condition equation

Figure 2.16 represents a pair of rays, $\overrightarrow{a_1}$ and $\overrightarrow{a_2}$, defined by two conjugate image points. From this configuration, a plane is defined between the pair of rays and the base vector, $\vec{b}$.

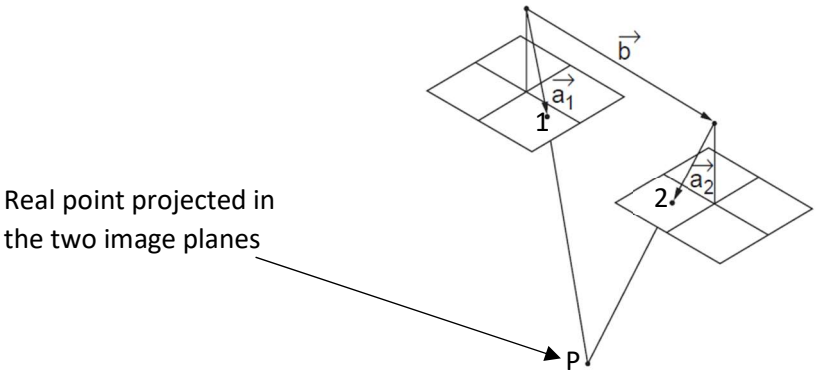


Figure 2.16: Illustration of the plane formed between the pair of rays and the base vector

Moreover, this geometrical configuration is reinforced by the coplanarity condition, and then, these three vectors are coplanar. This can also be proven by the triple scalar product between them equal to zero, or by the nullity of the volume of the parallelepiped defined by the three vectors.

By using the collinearity model defined in Equation 13, a_1 and a_2 are expressed as follows (Equation 14):

$$\begin{cases} a_1 = \begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = R_1^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_1 \\ a_2 = \begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = R_2^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_2 \end{cases} \qquad \text{Equation 14}$$

On the other hand, the base vector b is expressed as (Equation 15):

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} X_{L2} - X_{L1} \\ Y_{L2} - Y_{L1} \\ Z_{L2} - Z_{L1} \end{bmatrix} \qquad \text{Equation 15}$$

where point 1 has the coordinates $(X_{L1}, Y_{L1}, Z_{L1})^T$ and point 2, $(X_{L2}, Y_{L2}, Z_{L2})^T$

The coplanarity condition equation is satisfied for (Equation 16):

$$b.(a_1 \times a_2) = 0 \qquad \text{Equation 16}$$

And the triple-scalar product is expressed as (Equation 17):

$$F = det \begin{vmatrix} b_x & b_y & b_z \\ u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \end{vmatrix} = 0 \qquad \text{Equation 17}$$

Where the determinant of F is (Equation 18):

$$b_x.(v_1.w_2 - v_2.w_1) + b_y(u_2.w_1 - u_1.w_2) + b_z(u_1.v_2 - v_1.u_2) = 0 \qquad \text{Equation 18}$$

The most prominent application of Equation 18 is the relative orientation problem. However, the equation is non-linear and usually linearised with a linear approximation [94].

2.3.6. Least-square adjustment and linearisation of the collinearity equations

As explained in Section 2.3.4, Equation 13 is non-linear and needs to be linearised with the Least-square adjustment. Before doing this through the linearisation process, the Least-square adjustment method needs to be introduced.

2.3.6.1. Least-square adjustment method

The Least squares adjustment method is an iterative method, which may be used in short range photogrammetry because it uses statistical analysis based on a large number of redundant measurements of different types and weights to estimate the most likely coordinates for connected points in a measurement in a network.

As partial derivations are used, only approximate values of the image observations, and estimated image observations can be determined. However, partial derivatives are used when the function cannot be a full derivative due to multi dependencies. Then, this mathematical tool defines how a multivariable function changes as you tweak just one of the variables in its input. By alternately keeping one of these dependencies non-constant and setting the other constants to derive the function according to the latter, the inter-dependencies between these values can be missed, generating errors in the final result. Correction vectors have then to be introduced to solve this problem.

In this case, the correction vectors $\hat{x}$ is defined for each of the image observations, and has to be computed with the observation equations, to estimate the values of each parameter.

An observation is defined as a measured variable which has a variance-covariance; where the unknown parameters are the unknown variables wanted to determine.

The observation equations are defined for each observation, and are written in terms of the unknown parameters. These equations (Equation 19) are weighted according to the precision of the observations in order to return the errors back to their sources.

$$X = (J^T W J)^{-1} J^T W K \qquad \text{Equation 19}$$

Where J is the Jacobian matrix (matrix of partial derivatives), W is the weights for each observation, K is a vector of the differences between the observed and computed values for each measurement, and X is the matrix of corrections.

It is noted that Equation 19 minimises the sum of squares of the residuals and gives the most likely solution for any set of data. The least squares adjustment method can be seen as a method of solving

an overdetermined system of equations that gives the most probable values for the unknown parameters. In this process, all geometric constraints are satisfied [96].

The first step of the least square adjustment method is the formulation of the functional model [97] (Equation 20):

$$F(x, b) = 0 \qquad \text{Equation 20}$$

Where x is the unknown element to be estimated and b the m measured elements.

Usually, measurements are assumed to be independent random variables. Each measurement b_i has a standard deviation σ_i and a weight w_i defined as (Equation 21):

$$w_i = \frac{\sigma_0^2}{\sigma_i^2} \qquad \text{Equation 21}$$

Where σ_0^2 is the variance factor, or reference variance, or standard deviation of a measurement in weight unit.

The weight matrix, W, is a $m \times m$ diagonal matrix. The non-zero elements are the weight w_i. The covariance matrix is composed of independent measurements, C_b, and is a $m \times m$ diagonal matrix, where the non-zero elements are the variance σ_i^2.

The cofactor matrix Q is defined as (Equation 22):

$$\sigma_0^{-2}.C \qquad \text{Equation 22}$$

Where its independent measured elements, b, are (Equation 23):

$$Q_b = \sigma_0^{-2}.C_b = W^{-1} \qquad \text{Equation 23}$$

Using the weight matrix, in Equation 23, helps to overcome the difficulty of having redundant and inconsistent numerical data.

In the least square adjustment method, the linearisation of the functional model is an important step. It allows the removal of possible problems created by the approximation of the random variables, and to determine the unique solution by iteration.

Using Taylor's theorem, the linearisation of the functional model defined in Equation 20 is (Equation 24):

$$F(x, b) = F(x_0, b_0) + \left(\frac{\partial F}{\partial x}\right)_0 (x - x_0) + \left(\frac{\partial F}{\partial b}\right)_0 (b - b_0) = 0 \qquad \text{Equation 24}$$

Equation 24 can be expressed at the first order as (Equation 25):

$$F(\bar{x},\bar{b}) = F(x_0,b_0) + A(\bar{x} - x_0) + D(\bar{b} - b_0) = 0 \qquad \text{Equation 25}$$

Where $\bar{x}$ and $\bar{b}$ represent values that fit the functional model exactly; and x_0 and b_0 are first order approximations to x and b.

v can be defined as the corrections to be carried out to correct the measurements in order to fully satisfy the functional model, as (Equation 26):

$$v = \bar{b} - b_0 \qquad \text{Equation 26}$$

Noting that these corrections are numerically the same as the measurement residuals, but with an opposite sign.

By introducing $d = -F(x_0,b_0)$, Equation 26 can be rewritten as (Equation 27):

$$A(\bar{x} - x_0) + Dv = d \qquad \text{Equation 27}$$

The unique least squares estimate of $\bar{x}$ and $\bar{b}$, $\hat{x}$ and $\hat{b}$ are those that satisfy the weighted least squares criterion (Equation 28):

$$\begin{gathered} \varphi = v^t W v \rightarrow \text{Minimum} \\ or \\ \varphi = v^t W v + 2k^t(Ax + Dv - d) \rightarrow \text{Minimum} \end{gathered} \qquad \text{Equation 28}$$

Where k is a column vector of Lagrangian multipliers. The vector defined in Equation 28 is introduced to help to determine x and v.

$$\begin{bmatrix} W & D^t & 0 \\ D & 0 & A \\ 0 & A^t & 0 \end{bmatrix} \begin{bmatrix} v \\ k \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix} \qquad \text{Equation 29}$$

Where x and v are:

$$\begin{cases} x = [A^t(DW^{-1}D^t)^{-1}A]^{-1}A^t(DW^{-1}D^t)^{-1}d \\ v = W^{-1}D^t(DW^{-1}D^t)^{-1}\{I - A[A^t(DW^{-1}D^t)^{-1}A]^{-1}A^t(DW^{-1}D^t)^{-1}\}d \end{cases} \qquad \text{Equation 30}$$

To simply the equations, a rearrangement of the functional relationships between measured and unknown elements is completed. This procedure greatly simplifies least squares processes and makes them more flexible. Equation 30 becomes (Equation 31):

$$F(x,b) = f(x) - b = 0 \qquad \text{Equation 31}$$

And it reduces to (Equation 32) [97]:

$$Ax = b_0 - f(x_0) + v \qquad \text{Equation 32}$$

Equation 32 is the final form of the Least square method, used to calculate the stochastic model in Section 2.4.1.2. However before determining it, the collinearity equations have to be linearised.

2.3.6.2. Linearisation of the collinearity equations

By using the Taylor method, the linearisation of the collinearity equation is achieved. It is noted that the equations are not linear, and hence the partial derivative will be used. The steps of the process are:

1. The collinearity equations are (Equation 33):

$$\begin{cases} f_x = \ x - x_0 = -f\dfrac{r_{11}(X-X_0)+r_{12}(Y-Y_0)+r_{13}(Z-Z_0)}{r_{31}(X-X_0)+r_{32}(Y-Y_0)+r_{33}(Z-Z_0)} \\ f_y = \ y - y_0 = -f\dfrac{r_{21}(X-X_0)+r_{22}(Y-Y_0)+r_{23}(Z-Z_0)}{r_{31}(X-X_0)+r_{32}(Y-Y_0)+r_{33}(Z-Z_0)} \end{cases} \quad \text{Equation 33}$$

2. The linearisation assumptions are:
   a. Object points are constant.
   b. Object coordinates are considering as measured variables (called observation in Section 4.5.4.1).

3. The Taylor's series expansion is used to linearise Equation 33, and the terms with a higher order than 1 are neglected (Equation 34):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_x}{\partial \omega} & \dfrac{\partial f_x}{\partial \varphi} & \dfrac{\partial f_x}{\partial \kappa} & \dfrac{\partial f_x}{\partial X_0} & \dfrac{\partial f_x}{\partial Y_0} & \dfrac{\partial f_x}{\partial Z_0} \\ \dfrac{\partial f_y}{\partial \omega} & \dfrac{\partial f_y}{\partial \varphi} & \dfrac{\partial f_y}{\partial \kappa} & \dfrac{\partial f_y}{\partial X_0} & \dfrac{\partial f_y}{\partial Y_0} & \dfrac{\partial f_y}{\partial Z_0} \end{bmatrix} \begin{bmatrix} d\omega \\ d\varphi \\ d\kappa \\ dX_0 \\ dY_0 \\ dZ_0 \end{bmatrix} + \begin{bmatrix} f_x^0 \\ f_y^0 \end{bmatrix} \quad \text{Equation 34}$$

In which, $\begin{bmatrix} f_x^0 \\ f_y^0 \end{bmatrix} = \begin{bmatrix} f_x(\omega^0, \varphi^0, \kappa^0, X_0^0, Y_0^0, Z_0^0, X, Y, Z) \\ f_y(\omega^0, \varphi^0, \kappa^0, X_0^0, Y_0^0, Z_0^0, X, Y, Z) \end{bmatrix}$

Where ω, φ and κ are the rotation angles, X, Y and Z are the coordinates of a 3D object point, and X_0, Y_0 and Z_0, the coordinates of the projection centre.

These values were calculated using the current approximate values of parameters, and are considered as constant. Moreover, the matrix of the partial derivatives is called the Jacobian matrix, and the column vector composed of the total derivate, is called the correction vector.

4. The error equation is defined in Equation 35, where the residuals of v_x, v_y are calculated:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_x}{\partial \omega} & \dfrac{\partial f_x}{\partial \varphi} & \dfrac{\partial f_x}{\partial \kappa} & \dfrac{\partial f_x}{\partial X_0} & \dfrac{\partial f_x}{\partial Y_0} & \dfrac{\partial f_x}{\partial Z_0} \\ \dfrac{\partial f_y}{\partial \omega} & \dfrac{\partial f_y}{\partial \varphi} & \dfrac{\partial f_y}{\partial \kappa} & \dfrac{\partial f_y}{\partial X_0} & \dfrac{\partial f_y}{\partial Y_0} & \dfrac{\partial f_y}{\partial Z_0} \end{bmatrix} \begin{bmatrix} d\omega \\ d\varphi \\ d\kappa \\ dX_0 \\ dY_0 \\ dZ_0 \end{bmatrix} - \begin{bmatrix} x - f_x^0 \\ y - f_y^0 \end{bmatrix} \quad \text{Equation 35}$$

Where $v = Ax - y$, and v_x, v_y are image observations.

5. After the linearisation, the collinearity equations are written as (Equation 36):

$$\begin{cases} F_x = -f.\frac{U}{W} \\ F_y = -f.\frac{V}{W} \end{cases} \quad \text{Equation 36}$$

Where $\begin{bmatrix} U \\ V \\ W \end{bmatrix} = R.\begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}$

2.3.7.Camera and Photogrammetry Summary

Over the last twenty-eight pages, the following notions have been covered:

- Industry 4.0 and Smart Industry.
- Real camera's components, and the pinhole model used to model them.
- Photogrammetry and comparison with the Microsoft Kinect to determine the more suitable technique to use for this work.
- Mathematical theory of photogrammetry, such as the epipolar geometry, or the triangulation.

With the following key points:

- Industry 4.0 has generated new needs in terms of safety, object and people tracking, and optimisation. With the development of this new model of industry, the current factories have started to change and to called Smart Factory or Factory fo the Future. These new models are charaterised by implementation of new technologies such as 3D printing, artificial intelligence, digital twin, and sensors.

- Optical systems based on multi-camera networks are being introduced because they are cheap, easy to use, and not intrusive. Examples of this technology are the Vicon system (marker system), the Kinect V2 (marker-less system), or the photogrammetry method where multiple cameras take pictures of an object from different points of view and perform 3D reconstruction from the 2D images.

- However, optical systems have drawbacks linked to the placement of markers, the line of sight, the type of camera used, and the tracking volume which can be quite small.

- A real camera is composed of physical elements such as the lens, the focal length, or the shutter; but also, by no physical elements such as noise and lens distortion.

- The model used to model a camera, and to calculate its extrinsic and intrinsic parameters, is a pinhole model.

  Photogrammetry is a method uses for 3D measurements in industrial and engineering works by recreating a 3D model of an object/room from a set of camera's images. The 3D reconstruction is based on the triangulation, and the epipolar geometry, and the equations (from the triangulation and epipolar geometry) coded for the following experimental Chapters (4, 5, 6 and 7) are listed in the following Table (**Error! Reference source not found.**.4):

Table 2.4: Main equations used in the MATLAB algorithms

Intrinsic parameters (Equation 3)	$K = \begin{pmatrix} f_x & s & O_x \\ 0 & f_y & O_y \\ 0 & 0 & 1 \end{pmatrix}$
Triangulation problem (Equation 8)	Find X that minimizes: $d^2(x_1, P_{1X}) + d^2(x_2, P_{2X})$
Collinearity model (Equation 12)	$\begin{bmatrix} \mathrm{x} - \mathrm{x}_0 \\ \mathrm{y} - \mathrm{y}_0 \\ -\mathrm{f} \end{bmatrix} = \lambda . \mathrm{R}^{\mathrm{T}} \begin{bmatrix} \mathrm{X} - \mathrm{X}_{\mathrm{L}} \\ \mathrm{Y} - \mathrm{Y}_{\mathrm{L}} \\ \mathrm{Z} - \mathrm{Z}_{\mathrm{L}} \end{bmatrix}$ Where λ is a scale factor, and R, the rotation matrix of the camera
Coplanarity condition (Equation 14 and Equation 16)	$b.(a_1 \times a_2) = 0$ Where: $\begin{cases} a_1 = \begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = R_1^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_1 \\ a_2 = \begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = R_2^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_2 \end{cases}$
Least square (Equation 27)	$Ax = b_0 - f(x_0) + v$
Collinearity equations after linearisation (Equation 36)	$\begin{cases} F_x = -f.\frac{U}{W} \\ F_y = -f.\frac{V}{W} \end{cases}$ Where $\begin{bmatrix} U \\ V \\ W \end{bmatrix} = R. \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}$

2.4. Camera calibration methods

In metrology and measurement technology, calibration is the process of configuring an instrument to provide a result for a sample within an acceptable range. In machine vision, calibration is used to establish a mathematical model and solve the parameters of the camera through the correspondence between a series of scene points and pixel points. Various methods exist for camera calibration. The principal ones include traditional visual calibration algorithms, camera self-calibration algorithms, and active-vision-based calibration algorithms [98].

Traditional calibration methods

Traditional calibration methods involve calculating the inner and outer parameters of the camera through mathematical transformations, using the measurement of a calibration object such as length or height as a reference. These mathematical transformations are applied to a camera image that has already been processed and calculated under a certain camera model [99].

These methods can be applied to any camera model, and provide high accuracy calibration results. Different algorithms exist such as the direct linear transformation method (DLT method), nonlinear optimization method, two-step method, planar template method, and the dual-plane method [100].

The direct linear transformation method was developed by Abdel-Aziz and Karara in 1971 and consists of defining a set of intermediate parameters to solve linear equations, without iterative calculations, to find the camera parameters. The advantage of this method is its ease of calculation, and implementation due to the few parameters involved. However, non-linear distortion problems are not taken into consideration, although the accuracy of this method can be improved by including these non-linear factors in the calculation process. They can be calculated through a non-linear means [100].

The nonlinear optimization technique is similar to the direct linear technique but considers the lens distortion of the camera and uses iterative algorithms to solve non-linear equations. Using a large number of unknowns, and large-scale nonlinear optimization, it is a computational cost method but with high precision. A disadvantage of this method is its iterative nature, without a good initial estimate, the optimization process may be unstable, resulting in instability [99].

Two-step calibration is a solution to overcome the drawbacks of the linear solution, which does not consider the lens distortion, and the non-linear solution, which take into consideration the lens distortion but make the calculations much more complicated, with a result that is sometimes unstable and wrong [123]. The two-step calibration was proposed by Tsai in 1986 and consists of first solving the linear systems using the perspective matrix transformation method, before using the background parameters as the initial value to solve the non-linear part of the problem, i.e., the distortion factor, and finally using the optimisation methods a second time to improve the accuracy [100].

Its advantage is that the model assumed that the camera lens distortion was radial, and the vector remained unchanged from the image centre to the point direction of the image, regardless of changes in distortion. Reducing at the same time, the space dimension of parameters, it is a suitable solution for obtaining accurate measurements. However, with this method, the calibration of the equipment requirements is relatively high, not suitable to use on a simplified calibration [100]. Moreover, only the radial distortion is modelled, not the tangential.

The biplane method is similar to the pinhole model but does not require that all the light projected onto the surface pass through the optical centre of the light. This method does not explicitly use the camera model, but rather appears to follow the line of sight of the world coordinate system. For a given image point in an image plane, this method is able to calculate the corresponding points between the two calibration planes, and determine the lights used to generate the image point in the image plane. With the reference points, it is possible to calculate both planes using the insertion method. With the biplane method, it is possible to calculate relevant parameters by using a linear method, but it requires a large number of unknown parameters and tends to over-parameterize the problem [99].

Self-calibration technology

Self-calibration technology is used when it is impossible to use a calibration reference object for camera. The camera calibration is achieved through comparing the relationship between the images of the surrounding environment. This calibration process may be appropriate in cases where the intrinsic and extrinsic parameters of the camera are not fixed [100].

Currently many self-calibrations methods exist such as self-calibration method of directly solving Kruppa equation; hierarchical stepwise calibration method; self-calibration method based on the absolute quadric surface; module constraint method of Pollefeys [99].

Compared to traditional methods, this global approach only requires the establishment of the image match. This is a flexible but complex process, which cannot be involved in the real-time updating opportunity. However, this method is suitable for situations of lower accuracy such as virtual reality.

Active-vision-based calibration algorithm

Active vision calibration has been developed to overcome the cumbersome process of traditional calibration method. In an active vison-based calibration, the camera system is accurately installed in a controllable platform and, through the active control of the camera, multi-images are taken and used to determine the camera intrinsic reference and extrinsic reference parameters. At present, two main methods exist, the calibration based on three-orthogonal translational motion and the orthogonal motion method based on planar homography matrix [99].

The advantage of these methods is the use of simple algorithms. The disadvantage is the rigidity of the camera system. The number of rotations and translations of the camera are set manually, which limits the range of applications of the algorithm.

This problem comes from its design. The focus is put on solving the linear problem of the camera model parameters while minimizing camera movement restrictions. Without this restriction, the problem becomes a multivariate nonlinear optimization problem, and the calibration based on active vision is a self-calibration [100].

Traditional visual calibration algorithms, camera self-calibration algorithms, and active-vision-based calibration algorithms are the principal ways to perform calibration. However, a variety of other calibration tools exist, based on calibration artefact with specific pattern such as the checkerboard or zone circles boards, neural network or genetic algorithms.

Calibration tools based on pattern use graphic templates. Graphic templates were first proposed by Zhang Zhengyou, and composed of a drawn dot matrix. A camera took images of the template created from different orientations, generated a set of images. By matching these images with the template, homography matrix can be calculated, and the camera parameters can be calculated by solving linear problems with this matrix. This method takes into consideration the lens distortion, radial and tangential, but the process of calibration can be challenging, needing the extraction of the image point to match them with the template ones [100].

Nowadays this method is wide spread. The templates are diverse, going from circles to chess patterns, with different sizes [101]. Even MATLAB has developed a toolbox to perform camera calibration based on this type of calibration, including all the existing patterns [102].

With the development of artificial intelligence and deep learning, neural network solutions have started to be used for camera calibration to solve nonlinear models [100]. Methods such as nonlinear fitting algorithms [102], back propagation neural network optimized by genetic algorithm [100], or genetic algorithm by itself [103] are often used to solve them. This type of network can achieve accurate approximation of complex non-linear mapping. It is simple, practical, and needs less computational power than other calibration methods. However, the minimum found by a neural network applied in camera calibration maybe the local minimum, not necessarily the global one. The local minimum is a point where the value of the function is smaller than that of nearby points, but possibly still larger than that of a distant point. The global minimum is a point where the value of the function is smaller than that of all other constituent points of the function. Confusing the local minimum and the global minimum can cause convergence problems.

A large diversity of calibration methods exists, from traditional methods to the active-vision-based algorithms, via self-calibration technology. Different calibration artefacts exist such as spheres to calibrate the CMM (Coordinate-measuring machine) [104], checkerboard [105], or zone circle board [106]. The calibration process involves measuring objects shape in camera images with a set of algorithms developed in MATLAB. However, the outputs of the calibration might not be optimised or can be refined. Methods such as the Bundle adjustment based on the Levenberg-Marquardt algorithm, can be used to refine the result by recalculating the camera extrinsic parameters, presented in the following section.

In addition, other methods can be used in the process of measurement to the error, and their outputs used as an offset to correct the calibration parameter of the device, and/or as an indicator of the quality of the measurement according to external parameters.

2.4.1.Bundle Adjustment (Levenberg-Marquardt algorithm)

The Levenberg-Marquardt algorithm was created by Levenberg, in 1944, before being modified by Marquardt, in 1963. This method minimises the sum of the squares of the errors between the model function and a set of data points. If the model function is linear in its parameters, the least squares problem is quadratic in the same parameters. Otherwise, the least squares problem requires an iterative solution algorithm. The minimum found is a local minimum, which is not necessarily the global minimum [107-108].

The Levenberg-Marquardt algorithm is based on two numerical minimization algorithms:

- The Gauss-Newton method.
- The gradient descent method.

When the current solution is far from the correct one, the algorithm behaves like a gradient descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method [107-108]. This algorithm is robust but computationally expensive.

To try to reduce the size of the least squares problem, several solutions exist:

- Physically:
  For example, reduce the size of the problem by minimizing the number of parameters used, or the size of the matrices used.
- Mathematically:
  For example, using sparse methods.

2.4.1.1. Newton's method

Newton's method, also known as the Newton-Raphson method, is a root-finding method that uses the first few terms of the Taylor series of a function $f(x)$ in the vicinity of a suspected root.

For any function $f(x)$, the Taylor series about the point $x = x_0 + \epsilon$ is (Equation 37):

$$f(x_0 + \epsilon) = f(x_0) + f'(x_0) * \epsilon + \frac{1}{2} f''(x_0)\epsilon^2 + \theta(\epsilon) \qquad \text{Equation 37}$$

where $\theta(\epsilon)$ is the other terms of the Taylor series.

If the first order terms are kept, Equation 37 becomes (Equation 38):

$$f(x_0 + \epsilon) \approx f(x_0) + f'(x_0)\epsilon \qquad \text{Equation 38}$$

Equation 38 corresponds to the equation of the tangent line to the curve at $\big(x_0, f(x_0)\big)$, then $(x_1, 0)$, is the point where the tangent line intersects the x-axis. This equation is also used to estimate the amount of offset ϵ needed to land closer to the root starting from an initial guess x_0.

Setting $f(x_0 + \epsilon) = 0$, and solving for $\epsilon \equiv \epsilon_0$, Equation 38 becomes (Equation 39):

$$\epsilon_0 = -\frac{f(x_0)}{f'(x_0)} \qquad \text{Equation 39}$$

Equation 39 gives the first order adjustment to the root's position. If Equation 39 is put at the n order, the recursive equation obtained is (Equation 40):

$$\epsilon_n = -\frac{f(x_n)}{f'(x_n)} \text{ for } n = 1,2,3,\ldots \qquad \text{Equation 40}$$

However, Equation 40 can be unstable near a horizontal asymptote or a local extremum. To avoid that, the value of the previse root calculated needs to be added to the previous equation [109] (Equation 41):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{Equation 41}$$

2.4.1.2. Gradient descent

The gradient descent is an optimisation algorithm used to minimize some function by iteratively moving in the direction of steepest descent (Equation 42) [109].

$$\text{Evaluated at } \theta^0\text{: } \theta^1 = \theta^0 - \alpha . \nabla J(\theta) \qquad \text{Equation 42}$$

where θ^0 is the current position, θ^1 is the next position, α is the small step and $\nabla J(\theta)$ is the direction of fastest increase.

The gradient descent is based on the minimisation of the cost function of the problem. A cost/loss function is often used to evaluate the performance of machine learning algorithms, which identifies how good the model is at making predictions for different parameters. The loss function computes the error for a single training example. The cost function is the average of the loss functions for all the training examples.

Any cost functions can be defined as the sum of the square difference between a known value and its prediction. A cost function for N points can be defined as follows (Equation 43):

$$cost = \frac{1}{N} . \sum_{i=1}^{N} (Y' - Y)^2 \qquad \text{Equation 43}$$

where Y' is the predicted value and Y the actual value. Y and Y' are defined by the line equation: $mx + b$, where m is the director coefficient, and b is the intersection of the line with the y-axis.

The squared difference is taken in Equation 43 and not the absolute difference because:

- It is easier to derive a regression line.
- It increases the error distance between the predicted value Y', and the actual value Y, making the bad predictions more pronounced than the good ones.

The aim of the gradient is to minimise Equation 43, i.e., to find m and b for which the error is minimum.

To minimise Equation 43, the equation of the tangent (Equation 44) is used because $(Y' - Y)^2$ is in a quadratic form. If it is drawn, the curve will be of the form of $-x^2$, having its minimum close to the bottom of the curve.

$$\text{In a point } p,\ y = f'(p)(x - p) + f(p) \qquad \text{Equation 44}$$

where $f'(p)$ is the derivate of p, and $f(p)$ is the function associated.

As a consequence, the only mathematical tool to reach the global minimum (by moving from one end of the curve to its minimum, downwards) is the derivative. Derivatives are also used to increase or

decrease the parameters/weight of the function in order to increase or decrease any objective function.

If δ is defined as the variation, then new definition of m and b is:

$$\begin{cases} m = \delta m \\ b = \delta b \end{cases}$$

The research of the minimum starts by the definition of the cost function. Here Equation 43 can be written as following to highlight the connection between its elements:

$$J_{m,b} = \frac{1}{N}.\sum_{i=1}^{N}(Error)^2 \qquad \text{Equation 45}$$

where $J_{m,b}$ is the cost function expressed according to m and b and $Error = Y' - Y$.

To solve Equation 45, two possibilities exist:

- Looking at each error one at a time: Stochastic gradient descent.
- Looking at the error of all training examples at once: Batch gradient descent.

In this example, the first possibility was taken, the Stochastic gradient descent.

Because $J_{m,b}$ depends on two different parameters, the partial derivative are used. Two concepts can be used for calculation:

- Power rule: $f(x) = x^n \rightarrow \frac{\partial f(x)}{\partial n} = n.x^{n-1}.f'(x)$
- Chain rule: $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}.\frac{\partial y}{\partial x}$ here z depends on y which depends on x, so z depends on x.

The partial derivatives of $J_{m,b}$ according to m and b are (Equation 46):

$$\begin{cases} \frac{\partial J}{\partial m} = 2.Error.\frac{\partial}{\partial m}Error \\ \frac{\partial J}{\partial b} = 2.Error.\frac{\partial}{\partial b}Error \end{cases} \qquad \text{Equation 46}$$

In Equation 45, $Error\ as\ Y' - Y$ .If Y' is expressed with the line equation, Equation 45 becomes (Equation 47):

$$Error = mx + b - Y \qquad \text{Equation 47}$$

So the partial derivation relationship can be defined as (Equation 48):

$$\begin{cases} \frac{\partial}{\partial m}Error = x \\ \frac{\partial}{\partial b}Error = 1 \end{cases} \qquad \text{Equation 48}$$

Substituting the values back to the cost function and multiplying it with the step (Equation 49):

$$\begin{cases} \frac{\partial J}{\partial m} = 2.Error.x.step \\ \frac{\partial J}{\partial b} = 2.Error.1.step \end{cases}$$ Equation 49

The step parameter is defined by the user, and gives the size of the steps taken to reach the minimum [110].

2.4.1.3. Levenberg-Marquardt algorithm

The Levenberg-Marquardt (LM) algorithm is based on the Gauss-Newton method and the gradient descent method as previously stated.

The starting point of the LM is the equation of the gradient descent method (Equation 50):

$$w_{i+1} = w_i - \mu.d$$ Equation 50

where
w_{i+1} is the next position, w_i is the actual position, μ, the step and d, the derivate.

By using the Gauss-Newton method, the approximate quadraticity of the objective function is assumed in the parameters near the optimal solution. Equation 50 becomes (Equation 51):

$$w_{i+1} = w_i - H^{-1}.d$$ Equation 51

where
H is the Hessian matrix (a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field).

Levenberg introduced the notion of "blending" between the quadratic and linear approximation. A gradient descent method can be used until it approaches of a minimum, before switching to the quadratic rule. The error is used to determine the distance to a minimum. If the changing of the error is important, the minimum is far away, if the changing of the error is small, the minimum is close.

Levenberg's algorithm is formalised by introducing λ, a blending factor, which will determine when there is a need to switch between the gradient descent and the Gauss-Newton method. Thus, Equation 51 is rewritten as Equation 52:

$$w_{i+1} = w_i - (H + \lambda I)^{-1}.d$$ Equation 52

where I is the identity matrix.

Finally, Marquardt improved the method with an estimation of the local curvature information. The insight of Marquardt was to improve the determination of the step parameter. When λ is high, the

gradient descent method is used, allowing the use of the Hessian matrix. The diagonal of the Hessian matrix can used to move further in the directions which the gradient is smaller.

Thanks to that improvement, Equation 51 becomes the Levenberg-Marquardt algorithm (Equation 53):

$$w_{i+1} = w_i - (H + \lambda . diag[H])^{-1} . d \qquad \text{Equation 53}$$

2.4.1.4. Sparse method

Conditions for using the Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm is built on the Gauss-Newton, and the gradient descent methods. However, to use these algorithms, the LM function needs to satisfy some requirements:

- To be minimized:
  The function converges to a minimum.
- To converge:
  Have a minimum (local or global).
  Define positive, i.e.,$f(x) > 0$.
- To be convex [111]:
  Have a second derivate positive.
  If a line is drawn from one extremum of the function to another (A and B on the drawing below), the line stays in the same space.

Figure 2.17: Illustration of a convex and no convex space

- To be linear:
  Have a quadratic form.

The field of the function also needs to be defined correctly, to avoid potential divergence problems caused by a local extremum or a horizontal asymptote. The equation governing the Levenberg-Marquardt algorithm was defined previously. It was noted that the LM algorithm uses the Hessian matrix, and its diagonal. The Hessian matrix is the second derivate of any function.

When non-linear functions are used, the Hessian matrix is composed of partial second derivatives; and its size corresponds to the number of parameters in the problem wanted to solve. The greater the non-linearity, the greater the need for computing power, as well as the computing memory capacity to store all the parameters calculated during the minimum search process. To reduce the problem, the following methods can be applied:

- Physically:
  For example, reduce the size of the problem by minimizing the number of parameters used, or the size of the matrices used.
- Mathematically:
  For example, using sparse methods [108].

However, sometimes it is impossible to reduce physically the size of the system generated. Mathematical methods such as the sparse method have to be used.

Sparse method and Least squares algorithm

The sparse method is used to reduce the size of the linear system to a realistic size. It is based on the sparsity of the constraint matrix used in the problem. A matrix is defined as sparse when most of its elements are zero. There is no strict definition of the number of elements that must be zero in a matrix to be considered sparse, but a common criterion is that the number of non-zero elements is approximately the number of rows or columns.

This method is widely used in applications where computational costs need to be reduced, such as in image processing, signal processing, machine learning or medical imaging. Furthermore, the sparse method can be combined with different minimization methods such as the LU (lower–upper) factorisation or the least squares method to solve non-linear problems [112].

It is noted here that the LU factorisation results in the decomposition of matrix as a product of a lower triangular matrix, and an upper triangular matrix [112].

Least squares method

The linear least squares method is the simplest and the most commonly applied form of linear regression. It solves the problem of finding the best-fit straight line through a set of points [113-114].

For any function, the best-fitting curve for a given set of points will be found by minimising the sum of the squared offsets (also called residuals or errors) of the points from the curve. In this method, the sum of the squared offsets is used, instead of the absolute values of the offsets, to allow the residuals to be treated as a continuous differentiable quantity. However, this has the consequence that outliers can have a disproportionate effect on the fit.

The linear least squares, for any functions with n set points, is defined as (Equation 54):

$$R^2 = \sum [y_i - f(x_i, a_1, a_2, \dots, a_n)]^2 \qquad \text{Equation 54}$$

where $R^2$is the sum of the squares of the vertical deviations, n is the set of data points and f, the line equation $mx + b$.

R^2 is minimised if [113-114] (Equation 55):

$$\frac{\partial R^2}{\partial a_i} = 0 \qquad \text{Equation 55}$$

The non-linear least squares method works in a similar way to the linear method. For any function, this method is applied iteratively to a linearised form of the function until convergence is achieved. If uncertainties have been assigned to the points defining the function, they can be weighted differently to give more weight to the high-quality points [113-114].

A function f can be defined as (Equation 56):

- For function f of variable x, tabulated at m values, we have (Equation 56):

$$y_1 = f(x_1) \quad \text{Equation 56}$$

- The function depends on n parameters: $f(x_1; \lambda_1, \lambda_2, \dots, \lambda_n)$

Whilst the overdetermined set of m equations can be defined as (Equation 57):

$$\begin{cases} y_1 = f(x_1; \lambda_1, \lambda_2, \dots, \lambda_n) \\ y_m = f(x_m; \lambda_1, \lambda_2, \dots, \lambda_n) \end{cases} \quad \text{Equation 57}$$

To solve this problem, the optimum values of $\lambda_1, \dots, \lambda_n$ satisfying this system of equation is searched.

Starting by picking an initial guess for the λ_i (Equation 58):

$$d\beta_i = y_i - f(x_i; \lambda_1, \dots, \lambda_n) \quad \text{Equation 58}$$

To apply the minimisation condition expressed in Equation 55, f has to be linearised. In addition, the linearised estimation of changes in $d\lambda_i$ is needed to reduce $d\beta_i$ to zero (Equation 59):

$$d\beta_i = \sum_{j=1}^{n} \frac{\partial f}{\partial \lambda_j} . d\lambda_j \Big|_{x_i, \lambda} \quad \text{for } i = 1, \dots, m \quad \text{Equation 59}$$

Equation 59 can be rewritten as (Equation 60):

$$d\beta_i = A_{ij} d\lambda_j \quad \text{Equation 60}$$

where A_{ij} is the Jacobian of f.

By applying A^T to both sides, Equation 60 becomes (Equation 61):

$$A^T d\beta = (A^T . A) d\lambda \quad \text{Equation 61}$$

Equation 61 can be solved for $d\lambda$ by using standard matrix techniques such as Gaussian elimination. If the set of equations is overdetermined, analogous techniques can be used to solve such as the Euler angles. However, it should be noted that the convergence properties generally deteriorate as the number of free parameters increases [113-114].

The bundle adjustment is the traditional technique used to optimise and refine the results of a calibration process, based on the Levenberg-Marquart algorithm. This algorithm is based on an intelligent use of two algorithms to find the minimum of a given problem (Newton and gradient descent). In addition, when the system generated by the given problem starts to be too large, the sparse method can be used to reduce and simplify it, thus reducing the cost of computing power and the running time of the algorithm on a machine.

However, it is sometimes necessary to define and characterise the performance of the calibrated system and the resulting calibration error. The Levenberg-Marquart algorithm does not address this problem, so other solutions must be considered, such as the reprojection error.

2.4.2.Reprojection error

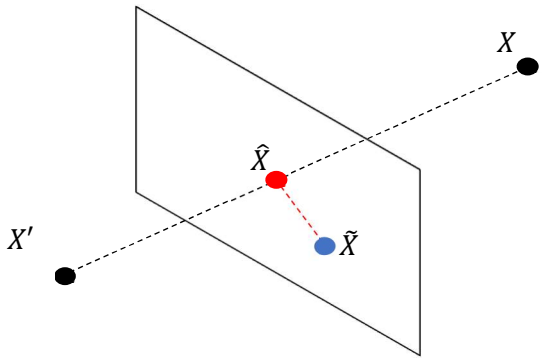


Figure 2.17: Reprojection error

The reprojection error [115] is a geometric error corresponding to the image distance between a projected point X and a measured point $\hat{X}$ (Figure 2.17). In this instance, based on the standard deviation of set of measurements, the reprojection error is used to quantify the extent to which the estimation of a 3D point, $\tilde{X}$, recreates the real projection of the point $\hat{X}$. In the case of detection, this error makes it possible to quantify the quality of circle (centre) detection by a set of cameras and to observe the variations in detection as a function of the positions of the cameras in the scene. It is noted that quantification of reprojection errors is inconsistently reported in literature and can also be reported as a function of the; root-mean-square of all the reprojection norms, mean of the reprojection errors, and, median of the reprojection errors.

With respect to Equation 62, the standard deviation represents the variation or dispersion of a set of samples around the mean on the basis that the data is parametric and follows a Gaussian or Normal distribution. In this instance it is used as a measure for the reprojection error but does not allow specific identity of error components or individual contributions.

$$\text{Error} = 1.96 \times \frac{\|\sigma\|}{\sqrt{N_e}} \qquad \text{Equation 62}$$

Where; σ is the standard deviation of the error mean, Ne is the number of errors, and 1.96 the multiplication factor to obtain a result at a coverage factor of k ≈ 2 (95% confidence limits based upon a Normal distribution).

The previous sections outlined the state of the art of optical systems, as well as the theoretical links with camera functionality, photogrammetry, and calibration. These sections have shown that camera-based solutions meet the requirements of Industry 4.0 (such as flexibility, agility, ease of use and versatility). However, the problem of optimising current camera-based solutions was also highlighted. Due to the number of possible camera-based solutions and the number of calibration techniques, it is difficult to choose and find the right technology for the right applications, as well as the right calibration technique for the right technology. Furthermore, due to the need to remain competitive and the need for smart manufacturing to link physical and virtual spaces [116], digital twins have started to be used to solve this problem [117-118].

2.5. Digital twin

2.5.1. Digital twin definition and application in machine vision

Until the 20th century in manufacturing industry, physical space played the main role in the control of production, potentially limiting efficiency, precision and transparency in the work process. The development of virtual technologies such as computers, simulation tools, the Internet and wireless networks, has introduced the notion of virtual space in the factory, and new possibilities, for example, to virtualise the physical assets of the production line [9].

The parallel coexistence of these two spaces has provided opportunities to conduct plans and operations more effectively and efficiently, leading to improved operating situations and technologies. However, this coexistence has triggered greater interactions between the virtual and the physical worlds than ever before, which raises the question of connecting physical and virtual spaces [10].

The notion of the digital twin was developed to meet this need. Allowing a better flexibility and scalability of manufacturing systems, the aim is to create a high-fidelity virtual model for each physical entity to emulate their states and behaviours with abilities of evaluating, optimizing, and predicting [10].

This new type of industry, based on physical and virtual spaces, with the possibility to fuse them together, has had many different names, depending on the country of origin. For instance, in Germany, it has been called “Industry 4.0”; in the United States “Advanced Manufacturing” or “Smart Manufacturing”; in Japan, “Society 5.0”; in China, “Made in China 2025”; or in France, “Industry of the Future”; but generally, in Europe, “The Factory of the Future” [10].

The concept of digital twins was first introduced by Grieves and Vickers, in 2003, during a lecture about the product life-cycle management, illustrated in Figure 2.18. Grieves was then describing the virtual product representations as “*…relatively new and immature*” and the data collected about physical products as “*…limited, manually collected, and mostly paper-based*” [10]. However, the concept of a “twin” representation, not digital, was introduced by the NASA, many decades earlier during the Apollo program. Two identical space vehicles were built to simulate the conditions of the space vehicle during the mission [10].

Various definitions of digital twin exist. For instance, the Defence Acquisition University defines a digital twin as “*an integrated multi physics, multiscale, probabilistic simulation of an as-built system, enabled by Digital Thread, that uses the best available models, sensor information, and input data to mirror and predict activities/performance over the life of its corresponding physical twin*.” [119].

An alternative has been proposed by, Dreisbach, NAFEMS (National Agency for Finite Element Methods and Standards) council member as: “*a physics-based dynamic computer representation of a physical object that exploits distributed information management and virtual-to-augmented reality technologies to monitor the object, and to share and update discrete data dynamically between the virtual and real products.”* [119].

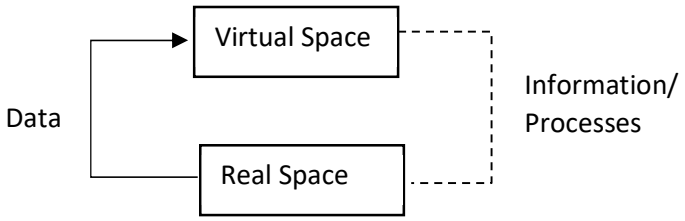


Figure 2.18: Definition of the digital twin by Grieves [10]

According to Grieves, the digital twin is composed of three components, which describes a cycle between the physical and virtual states:

- A physical product.
- A virtual representation of that product.
- The two loops of data connections. One from the Real Space to the Virtual Space, for the data transmission; and another one from the Virtual Space to the Real Space, for information/processes communication.

With the development of artificial intelligence [120], a digital twin starts to have potential for real-time remote monitoring and control, allowing for system updating in a real-time [119]. However, this new function does not define a digital twin, it is just an update, a new branch, providing accurate description of objects changing over time. According to the initial definition, the virtual model needs not be a data-driven model, but should produce results directly equivalent to a measured quantity in the real model [119].

Digital twins are used in various applications for design, optimisation, process control, virtual testing, predictive maintenance, and lifetime estimation [119]. Nonetheless, it can be costly, require the fitting of sensors to provide better data, with their integration in the model often being complex and computer power consuming [121]. Furthermore, being described as a virtual model of a physical thing or system, the distinction between virtual modelling, simulation and predictive systems becomes blurred.

A 3D virtual model of an object is the process of developing a mathematical coordinate-based representation of any surface of an object (inanimate or living) in three dimensions via specialized software by manipulating edges, vertices, and polygons in a simulated 3D space [121 - 123]. As its name indicates it, a digital twin of an object is its virtual twin, i.e., it is a sufficiently accurate (such as texture, size, colour, imperfections) representation of the physical object to be its twin [119].

Both, digital twins and simulations use digital models reproducing real objects and scene. However digital twins use a virtual environment to run simulations to predict the reactions and the behaviours of the real system, before sending the results back to the real system. Moreover, a two-way flow of information should exist between the real system and its twin [120].

The key differences between digital twins and simulations are shown in Table 2.5 [120]:

Table 2.5: Key differences between digital twins and simulations

Digital twin	Simulation
Active (two loops of communication)	Static
Actual (replicate what is happening)	Possible (test against parameters to determine possible behaviours)
All stage of product's lifecycle	Test different scenarios for product design

Predictive systems are tools capable of discovering and analysing patterns in data so that past behaviour can be used to forecast likely future behaviour. However, as explained previously, one of the key characteristics of a digital twin is its two loops of communication between the virtual and real state. In predictive systems, this key element does not exist.

In machine vision, digital twins are starting to be used to solve optimisation problems [117-118,], allowing rapid investigation of installation and positioning issues but with significantly reduced collateral and financial overheads at the investigation stage. In addition, they allow for rapid changing of camera specifications as a further system optimisation.

Examples exist such as in the work of Nikolakis et al [19], where a digital twin is employed in a cyber-physical system to enable optimisation of the planning and commissioning of human-based production processes. Using simulation-based approaches, sensor data are merged with motion recognition of human activities. In the work presented by Wang et al [124], where the concept of digital twin is used with virtual reality (VR) to generate human-robot interaction such as a person remotely demonstrating a task plan to a robot. In the work of Zhu et al [125], where digital twin is combined with augmented reality (AR) to solve manufacturing problems. Here, the AR uses Microsoft HoloLens to visualise the data generated by the digital twin of a CNC milling machine, for monitoring and controlling the machine tool in real-time, as well as updating the digital model simultaneously. In work presented by Stączek et al [126], digital twins are used to test the operating environment of an autonomous mobile robot (AMR). These systems are based on the sensor fusion, methodology, integrating camera image, and cloud digital twin system.

Nonetheless, the concept of digital twins is still quite recent and questions concerning reliability, the quality of the digital model of a scene/object and the analysis of the simulation results are addressed, and answered by characterising the digital twin [9, 127] through a metrology process. In this study the development of a digital twin environment was considered, whereby motion tracking camera positions were flexibly and rapidly modelled to determine optimum locations – thus significantly reducing practical set-up times in the work place.

However, digital twins are often merged with other technologies such as virtual reality or augmented reality, avoiding the digital design of the camera as part of the digital model [120]. When, the camera is part of the virtual model, as it is presented in the work of Steil et al [117] and Farrell et al [118], the whole virtual camera from the camera sensor to the radiometric characteristics of a real camera is designed. This is a complex process, that is not necessarily accurate and is time-power consuming. Other alternatives exist such as the virtual cameras used for rendering in 3D animation software.

2.5.2. 3D modelling space

Digital twins have the potential to reproduce a real model in a digital space, with significant accuracy. This characteristic is also one of the key elements of 3D animation software. 3D animation software belongs to the family of computer graphic software. It enables the design, development and production of 3D graphics and animations. It helps the users to, for instance, visualise, design and control objects, environments in a 3D world, and has applications in architecture, game designer, film production, and the medical industry [127]. The most well-known 3D animation software are Autodesk Maya, Autodesk 3DS Max, Blender, Cinema 4D, and Houdini [128 - 130].

Autodesk Maya

Autodesk Maya, also known as Maya, was developed in 1998 by Alias, and was bought by Autodesk in 2005, which currently owns and develops it. It is written in C++, MEL (Maya Embedded Language), Python and C#, runs on Windows, macOS and Linux operating system. Maya creates assets (used to create templates to plan and organize the capabilities and attributes of various parts of the scene) for iterative 3D applications, animated films, TV series, and visual effects [131]. This software was used in collaboration with Walt Disney Feature to develop their movie “Dinosaur”.

Autodesk 3DS Max

3DS Max was developed in 1996 by Yosh Group, and published by Autodesk. It is written in C, C++, and Python. 3DS Max has modelling capabilities and a flexible plugin architecture and is used on the Microsoft Windows platform. It is frequently used by video game developers, many TV commercial studios, and architectural visualization studios [132]. However, it has less animation-oriented functions than Maya, it is the software that is generally associated with architectural visualisation and real-time 3D, due to its powerful polygonal modelling tools in its "Graphite" module and the large library of proprietary 3D content [133].

Blender

Blender was developed in 1994 by the community Blender Foundation. It is a free and open-source 3D computer software used for creating animated films, visual effects, art, 3D-printed models, motion graphics, interactive 3D applications, virtual reality, and, formerly, video games. It is written in C, C++ and Python, and runs on Linux, macOS, Windows, FreeBSD, OpenBSD, NetBSD, DragonFly BSD, and Haiku operating systems [135]. When introduced, Blender was considered as a revolution in the 3D animation world, but was not powerful or developed enough to impose itself on the market [134]. More recently, due to the evolution of its render engine Cycles, allowing new rendering tools, and based on a better ray tracing algorithm, has become one of the most important software on the market [–135-137].

Cinema 4D

Cinema 4D was developed in 1990 by MAXON computers. It is written in C++, and Python, and runs on macOS and Windows operating system. It is a tool for modelling, texturing, animating and

rendering 3D objects. Similar to Maya, it is a commercial product with a free version ("LITE"), offering a limited set of tools with the basic option for the 3D. Different commercial version exist such as "PRIME", and "STUDIO", at different prices, proposing different advance set of tools depending on the needs [138].

Houdini

Houdini was developed in 1996 by Toronto-based SideFX. It is used to produce different effects such as complex reflections, animations and particle systems. Major visual effects (VFX) companies such as Walt Disney Animation Studios, Pixar, DreamWorks Animation, Double Negative, ILM, MPC, Framestore, Sony Pictures Imageworks, Scanline VFX, Method Studios and The Mill; have collaborated with it on the development of films and video games [139]. For instance, from the collaboration with Disney, the movies Frozen, Zootopia, and Raya and the Last Dragon have been created [139]. Houdini is a reference in realistic simulation of fluids and particles [139]. It is a commercial product with a free version for students and academia [140]. Different version of this software exists such as Houdini FX or CORE, meeting different needs, and offering different packaging as Cinema 4D.

In summary, from a specification point of view, Maya is specialised in 3D animation and special effects. 3DS Max is the reference in polygonal modelling. Blender is an open-source software, and the new challenger, specialised in anything. Cinema 4D is well known for its extremely fast and efficient dynamic simulations. Houdini is the software for realistic simulations of fluids and particles.

From a markert point of view, Maya, 3DS Max, Cinema 4D, and Houdini are commercial products, with a free option offering limited tools for Cinema 4D. In comparison Blender is free and open-source product. However, it is not the only free and open-source system. Other software such as Synfig, OpenToonz, TubiTube Desk, and Pencil2D exist, but they are either have less tools than Blender, or 2D animation software.

The problem with most 3D animation software on the market today is its specialisation, which forces users to switch from one platform to another depending on their working context and needs [140].

The following Table compares the various 3D animation software packages on the basis of criteria taken from the previous software presentation.

Table 2.6: Comparison of 3D animation software

	Free	**Specialise**	**Python script**	**Ray tracing**	**Modelling**
Autodesk Maya		x	x	x	x
Autodesk 3DS Max		x	x	x	x
Blender	x		x	x	x
Cinema 4D		x	x	x	X
Houdini		x	x	x	x

According to the previous Table, Blender is the best option to design a metrology digital twin of optical camera systems because it is free and combines the functionality to model the interactions of scenes with light sources. It also generates simulated images based on modelled cameras using its internal rendering engine based on ray-tracing and Python scripting options. In addition, recent digital applications using Blender relevant to metrology have been developed, such as BALINDER [141], an add-on for Blender allowing different depth sensors to be loaded from pre-sets; customized sensors can be implemented and different environmental conditions (e.g., influence of rain, dust) can be simulated, and BlenderProc [142], a procedural pipeline helping in generating real looking images for the training of convolutional neural networks. The applications of this work are numerous, such as segmentation, depth, and normal and pose estimation.

Blender seems to be a valuable tool for digital twin creation. However, a digital twin is "*a virtual representation of a physical product containing information about said product, with its origins in the field of product life-cycle management*", and its virtual model reproduces with high accuracy the shape, the texture, and the light interaction of the real object of interest.

According to this definition, a basic model designed in any 3D animation software cannot be a digital twin because of the potential for a lack of realism. 3D animation software is used for artistic, cinematographic, entertainment and architectural purposes. In all these fields, a level of realism must be achieved, in order to deceive the user's perception or to show a final rendering for decision making.

Moreover, rendering is completed through a camera process, where a digital camera takes a picture of the scene created. In art, a movement called photorealism, was created to generate a piece of art almost identical to picture taken per a camera. This art movement is the solution used by all the above-mentioned domains to generate rendering images that look like real images.

2.6. Conclusion

Industry 4.0 has generated new needs in terms of safety, object and people tracking, and optimisation. Optical systems based on multi-camera networks are being introduced because they are cheap, easy to use, and less intrusive compared to other systems. Examples of this technology are the Vicon system (marker system), the Kinect V2 (marker-less system), or the photogrammetry method where multiple cameras take pictures of an object from different points of view and perform 3D reconstruction from the 2D images.

However, optical systems have drawbacks linked to the placement of markers, the line of sight, the type of camera used, and the tracking volume which can be quite small. In addition, they are not necessarily optimised solutions, where the cameras are placed by hand (through a trial-and-error process), or by using simple mathematical tools.

Due to the need to remain competitive, and the desire of companies to link together the physical and virtual spaces, digital twins, and digital models start increasingly to be investigated. Designing a digital twin of a camera has already been achieved and have concentrated on designing the whole body and radiometrically map of the camera. They are often time consuming, requiring significant computer power, with potential accuracy issues.

An alternative solution is 3D animation software, using a virtual camera to perform image rendering. The biggest names on the market are 3DS Max, 3DS Maya, Blender, Houdini, and Unity. Blender was chosen because it is a free-open-source software, multidisciplinary in its applications.

However, designing a digital twin in a 3D animation software is not an easy task. Photorealism principles have to be used to add realism, as well as calibration process and image fidelity, to calibrate the cameras and compare the results from both models (real and virtual).

This Chapter has also explored the definition of the camera in the real world, as well as defining the photogrammetry principal, and the physic behind. The functioning of real camera is usually described by the pinhole camera model. This model describes the fundamental working principle of the thin lens camera (such as no panoramic, fish eyes), and to calculate the camera parameters, e.g., the intrinsic matrix (intrinsic parameters), the rotation matrix and the translation vector (extrinsic parameters). The light coming from the scene passes through the aperture, modelled by a hole, and projects an inverted image onto the image plane. This image is inverted due to the light propagation.

While photogrammetry is based on triangulation and epipolar geometry. The accuracy and precision of the output of the triangulation is generally refined with the Bundle adjustment based on the Levenberg-Marquardt algorithm, by recalculating the camera parameters, and the 3D reprojection of the real points in the image plane.

The equations of primary importance to this research are coded in the following experimental chapters (4, 5, 6 and 7), and are listed in Table 2.7, which is an extension of Table 2.4:

Table 2.7: Extension of Table 2.4 - Main equations used in the MATALB algorithms

Intrinsic parameters (Equation 3)	$K = \begin{pmatrix} f_x & s & O_x \\ 0 & f_y & O_y \\ 0 & 0 & 1 \end{pmatrix}$
Triangulation problem (Equation 8)	Find X that minimizes: $d^2(x_1, P_{1X}) + d^2(x_2, P_{2X})$
Collinearity model (Equation 12)	$\begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix} = \lambda . R^T \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix}$ Where λ is a scale factor, and R, the rotation matrix of the camera
Coplanarity condition (Equation 14 and Equation 16)	$b.(a_1 \times a_2) = 0$ Where: $\begin{cases} a_1 = \begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = R_1^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_1 \\ a_2 = \begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = R_2^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix}_2 \end{cases}$
Least square (Equation 27)	$Ax = b_0 - f(x_0) + v$
Collinearity equations after linearisation (Equation 36)	$\begin{cases} F_x = -f.\frac{U}{W} \\ F_y = -f.\frac{V}{W} \end{cases}$ Where $\begin{bmatrix} U \\ V \\ W \end{bmatrix} = R. \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}$
Levenberg-Marquart algorithm (Equation 53)	$w_{i+1} = w_i - (H + \lambda . diag[H])^{-1}.d$
Reprojection error (Equation 62)	$\text{Error} = 1.96 \times \frac{\|\sigma\|}{\sqrt{N_e}}$

This literature review has highlighted the following research gaps:

- Current existing digital twin models of camera systems are too complex and time consuming.
- 3D animations software has not been investigated as potential metrology tool to design digital twin.
- The traceability and reliability of 3D models has been addressed but current methods are not satisfactory.

These gaps will be addressed in the experimental chapters (4, 5, 6, and 7), while Chapter 3 will present the core methodology of the thesis.

Chapter 3

Thesis methodology

As explained in Chapter 1, the aim of this thesis is the development of a digital twin/model of multi-camera network systems in Blender. The output is to be a digital model of a complex real-world scene, to be judged similar enough to the real one, by using various data analysis tools to characterise it. However, due to the nature of the research, this output was generated through a constantly challenging process.

The use of Blender, for this purpose, is new and unexplored, as explained in Chapter 2. It was therefore necessary to carry out experiments to determine its suitability for this task before beginning to use it. This first step proved that Blender can be used as a potential metrology tool. However, due to the lack of realism in the virtual scene, as well as incomplete definitions of the modelling boundaries, the quality of the model was found to be lacking.

The second step, presented in Chapter 5, is to accurately simulate the real environment in Blender. Photorealism is used as a tool to proactively degrade the Blender “perfect” environment to a virtual model that is "sufficiently similar" to the real environment.

This new virtual environment raised questions about calibration and traceability, as well as limitations. Various tools, artefacts, and experiments were then designed and considered to answer these questions and produce the final output of this thesis.

However, the experimental methodology of this thesis is built around the utilisation/reutilisation and modification (depending on the Chapter context) of two core experimental scenarios concerned with sphere diameter measurement (single camera) and multiple camera position optimisation. Chapter 3 has been developed to introduce these primary method statements for the whole thesis, while Chapters 4, 5, 6 and 7 present the method exploration, as well as the results and discussions of each experiment.

In this Chapter, Section 3.1 presents the artefacts used primarily in the research, Section 3.2 introduces the MATLAB functions used to perform image analysis and extract the output needed by the experiment, and Section 3.3 discusses the method statements to be used in Chapters 4, 5, 6, and 7.

3.1. Artefacts

3.1.1. Simple objects

Simple objects were used to perform object measurement using a single camera to assess camera performance in Blender. In addition, these object measurements were also used to design the final virtual model and to test its robustness.

These simple objects were:

- Diffuse white expanded polystyrene spheres of 0.09 m, 0.15 m, 0.20 m, 0.25 m, and 0.30 m diameter.
- Diffuse white paper circles of 0.07 m and 0.09 m diameter, printed on a black background.
- Diffuse white reinforced plastic cubes with sides 0.1 m, and 0.2 m long.

All spheres have been remeasured using the LK Ultra CMM, using a multi-point measurement routine with multiple repeats.

CMMs in touch trigger mode are verified for performance with respect to ISO 10360-2 on an annual basis. This identifies that the CMM is performing to manufacturers specification under a limited set of measurement conditions. However, it does not provide an explicit statement of measurement uncertainty – it provides a statement of machine conformance.

Currently the CMM is assessed against a Maximum Permissible Error of [EMPE = 1.75 + L/127] micrometres, where L is the length of the measured artefact. Consequently, for a 90 mm sphere, this would create an upper performance boundary of 2.46 micrometres. For the 300 mm sphere – this value would be 4.11 micrometres.

In reality, the performance of the CMM is not an issue here. The main source of sphere geometry error is caused by the fact that the spheres are manufactured from blown polystyrene in a two-part mould – that approximates to a sphere but is not manufactured to metrology reference sphere standards. Hence, the most appropriate method of demonstrating variance in the data and hence in the geometry of the spheres is to present the Standard Deviation of the multiple sphere measurement results. It is noted that SD is not a direct indication of measurement uncertainty, but the geometry variation will be very significantly larger than any CMM uncertainty.

Spheres were chosen because they are widely used in metrology applications for camera calibration [143-144], in addition the use of a sphere allows for simulated light interactions with 3D objects. A sphere is also convenient because geometrically it is always observed as a 2D circle no matter which orientation it is viewed from. This characteristic compensates for potential imperfections of the real and virtual experimental setup, for instance physical camera misalignment, allowing experiments to be reproduced without adding new sources of error such as detection problems coming from such misalignment.

Whilst a sphere is straight forward in concept the simple geometry still introduces consequences for the measurement of the sphere diameter using a 2D camera Figure 3.1 shows a sphere albeit in this case illustrated as a circle. Point C represents the camera's optical centre whilst points A and B are the intersection points between the extremities of the FOV and the circle. These extremities are tangential to the circle.

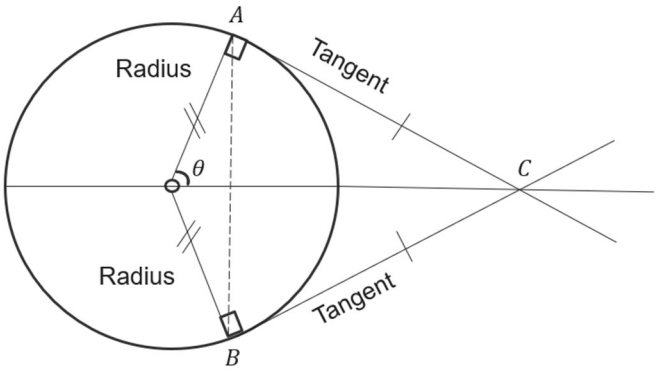


Figure 3.1: Relationship between a sphere and camera FOV

The segment [AB] is not equal to the real sphere's true diameter, but represents the diameter perceived by the camera because these points are the interception points between the camera's FOV, and the sphere. The angle ɵ will increase with increasing object distance between the sphere and the camera. Consequently, as the object distance tends to infinity then ɵ approaches 90˚, and the segment [AB] as recorded by the camera would then approximate to the true diameter. However, it is also noted that as the object distance tends to infinity, then the spatial resolution of the camera image will degrade as a function of reduced pixel density thus introducing alternative error terms.

2D circles were also considered for measurement to remove the geometric error related to the detection of the sphere cited above, and to keep a homogeneity in the objects used in the experiment. The outcome criterion from this real and virtual experimentation was the comparison between deviation of the measured circle diameter and the theoretical ones.

Cubes were used because geometrically speaking they are observed as 2D squares if the camera axis is normal to the cube, otherwise a rectangle will be observed . This characteristic compensates for potential imperfections of the real and virtual experimental setup, for instance physical camera misalignment, allowing this experiment to be reproduced without adding new sources of error such as detection problems coming from such misalignment.

In addition, a cube does not introduce geometrical errors as seen for a sphere and allows to generate a 3D measurement reference due to perfect detection at any distances, and to validate the measurement process. Figure 3.2 shows a cube albeit in this case illustrated as a square. Point C represents the camera's optical centre whilst points A and B are the intersection points between the extremities of the FOV and the square. These extremities are perpendicular to diagonal of the square.

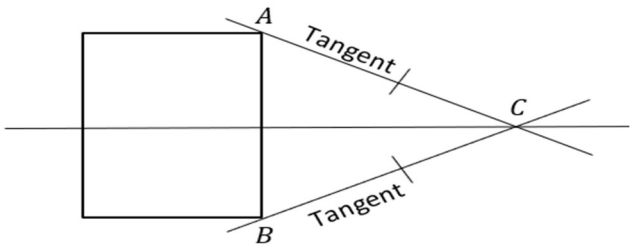


Figure 3.2: Relationship between a cube and camera FOV

The segment [AB] is equal to the real square's true dimensions (length or width). However, it is also noted that as the object distance tends to infinity, then the spatial resolution of the camera image will degrade as a function of reduced pixel density thus again introducing alternative error terms.

3.1.2.Board artefacts

Camera calibration board artefacts were used to evaluate the ability of Blender to predict the camera locations relative to a camera calibration artefact, using photogrammetry techniques. They were also used to assess the robustness of the final output of this work.

These two artefacts used were:

- Zone circles board.
- Checkerboard.

The zone circles board was chosen because circular targets have been widely used in image processing and computer vision for their invulnerability to partial occlusion and their ease of access [145 - 147]. However, the accuracy of circle centre detection is sensitive to distortion, centre deviation and other stochastic factors in the imaging process [145]. To minimise the impact of these sensitivity factors, the pattern used was comprised of five zone circles. For better detection and localisation of the zone circles in the images, an alternating black and white zone circles was used.

There is significant debate within the computer image community about the performance characteristics of zone circle-based calibration artefacts versus checkerboard-based artefacts with performance dependent on the nature of the pixel processing algorithms used. In this instance, reliable results were obtained with the zone-circle pattern because all pixels on the periphery of the circles can be used to accurately determine the circular shape of the pattern. In addition, the zone

circles enabled three estimates of the target centre to be made by considering the central black circle, the white circle that surrounds It and finally the outer back circle.

In Figure 3.3**Error! Reference source not found.** , the zone circles cover almost the entire FOV, which facilitates detection from different viewpoints. Circle 3 positioned at 0 mm from the x-axis was used as a reference to check the measurements, and the placement of the pattern on the virtual board. The origin of the frame is placed in the centre of the artefact, along with the zone circle order.

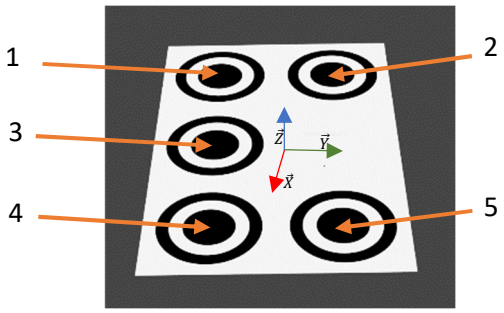


Figure 3.3: Zone circles calibration board

The zone circles were 0.1 m (black), 0.15 m (white) and 0.2 m (black) respectively in diameter on a white diffuse substrate measuring 0.5 m x 0.75 m x 0.025 m. However, due to the overall size of the zone circle artefact, it was challenging to cause the entirety of the object to be within the FOV of each camera at 0.2 m, 0.4 m, 0.6 m and 0.8 m object distance – the artefact being partially cropped within the images at these object distances.

A smaller additional artefact was developed whereby the zone circles were 0.029 m (black), 0.048 m (white) and 0.069 m (black) respectively in diameter on a substrate measuring 0.21 m x 0.29 m x 0.005 m. Both artefacts were likewise simulated in Blender by importing the original digital artwork. The (x,y) coordinates of each zone circle for both the small and large artefacts *are listed in the* **Error! Reference source not found.** with reference to the origin sited at the middle of each board.

Table 3.1: Centre coordinates of the circles for the two zone circle artefacts

Circle number	Coordinates large artefact (x,y) (mm)	Coordinates small artefact (x,y) (mm)
1	-0.125, -0.250	-0.0523, -0.0871
2	0.125, -0.250	0.0523, -0.0871
3	-0.125, 0	-0.0523, 0
4	-0.125, 0.250	-0.0523, 0.0871
5	0.125, 0.250	0.0523, 0.0871

The coordinates of each individual zone circle centre can be extracted from each set of camera images, before using triangulation to create a 3D reconstruction of their positions in relation to the camera positions. The reprojection error for each zone circle centre can then be determined by calculating the deviation between the reference circle centre coordinates and the coordinates determined within MATLAB, with the final answers being expressed in terms of the standard deviation.

The checkerboard was also considered as equipment to perform multi-camera calibration, and extract the values of the focal length and the lens distortion of the camera measured with the MATLAB *cameraCalibrator* app [148]. However, it was also used in the purpose of determining which

calibration board (zone circles or checkerboard) was the more accurate for multi-camera measurements.

For the single camera calibration, the board pattern used was a black and white checkerboard with 30 mm squares as shown in Figure 3.4. For the multi-camera measurements due to the size of the FOV of the camera a smaller identical checkerboard was used with 20 mm squares.

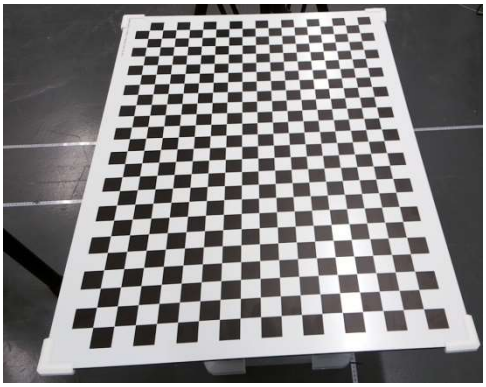


Figure 3.4: Artefact used of the camera calibration

3.1.3.Complex objects

Complex objects were used to determine the weaknesses and strengths of the virtual model generated through the photorealism process (Chapter 7). These artefacts were:

- A calibration ball bar.
- Five duck eggs.

The calibration ball bar was chosen because it is a more complex object but still related to the experiments about the sphere. In addition, the length between the geometrical centre of the two spheres is known through a calibration process and are a sensible transition between a simple and more complex measurement.

Figure 3.5: Ball bar (Top bar: 500 mm, middle bar: 300 mm, bottom bar: 150 mm)

The ArxGeometres GbR artefact, shown in Figure 3.5, consisted of two balls of 38 mm diameter and three bars of 150 mm, 300 mm and 500 mm length. The balls were made of ceramic and were attached

to each end of a bar with magnets, allowing for straight forward change of the bar (and the length of the artefact) without damaging the spheres. The bar was made of polymer (bar ends), and carbon fibre (bar body).

The artefact was calibrated in June 2022 (and subsequently in October 2023), using a LK Ultra 10.7.6 CMM, located in the Wolfson School (Loughborough University). Table 3.2 shows the results of this latest calibration. The sphere diameters and the distances between their geometric centre was measured three times to obtain a repeatable result.

Table 3.2: Calibration results of the ball bar from CMM

Object	Dimension ($\boldsymbol{mm}$)	Mean ($\boldsymbol{mm}$)	Standard deviation ($\boldsymbol{mm}$)
Ball	38.0980	38.0867	0.0128
	38.0933		
	38.0689		
Ball	38.1019	38.0979	0.003
	38.0947		
	38.0972		
Bar 150 mm	150.2975	150.2874	0.0079
	150.2782		
	150.2865		
Bar 300 mm	300.3447	300.2115	0.0944
	300.1365		
	300.1533		
Bar 500 mm	500.2197	500.2184	0.0142

Finally, duck eggs (illustrated in Figure 3.6) were chosen for their size (length between 45 mm and 47 mm), and to simulate a more complex object, but still related to and an extension of the spheres.

Figure 3.6: Duck eggs

A simplified schematic version of an egg is given in Figure 3.7, with the identification of the minor axis, noted $\boldsymbol{a}$ (width) and major axis, noted $\boldsymbol{b}$ (length) of the egg assuming here that the eggs are elliptically symmetric. Noted that this schematic is used in the MATLAB script to plot the shape of the egg detected in the set of images.

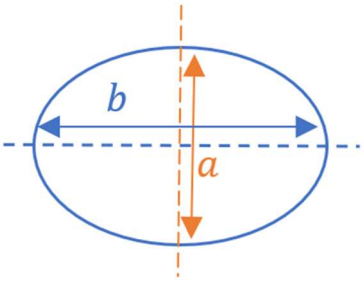


Figure 3.7: Geometrical scheme of an egg

Using Figure 3.7, the theoretical perimeter of the egg was determined with Equation 63[149].

$$p \approx 2\pi \times \sqrt{\frac{a+b}{2}} \qquad \text{Equation 63}$$

The major and minor axis were measured with a vernier calliper. **Error! Reference source not found.**.3 and **Error! Reference source not found.** shows the results of this measurement process.

Table 3.3: Major axis measurements

Egg	Measurement (mm)	Mean (mm)	Standard deviation
1	65.03	67.07	0.030
	65.10		
	65.05		
	65.09		
	65.07		
2	61.35	61.35	0.014
	61.35		
	61.32		
	61.36		
	61.35		
3	63.96	63.92	0.050
	63.98		
	63.86		
	63.88		
	63.90		
4	64.56	64.56	0.029
	64.58		
	64.59		
	64.54		
	64.51		
	57.65		
	57.75		

5	57.50	57.57	0.110
	57.49		
	57.47		

Table 3.4: Minor axis measurements

Egg	Measurement (mm)	Mean (mm)	Standard deviation
1	45.39	45.43	0.030
	45.42		
	45.47		
	45.47		
	45.42		
2	47.42	47.41	0.050
	47.31		
	47.43		
	47.43		
	47.47		
3	48.46	48.41	0.040
	48.37		
	48.39		
	48.45		
	48.39		
4	47.14	48.13	0.016
	47.12		
	47.12		
	47.16		
	47.12		
5	48.49	48.47	0.017
	48.46		
	48.46		
	48.48		
	48.44		

3.2. Raspberry Pi V2

The Raspberry Pi V2 has been chosen for this work due to:

- Straightforward modelling in Blender.
- Fitting industrial applications.
- Tracking objects in a small space where the real experiments were taking place.

As explained in Chapter 3, Blender provides some basic pre-built camera models. These cameras can be categorized into:

- Digital single-lens reflex (APS-C Canon).
- 2D marker less camera (1 inch, 1.8-inch camera).
- Camera of portable phone (Samsung Galaxy).
- Movie cameras (Blackmagic Pocket 4K).
- Digital cinema camera (Arri Alexa 65).
- Full frame and analogue cameras.

The types of cameras generally designed in Blender are there for artistic purposes rather than for industrial purposes. Thus, the camera model used in this project has to be designed using the available Blender options.

In industry, cameras such as Raspberry Pi V2 cameras, USB(2 or 3)-camera , GIGE-camera are used [150 - 151]. Among all of them, the Raspberry Pi V2 was chosen because of its characteristics, as follows:

- A field of field: Allowing tracking in small volume.
- A high depth of focus: Allowing tracking in small volume.
- Fixed lens.
- Well characterised camera: Straightforward modelling in Blender.
- Cheap.
- Used in a large number of applications.
- Easy to use.

The lens of the Raspberry Pi V2 can be categorised as a thin lens because “light from a finite area and produces a well-focused image at a particular distance” [152]. According to [153], the model used to model thin lens camera is the pinhole model.

3.3. MATLAB functions used for the experiments

MATLAB was used to perform processes of the images generated in the real world and in the virtual world (Blender). For the needs of different artefacts and experiments, different series of MATLAB functions were generated.

Their purpose was to:

- Detect a sphere in a cropped image.
- Find the coordinate of each circle's centre on the zone circle board.
- Calculate the Fast Fourier Transform of images.
- Calculate the blur percentage of the image compared to a MATLAB generated image.
- Detect a hemisphere in a cropped image.

3.3.1. Sphere detection in a cropped image

The first series of MATLAB functions were developed to detect the sphere within the FOV and return key data elements. These series of functions were used in Chapters 4, 5, 6, and 7 to detect the spheres, the circles, the cubes, and the spheres of the ball bar. Different functions in MATLAB exist to transform a RGB image into a binary image, and to detect a sphere in an image. This research used the *im2bw(originalImage, 0.4)* function for the binary transformation, and the *regionprops* (in the script *regionprops(logical(binaryImage), originalImage, 'all')*) function for the sphere detection, based on the accuracy of the *regionprops* function in detecting a sphere in an image, and returning key values such as the diameter of the sphere in pixels. The camera image was cropped manually (examples shown later in Figure 3.8), by drawing a rectangle around the sphere to remove undesirable background elements.

The processing steps followed were:

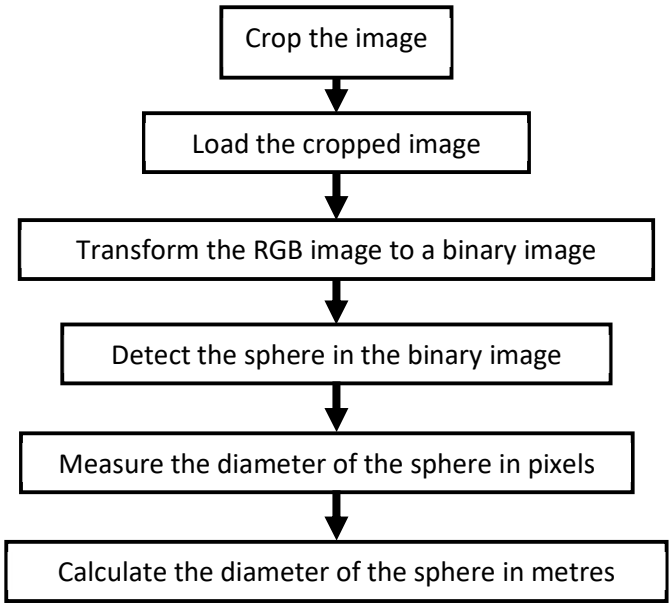


Figure 3.8: Flowchart of the sphere detection script

The calculation of the sphere diameter in metres (last step of the flowchart Figure 3.8) was achieved through a conversion process of the sphere diameter measured with *regionprops* in pixels into a diameter, in metres, by Equation 64:

$$diameter\ (m) = \frac{distance_{camera-sphere}}{focal\ length} \times \frac{\left(sensor_{width} \times diameter\ (pixel)\right)}{Image_{width}} \qquad \text{Equation 64}$$

3.3.2. Determination of the coordinate of the centre of each circle on the zone circle board.

In Chapters 5 and 6, a MATLAB based set of algorithms are described to complete the data processing across experiments and return key data elements. Equation 3, 8, 12, 27, 43 and 62 in Chapter 2 were primarily used to develop these algorithms.

The generic processing steps followed were:

- Improve the contrast of each picture to allow for better detection with the function *imadjust(Image, [0.6, 0.7]*).
- Define the focal length and sensor size of the camera according to the datasheet [154].
- Calculate the intrinsic matrix with *cameraIntrinsics(focal, principalPoint, imageSize).*
- Filter the noise in the image with the function *medfilt2(Image, 'indexed').*
- Transform the RGB image to a grey scale image with the *function rgb2gray(DenoiseImage).*
- Transform the grey image to a binary image with the function *imbinarize(GrayImage, 'adaptive').*
- Calculate the complementary image with the function *imcomplement(BinaryImage).*
- Detect the centre of each circle with the functions *regionprops(Image, 'basic')* on the binary and complimentary images before using *k-means (centers, 5)* to find the centre of the cluster created.
- Check the order of the circle centre.
  Circles 1-3-4: The cross product of the vectors created between circles 1-4, 1-2 is calculated, and its sign checked. If the sign is null, the centre of the board is detected correctly; if the sign is positive circle indices 1 and 4 are correct.
  Circles 2-5: The angles between circles 1-4, and 1-2 is calculated, as well as the angle between 1-5, and 1-4, using simple trigonometric functions. If the circle order is correct, the angle formed between 1-4, and 1-2 is higher than the other one.
- Calculate the camera parameters (rotation matrix and translation vector) with the function *extrinsicsToCameraPose(centers, reference centers, intrinsics parameters).*
- Calculate the 3D coordinates of each target centre by triangulation using the point reprojected in the image frame, and the camera positions using Equation 8 in Chapter 2.
- Calculate the reprojection error within Equation 62 in Chapter 2.

The details of how the MATLAB based set of algorithms work are given in Appendix 2.

The function *extrinsicsToCameraPose* is used to calculate the camera parameters. These parameters were later used to manually set up the virtual camera in Blender, allowing the generation of equivalent real-world environment, and to provide data which could be compared to the results obtained from the modelling scenario.

However, a transformation function was designed to convert the MATLAB camera parameters into the Blender camera parameters. This was required due to an inversion of the Blender coordinate frame relative to the MATLAB frame caused because Blender uses a right-handed coordinate system, while MATLAB uses a left-handed coordinate system [155].

The transformation function is expressed as:

Sens: From MATLAB to Blender

- Rotation
  - $R_{Bx} = R_{Mx}$
  - $R_{By} = R_{My}$
  - $R_{Bz} = R_{Mz}$

- Translation:
  - $T_{Bx} = 180 + T_{Mx}$
  - $T_{By} = -T_{My}$
  - $T_{Bz} = -T_{Mz}$

Where:
(R_{Bx}, R_{By}, R_{Bz}) are the components of the Blender rotation matrix.
(R_{Mx}, R_{My}, R_{Mz}) are the components of the MATLAB rotation matrix.
(T_{Bx}, T_{By}, T_{Bz}) are the components of the Blender translation vector.
(T_{Mx}, T_{My}, T_{Mz}) are the components of the MATLAB translation vector.

3.3.3. Checkerboard detection

In Chapter 5, the checkerboard was used to measure the focal length and the lens distortion of the camera. Pictures of this artefact were taken, and the MATLAB *cameraCalibrator* app [148], for single camera calibration was used to detect the chess pattern and to calculate the cameras parameters (extrinsic and intrinsic). The chess pattern was detected with the *detectCheckerboardPoints* function, and an example of how works the app is shown in Figure 3.9.

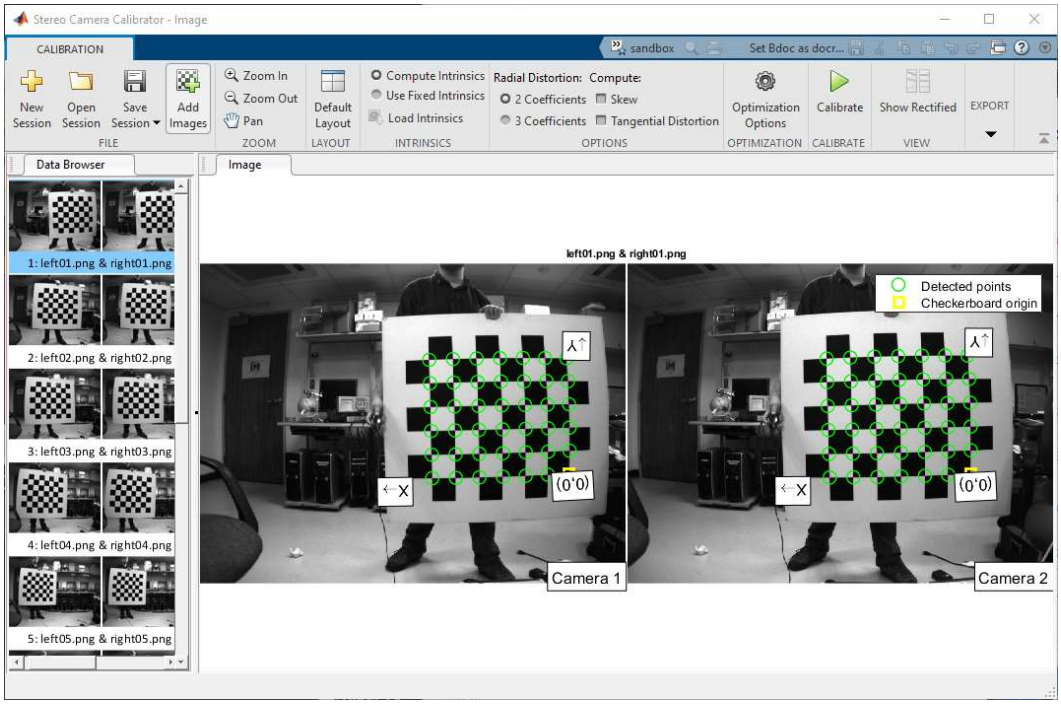


Figure 3.9: Detection of the checkerboard with the *cameraCalibrator* app [148]

3.3.4.Fast Fourier Transform

The following process was used used to plot the Discret Fast Fourier and its logarithm (Figure 3.10) in Chapter 5:

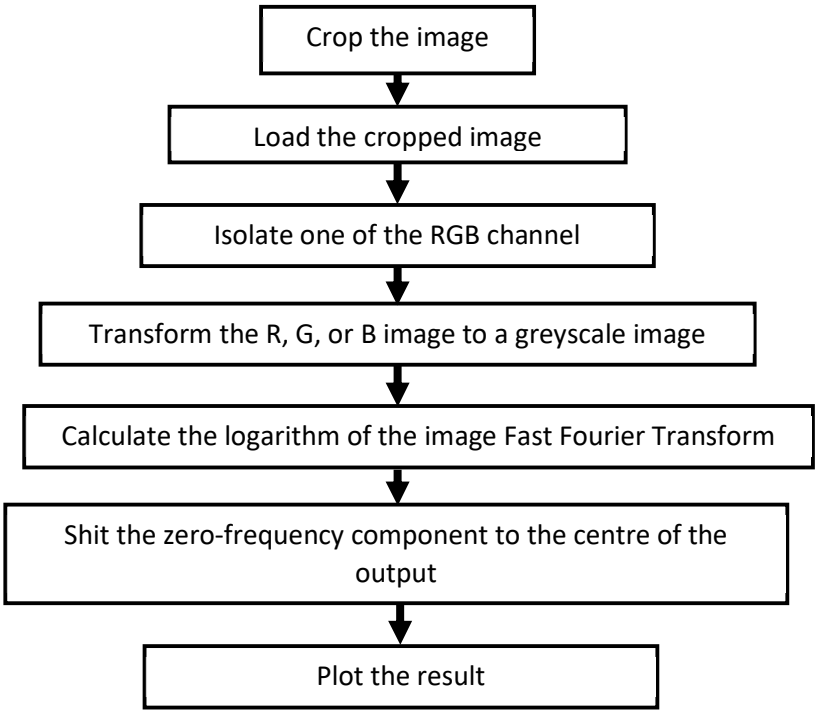


Figure 3.10: Flowchart of the script used to calculate and visualise the Fast Fourier Transform of a 2D image.

The sphere pictures were cropped to isolate the frequency spectrum of the sphere, without any interference from the objects composing the background of the picture. One of the RGB channels was also isolated because each of the three channels is composed of the same frequency spectrum of the image, only the colour map is different. The MATLAB functions used were:

- *rgb2gray(Image)* to convert the RGB image into a greyscale image.
- *fft2(log(1+abs(fft2(GrayImage))))* to calculate the Fast Fourier Transform of the greyscale image.
- *fftshift(log(1+abs(fft2(GrayImage)))*) to shift the zero-frequency component to the center of the output.
- *surf(abs(FFT))* to plot the result.

3.3.5. Blur percentage

In Chapter 5, to quantify the percentage of blur in an image, generally a non-blurred image is used as a reference, which was taken in the same environment, with the same camera, at the same distance from the object as the images under test. In this experiment, this reference image was not available, due to the impossibility of changing the focus distance of the real camera. An alternative solution was found by artificially generating this reference image in MATLAB. This image consisted of a pure white 0.09 m sphere on a pure black background, with the same resolution as the real and virtual images, and generated through the following process:

- The dimensions of the image in x and y were set up.
- With meshgrid the image was created: *meshgrid(1: Resolution in X, 1: Resolution in Y)*.
- The center coordinates of the image were determined by divided the image resolution by 2.
- The radius of the circle was set up.
- The circle was created by using the circle equation:

$$\left(Image\ row - Image\ center_y\right)^2 + \left(Image\ colum - Image\ center_x\right)^2 \leq Radius^2$$

This resulted in a perfect reference image, digitally idealised, taken by an ideal pinhole camera, with an infinite focus of distance [156 - 158]. The blur percentage between this reference image and both real and virtual images was calculated using a MATLAB function, *blurperc*, a community MATLAB function [159].

3.3.6. Signal-to-noise ratio

In Chapter 5, the noise level was calculated with the signal-to-noise ratio, using the MATLAB, according to the following process:

- Convert the image in a double object with *Image = double(Image(:))* to calculate the standard deviation of the image.
- Determine the maximum value in the image with *max(Image(:))*.
- Determine the minimum value in the image with *min(Image(:)).*

- Determine the standard deviation of the image with *std(Image(:)).*
- Calculate the signal-to-noise ratio with *10*log((max-min)/std).*

3.3.7. Hemisphere detection in a cropped image

In Chapter 7, the detection of the egg in the image was performed through the follow process:

- Crop the image to have a better detection of the egg, and to remove undesirable background elements.
- Load the cropped image.
- Transform the RGB image to a binary image with *imbinarize(grayImage)*.
- Fill image regions and holes with *imfill(binaryImage, 'holes').*
- Extract objects from binary image by size with *bwareafilt(binaryImage, 1)*.
- Measure the centre, the perimeter, and the minor and major axes (width and length) of the egg in the binary image with *regionprops(logical(binaryImage), originalImage, 'all').*
- Calculate the perimeter of the egg in metres:
  The diameter in pixels found with r*egionprops* was converted to metres using Pythagoras's theorem to the geometry exposed in Equation 65:

$$perimeter\ (m) = \frac{distance_{camera-sphere}}{focal\ length} \times \frac{\left(sensor_{width} \times perimeter\ (pixel)\right)}{Image_{width}} \quad \text{Equation 65}$$

- Calculate the major axis of the egg for each point.
- Calculate the rotation matrix of the egg.
- Display the egg in the cropped image.
- Plot the major axis of the egg in that cropped image.

The diameter of the egg was plotted by:

- Make an angle array of the same number of angles as there were pixels in the perimeter. The function used was theta = linspace(0, 2*pi, ceil(props.Perimeter));
- Determine the coordinate of each point, in x and y, constituting the perimeter with Equation 66:

$$\begin{cases} \frac{minor\ axis}{2} * cos(theta) + O_x \\ \frac{major\ axis}{2} * cos(theta) + O_y \end{cases} \quad \text{Equation 66}$$

  Where $O(O_x, O_y)$ is the centre of the egg.

- Multiplication between the coordinates obtained and the rotation matrix.

The rotation matrix of the egg was coded manually using the following form (Equation 67):

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \quad \text{Equation 67}$$

3.4. Method statement

3.4.1. Sphere diameter measurement (single camera)

The single camera experiment was comprised of a diffuse white polystyrene sphere with a diameter of 0.09 m, and the Raspberry Pi V2 pinhole camera located on the x axis. A sphere of 0.09 m was chosen because it was the smallest size available from the supplier.

The cameras were positioned in front of the sphere, in such a way that the geometrical centre of the sphere was aligned with the optical centre of the camera. The distance between the sphere and the camera was measured with a resolution of 1 mm.

The cameras were located at an object distance varying from 0.2 m to 2 m, in steps of 0.2 m from the sphere. This distance range was chosen to visualise the impact of the distance on the measurement, as well as the evolution of the geometrical error related to this parameter.

As identified in Section 4.2, the Raspberry Pi V2 was chosen because it is commonly available, cheap, straightforward to set up and use, with fixed optics that are characterised by the pinhole camera model. It is noted here that a consequence of the design means that the camera has a very large depth of field although best focus for this camera version is typically found in the first few metres of the image. The wide field of view (horizontally 62°, vertically 49°) makes this camera very suitable to internal confined space work hence its selection.

The outcome criterion from this real and virtual experimentation was the comparison between deviation of the measured observable sphere diameter and ɵ to the theoretical values. The theoretical observable sphere diameter and ɵ were calculated using the geometry shown in Figure 3.1, and Pythagoras's theorem (Equation 65).

This overall method is likewise used for the circles, cubes, eggs, and the spheres of the balls bar, and will be explained in more detail in Chapters 6 and 7.

3.4.2. Cameras position optimisation (multi-camera)

This experiment was about comparing the reprojection errors of the detection of the centre of five zone circles printed on a calibration board using an array of eight Raspberry Pi V2 cameras (with the same characteristics as those used for the sphere experimentation), were placed around the real zone circle calibration board. Two cameras were used at each position defining a triangular pattern of analysis. The distance between each camera on the z axis was 0.25 m. The camera-object distance varied from 0.2 m to 1.8 m (limited by physical room dimensions) incrementing in steps of 0.2 m. Multiple repeats (n = 3) of the experiments were completed to determine any variance in the output. The calibration zone circle board was placed in the middle of the measurement volume, approximately equidistant to each camera.

Noted that both experiments took place in real world and the Blender equivalent, which are defined in Chapter 4, 5, 6 or 7.

Chapter 4

Exploring Blender as a potential digital twin based for multi-camera metrology system

In machine vison, digital twins bring flexibility, agility, space-saving, optimisation and reliability, key parameters needed to face the challenges of the tomorrow's industry. However, the existing camera systems models are predicated on designing a whole virtual camera from the original camera sensor through to the radiometric characteristics of a real camera. This is a complex process, that is not necessarily accurate and can be time-power consuming.

An alternative was identified to overcome these issues, specifically 3D animation software which uses a virtual camera for rendering. Diverse software exists such as Autodesk Maya or Cinema 4D, but Blender met the research requirements by its capacities to be used in a large range of applications. Nevertheless, there is limited evidence that Blender itself has previously been considered as a viable metrology compliant environment for multi-camera system modelling.

In this chapter, the initial investigation and suitability of using Blender to model multi-camera measurement systems has been explored. This exploration is in three parts, where an exploration of how to use Blender (with a list of its tools) will be presented in the first section. The second and third parts present how to create a virtual camera in Blender, how to set up a light environment in this virtual environment, as well as how the texture management work in Blender. Finally, the fourth part describes the experiments conducted to validate the use of Blender as a possible digital twin base.

The following challenges are addressed:

- Is Blender suitable to model metrology multi-camera system?
- Are the optical laws respected?
- Which blender tools are needed to model a metrology multi-camera system?
- How to model an illumination environment in Blender?
- How to model a camera in Blender?

These questions lead to the following potential innovations:

- Blender has potential to be used in metrology applications.
- Characterisation of the Blender response through two experiments.
- The real model and its virtual twin are different. Blender cannot therefore be used as a digital twin base due to a lack of realism and incorrect definitions of modelling limits.

4.1. 3D animation software: Blender

Blender was developed in 1994 by the Blender Foundation. It is a free and open-source 3D computer software used for creating animated films, visual effects, art, 3D-printed models, motion graphics, interactive 3D applications, virtual reality, and video games. It is written in C, C++ and Python, and runs on Linux, macOS, Windows, FreeBSD, OpenBSD, NetBSD, DragonFly BSD, and Haiku operating systems [135]. Blender combines the functionality to model the interactions of scenes with light sources, and generate simulated images based on modelled cameras using its internal rendering engine based on ray-tracing and Python scripting options.

As explained in Appendix 5, ray tracing allows the modelling of light sources and the interaction of the light with 3D objects in a visually realistic way, to create photorealist scenes. The Python scripting capability, based on API (application programming interface) commands, can be used to customize the application, and write specialized tools, bringing the freedom to manipulate and automate the scene created. This enables many variants of a simulated scene and the associate network of cameras and light sources, that form the measurement system, to be automatically created [135 - 137].

This provides a potential environment for the automated exploration of design choices when creating a camera-based 3D measurement system. Recent digital applications using Blender relevant to metrology have been developed, such as BALINDER [141], an add-on for Blender allowing different depth sensors to be loaded from pre-sets; customized sensors can be implemented and different environmental conditions (e.g., influence of rain, dust) can be simulated. In addition, BlenderProc [142], a procedural pipeline helping in generating real looking images for the training of convolutional neural networks. The applications of this work are numerous, such as segmentation, depth, and normal and pose estimation.

In this work, it is noted that the Blender version 2.9.2 was used.

4.1.1. Blender tools

The principal tools of Blender which are relevant to that research are [160]:

- Composition (post-processing).
- Modelling.
- Rendering.
- Shading.
- Scripting.
- Simulation.

Modelling

Modelling is the basic tool for creating game environments, artistic scenes and animated characters. The model consists of points, lines and polygons, and are created from basic shapes such as a cube or cylinder, modified with the modelling tools. Different modelling modes exist, depending on the applications, as shown in Table 4.1 [161]:

Table 4.1: List Blender modelling modes

Icon	Name	Details
	Object	The default mode, available for all object types, as it is dedicated to Object data-block editing (e.g., position, rotation, size).
	Edit	A mode available for all renderable object types, as it is dedicated to their "shape" *Object Data* data-block editing (e.g., vertices/edges/faces for meshes, control points for curves/surfaces, strokes/points for Grease Pencil).
	Sculpt	A mesh-only mode, that enables Blender's mesh 3D-sculpting tool.
	Vertex Paint	A mesh-only mode, that allows you to set your mesh's vertices colours (i.e., to "paint" them).
	Weight Paint	A mesh-only mode, dedicated to vertex group weighting.
	Texture Paint	A mesh-only mode, that allows you to paint your mesh's texture directly on the model, in the 3D Viewport.
	Particle Edit	A mesh-only mode, dedicated to particle systems, useful with editable systems (hair).
	Pose	An armature only mode, dedicated to armature posing.
	Draw	A Grease Pencil only mode, dedicated to create Grease Pencil strokes.

When the model is created, modifiers can be added. These tools are automatic operations that affect the object in a different way of editing, performing curving, smoothing and many other effective surface-related edits to blend our model.

With UV and shading editors, textures are applied on the model created. The UV editor is used to map 2D assets like images/textures onto 3D objects. The UV unwrapping can be defined automatically according to the edges naturally defined by shape used, or manually by the user. UV unwrapping is the process of applying a 2D texture model to a 3D mesh. U and V letters refer to the horizontal and vertical axes of the 2D texture (2D space), as x, y and z define the coordinate of the 3D mesh (3D space). UV unwrapping tells the software where to apply the texture, defined in the shading editor [162-163]. Noted, UV unwrapping and texture are intertwined.

In the shading editor, two "nodes" are typically created. The "node" corresponds to an operation on the material, changing how it will appear when applied to the mesh, and passes it on to the next node.

The Material Output is the output of the node system, applying the texture on the object. The Principled BSDF (bidirectional scattering distribution function) (explained in Section 4.3.3) is a complex node, based on the Disney principled model PBR (Physically based rendering) shader, allowing the creation of a wide variety of materials. The base layer gathers diffuse, metal, subsurface scattering, transmission, specular, sheen and clearcoat effects.

To generate complex texture, diverse nodes can be linked to the Principled BSDF such as roughness or normal map to define the roughness of the object and its defaults on its surface. Moreover, size of the image textures imported as well as colour distribution can be modified with nodes.

Rendering engines

The rendering window is the visualisation of the 3D scene into a 2D image through the virtual camera lens. Four render engines are typically available, and they rendering images presented in Figure 4.1:

- Workbench.
- Eevee.
- Cycles.

Figure 4.1: Renders with Workbench, Eevee, Cyles and Persistence of Vision (left to right) (Source: [164]).

Workbench is the most elementary renderer that comes with Blender. It is used as a pre-processing device. The render is a real-time 3D viewport but looks like a screenshot of the 3D viewport, with the same options as renderer's options.

Eevee is Blender's real-time renderer, designed through a collaboration with Epic Games. It uses tools to mimic the behaviour of light, shadows, reflections, refractions and more. It is the quickest rendering device, making it very suitable for animation rendering. However, it is less suitable for more advanced rendering techniques such as bounced light and recursive reflections, used in the design of scenes.

Cycles is the oldest of the three renderers. It is capable of achieving realistic results in a limited rendering time. It is a frequently updated path tracer capable of realistic light behaviour calculations. Path tracing is the technique of shooting a ray from the camera, which bounces off surfaces until it hits a light source. The theory links to ray tracing which is explained in more detail in Appendix 5.

Cycles offers advanced rendering features such as bounced light calculations, recursive reflections, accurate refractions, and caustics [164]. Caustics can be described as the patterns of light and colour that occur when light rays are reflected or refracted from a specular surface onto a diffuse surface.

Script

The Python scripting capability, based on API commands, can be used to customize the application, and write specialized tools, bringing the freedom to manipulate and automate the scene created.

Simulation

In Blender, it is possible to simulate a number of different real-world physical phenomena such as hair, grass, flocks (group of birds), rain, fire, smoke, dust, water, cloth, and gravity [165].

Now that Blender's key features relevant to this research have been identified, an explanation of scene creation will be given, with some examples of applications of Blender in the movie and game industries.

4.1.2.Creation of a scene in Blender with applications

The Blender software must to be installed on computing hardware with the [166] hardware needs detailed in Table 4.2.

Table 4.2: Hardware requirements

Minimum	Recommended	Supported Graphics Cards
- 64-bit quad core CPU with SSE2 support. - 8 GB RAM Full HD display. - Mouse, trackpad or pen + tablet. - Graphics card with 2 GB RAM, OpenGL 4.3. - Less than 10-year-old.	- 64-bit eight core CPU - 32 GB RAM. - 2560×1440 display. - Three button mouse or pen + tablet. - Graphics card with 8 GB RAM.	- NVIDIA. - AMD. - Intel. - macOS.

When Blender is opened, the interface is displayed in Figure 4.2:

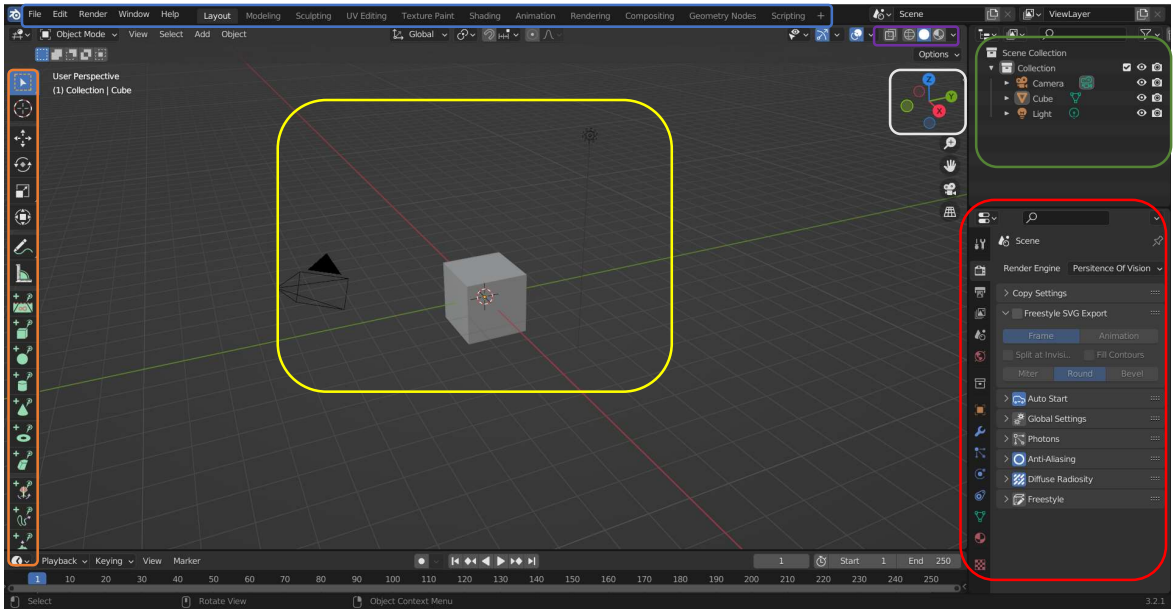


Figure 4.2: Blender version 2.9.2 interface

The key elements of this interface are:

- In orange: Tools for the modelling such as extrude, cute, or bevel.
- In red: Objects, lights, camera and rendering parameters.
- In blue: Principal tools.
- In green: Library of objects created in the scene.
- In purple: Switch between the mesh, object, texture, and rendering view.
- In yellow: Default object (cube, light and camera).
- In white: Coordinate frame. The cube in Figure 4.2 is placed at the origin of the frame.

The modelling process of objects and their environment can be summed up as follows:

- Modelling process:
    - Identifying the basic shape of the element wanted to model.
    - Using the object image as reference or information about the object to model it.
    - Modelling.
- Adding texture according to the real object or the personal tastes.
- Adding light according to the real world or the personal tastes.
- Rendering process.
    - Choosing the location of the camera.
    - Choosing the rendering engine and the rendering parameters.

Modelling process

Before completing any modelling work, it is important to have an understanding of the object of interest. This understanding comes through the definition of the size and proportion of the object, an idea of how to model it, which texture is on it, and how the light interacts with it.

When this pre-work is done, the modelling in Blender can start. The 3D animation software uses pre-defined shapes as a base. For instance, a knife can be designed from two cubes, one for the body, and another one for the blade. The cube can be modified with the correct tools, to take a curved shape, and become a blade.

The same principal is used here. The reference shapes are a plane, a cube, a circle, a UV sphere, an icosphere, a cylinder, a cone, a torus and a monkey. A UV sphere is a sphere with the following two attributes: segments, defining the number of vertical channels running between each pole; and rings, the number of parallel channels running horizontally, parallel to the primitive's equator [167]. In comparison, in Blender, an icosphere is defined as a polyhedral sphere, Platonic solids composed completely of flat, straight, and congruent sides [167].

The difference between an UV sphere and icosphere is the definition of the faces. The UV sphere has faces with 4 edges, giving a smooth appearance to the object, where as an icosphere is composed of triangular faces, with a jagged appearance.

When the default object is put into the scene, an image (such as pdf, jpg) can be imported from outside of Blender, and be used as a model to design the object. A ruler tool is available to scale the proportions, the size, and the dimensions of every mesh created with or without model.

When an element is created, it is created in the middle of the coordinate frame (in the white frame in Figure 4.3). Its translation, and rotation coordinates can be modified in the right panel in the red frame.

When all the elements are in the scene, the modelling can start. The modelling options change from object mode to edit mode, and the object is orange, indicating that it was selected.

All objects created are added to the library (green frame in Figure 4.3). Moreover, only one object can be selected and modified in the edit mode at the time, except if they are linked. This can happen when, in edit mode, an object is duplicated. Then, they have the same object base, and work together.

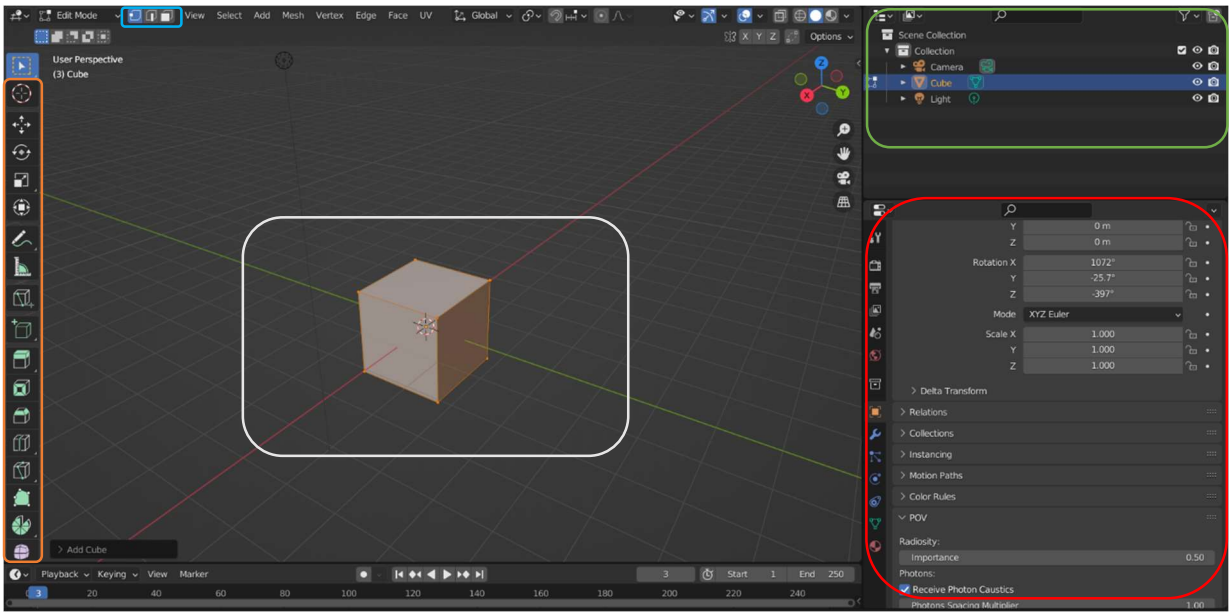


Figure 4.3: Edit mode

Different modelling tools are available such as the extrusion, making a line, the knife to cut following a random pattern, the loop cut to slit the object into a certain number, the meter to measure lengths, widths, and other dimensions; and the bevel. All these tools are located in the orange frame in Figure 4.3. Moreover, it is possible to modify the object with only its faces, points, edges or all of them. When the object is modelled, textures can be added to it.

Texture processing

Adding texture to an object can be achieved by using the  icon in the right panel in the red frame in Figure 4.3. This section is composed of a number of options.

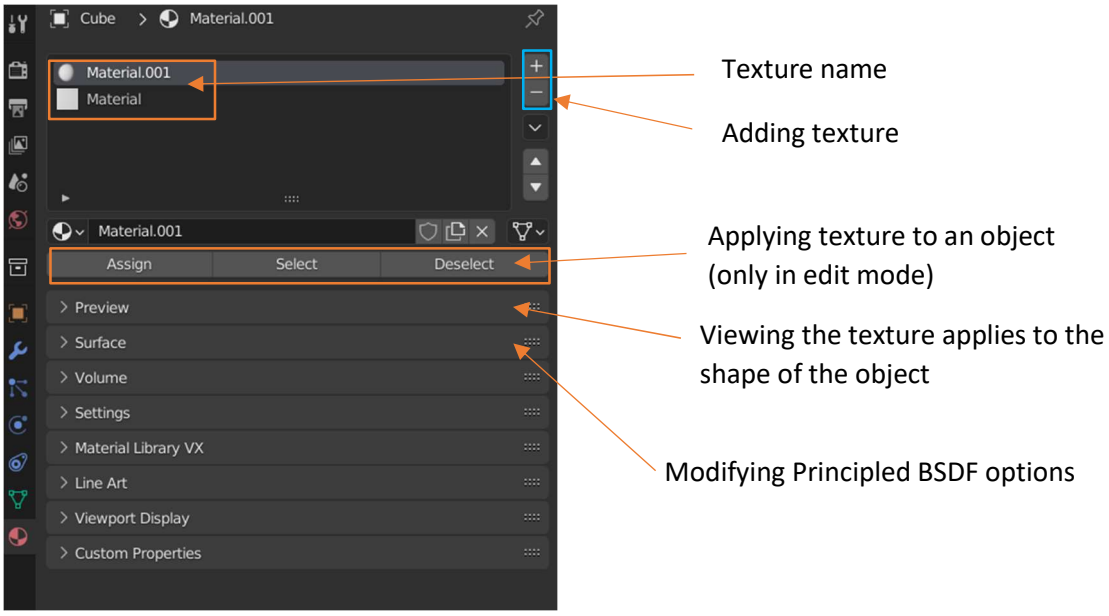


Figure 4.4: Texture editor

However, textures can also be added in the shading editor, accessible in the tool bar in the blue frame in Figure 4.2. In this editor, nodes can be added to have more realistic results with more control, as shown in Figure 4.4.

Light and camera

Lights and cameras are added from the mesh menu. Different types of lights exist: Point, Sun, Spot and the Area, accessible in the same menu in Figure 4.4. The type of light, the colour, the power, the diffuse, specular, and volume characteristics, as well as the radius of the light can be defined.

- The Point light is an omni-direction point of lights, emitting the same amount of light in all directions.
- The Sun light provides light of constant intensity emitted in a single direction from infinitely far away.
- The Spot light emits a cone-shaped beam of light from the tip of the cone, in a given direction.
- The Area light simulates light originating from a surface (or surface-like) emitter [168].

The light power, in Blender, is defined in lumen or watt per square metre, corresponding to radiant flux or radiant power. It is different from the luminous power defined in reality by the watt, for the electrical power of the light [168] (explained in Section 4.3).

The camera menu is accessible in a similar manner. The type of camera (perspective, orthographic or panoramic), the size and type of the sensor fit, the Focal length (in mm or degrees), the depth of Field with the F-stop, the distance of focus, and the number of blades with their rotations can be chosen. A library of default camera parameters exists, including information about the type of camera (perspective, orthographic or panoramic), the size and type of the sensor fit and the Focal length (in mm or degrees). However, other parameters must be defined by the user such as the aperture size, the noise and distortion level, or the F-stop.

The location of the light and the camera depends on the user, and the application. The same goes for the number of cameras or lights. A careful starting point to get a better result is to put the scene creation in a box so that the light bounces off the box wall. If something similar is not done, the light will just propagate into the void.

4.2. Virtual life camera

As explained earlier, Blender is a powerful 3D animation software package that can be used to generate complex 3D environments and objects. However, as discussed in Chapter 2, section 2.2, real cameras can be complex to model due to their components, and the mathematical models used to model them. In Blender 2.9.2, cameras have already been built following, in one way or another, the mathematical models that govern them. Users simply need to modify certain options, such as focal length, to adjust the models proposed and capture more of the cameras' behaviour.

4.2.1.Camera definition in Blender

In Blender, the camera can be created in the modelling editor (Layout in Figure 4.5). Noted that the screenshots of the Blender interfaces are meant to show the appearance of the interface and the location of the main tools it contains. In addition, the screenshots were taken with the higher pixel resolution possible, but they are still limited.

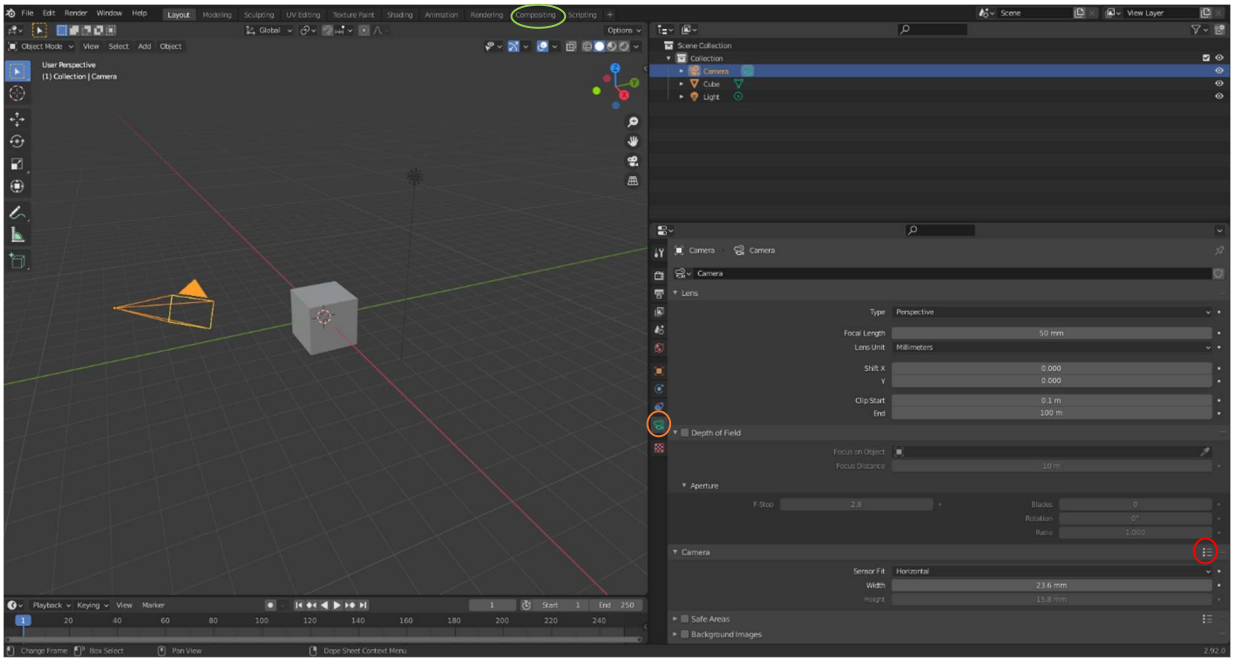


Figure 4.5: Camera panel in Blender

The camera options are accessible on the left panel, under the  icon (surrounded in orange in Figure 4.5). Three main categories of variables are available:

- Lens.
- Depth of field.
- Camera.

Noise and the distortion can be added in the *compositing panel* (surrounded in green in Figure 4.5), after the rendering of the image has been processed. They are categorised as post-processing parameters.

Moreover, pre-camera models (called presets in Blender) can be loaded from the three dots surrounded in red in Figure 4.5. This option was created to match real-world camera models [169]. However, it looks more like a mix of common sensor sizes and specific camera models [170]. They can also be seen as field of view scale factors to match certain real cameras [171].

Lens section in Blender

The lens can be defined as being either perspective, orthogonal or panoramic, with options to adjust the focal length value and its units (mm or degree), the position of vanishing points, called shift, and the exposure time, defined as the clip.

Camera lens types can be:

- Perspective [172]:

the objects seem smaller with increasing distance between them and the observer. Moreover, parallel lines will appear to converge as they get father away.

- Orthogonal [172]:
  the objects appear always at their actual size, regardless of distance. In addition, parallel lines appear parallel, and do not converge like they do with Perspective.
- Panoramic [172]:
  only available with the Cycles rendering engine, three types exist, the Equirectangular (panoramic view of the scenes from the camera location, using an equirectangular projection), the Fisheye (wide angle lenses with strong distortion), and the Mirror Ball (render as if taking a photo of a reflective mirror ball).

The focal length is the real value of the camera focal length found in the datasheet, or calculated through a calibration process. Its units can be millimetres or field of focus, i.e., degree in Blender [209].

The shift means adjusting the position of vanishing points, which are the points where parallel lines converge in a perspective camera's lens. In this panel, the position on x and y of these points can be modified [172].

The clip is the camera exposure time, e.g., the interval in which objects are directly visible by the camera. Its length can be decided by modifying the starting and ending frames [172].

Depth of field

The depth of field defines the focus distance, and the aperture elements (F-stop, Blades, Blades rotation and ratio).

In Blender, the focus distance can be set up in two different ways:

- By forcing the camera to have a particular object in focus all the time, regardless the distance.
- By numerically set up the value of the focus distance. For instance, the focus distance is set up at 0.2 m. This means that everything between 0 m and 0.2 m will be blurred, and everything after 0.2 m will be blurred. Only objects are 0.2 m will be in focus [172].

The aperture can be defined using a number of parameters:

- A numerical F-stop value, which will define the amount of blurring (found in the datasheet of the camera used).
- Shape of aperture is defined by the number of blades. They are polygonal, and used to alter the shape of the blurred object in the render, and the render preview. 3 is the minimum number of blades, enabling the bokeh effect (aesthetic quality of the blur produced in out-of-focus parts of an image) [173-174]; and 16 the maximum number.
  Noted that this is equivalent to a real aperture in a SLR camera lens.
- The rotation of the blades added. They rotate along the facing axis, in a clockwise, and counter-clockwise fashion.
- The ratio of blades to change the amount of distortion to simulate the anamorphic bokeh effect. A setting of 1.0 shows no distortion, where a number below 1.0 will cause a horizontal distortion, and a higher number will cause a vertical distortion [172].

Camera

The *camera* menu is about the sensor definition with the type of sensor used by the camera, called sensor fit in the Blender sub-menu (Automatic, Horizontal, and Vertical), and its dimensions (width and length).

Sensor fit adjusts the camera's sensor fit related to the angular field of view. The Auto fit calculates a square sensor size based on the larger of the resolution dimensions.

The Horizontal fit calculates the height of the sensor based on the aspect ratio of the image resolution, and the numerical value setup by the user of the sensor's width (found in the datasheet of the camera used).

The Vertical fit works in the same way as the Horizontal fit but it is the width which is calculated, and the sensor's length can be manually setup by the user (found in the datasheet of the camera used) [172].

The Size category depends on the sensor fit choice. The width and length of the physical camera are setup here. It is also an alternative way to control the field of view, because they are linked to each other. For instance, a small sensor will have a narrow field of view while a large sensor can provide a wide field of view [172].

Post-processing

Noise and distortion are added after creating the image, in the compositing editor shown in Figure 8. Noted that in that Figure, elements are linked to each other in a node system. This system can be understood as a network, where blocks containing structured data are connected to each other to transform inputs into outputs based on parameters. The output can be, for instance, surface textures, light shape variation, lens distortion, or image noise [175]. In addition, the data contained in the blocks called nodes, can be, for instance, mathematical operations, light effect, camera effects.

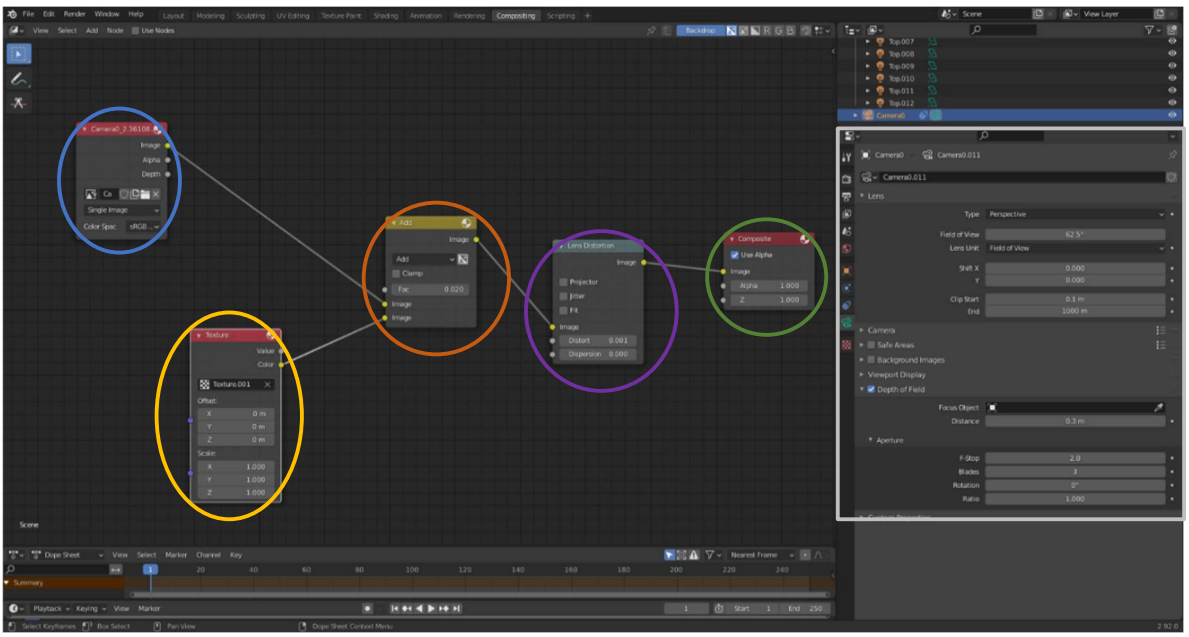


Figure 4.6: Compositing editor

The image noise (surrounded in blue) is added to the image (surrounded in yellow) using the logical operator AND (surrounded in orange). The output of this operation is used at the input of the Lens Distortion node (surrounded in purple). The output of this node will be the image distortion and noise, connected to the renderer by the composite node (surrounded in green).

The image noise is generated through the *composite table* (white square in Figure 4.6) as a texture, in the property editor (surrounded in grey), under this icon  , and added to the *compositing editor* (Figure 4.6) by the texture node (surrounded in yellow).

The ADD node performs the logical operator ADD between its two outputs (the image and the noise texture here). To add it, a Math node has to be used, and its properties has to change for ADD. The value of the node, noted FAC, defines the level of connection between the two inputs [176].

The distortion is added to the ADD node output, through the Lens distortion texture. This node is used to simulate distortions that real camera lenses produce. On this node three properties are available:

- Projector: apply horizontal distortion to the image generated in the rendering process.
- Jitter: add jitter (variation of the phase of a timing signal from its ideal positions in time [177]) to the distortion (Faster but noisier);
- Fit: scale the image to remove the black areas created when positive distortion is applied. Positive distortion is characterised by an expansion of the image, generated by the radical displacement of each point of the image towards the outside of the image centre.

The Lens distortion also has two other properties, distort and dispersion. Distort creates a bulging or pinching effect from the centre of the image, which gives a visual impression that the centre of the image is bulging, with a translation of the image points near the centre up or down the z axis.; and dispersion simulates chromatic aberrations [178].

The definition of the parameters composing the real camera, and how to simulate a virtual camera has been introduced. However, the virtual generation of an image depends on the rendering engine used. A rendering engine is software that draws text and images on the screen through a rendering process. Nevertheless, before introducing the rendering process, the definition of “image noise”, “surface texture” and “node system” has to be defined to highlight the differences between Blender and the real world.

Note that the parameters of the virtual camera model will be defined in chapter 4, for the basic camera model, and in chapter 5, for the final camera model. In this chapter, a definition of each parameter defining a characteristic of the virtual camera has been given to help the reader better understand how a virtual camera is constructed in Blender.

4.2.2.Image noise

In the real environment, image noise is defined as a random variation of brightness or colour information in images. Generally, three main sources generate noise: electricity, heat and sensor illumination levels. Different types of noise exist such as gaussian noise or film grain, but in a digital camera, the noise is generated by the conversion of the photons into voltage, the sensor size due to the effective light collection area per pixel sensor, the sensor fill factor (number of photosites to collect light from a given area), and the sensor heat [–179-181].

In Bender, image noise is added during post-processing, in the compositing window, as a 'texture node'. A "texture node" is a node that adds a texture to an object (in the modelling window) or to the rendered image (in the compositing window). This texture is generated in the texture properties of the properties toolbar, highlighted in white in Figure 4.6. There are different textures such as noise, cloud or magic, which allow the user to generate different types of “noise”. When the texture is generated, it is imported into the “node system” via the “texture node”, as shown in Figure 4.7. The scale and position of the noise in the image can be controlled by the offset parameters (x, y and z) and the scale parameters (x, y, z). So, it seems that noise can be 'controlled' in Blender but cannot be random.

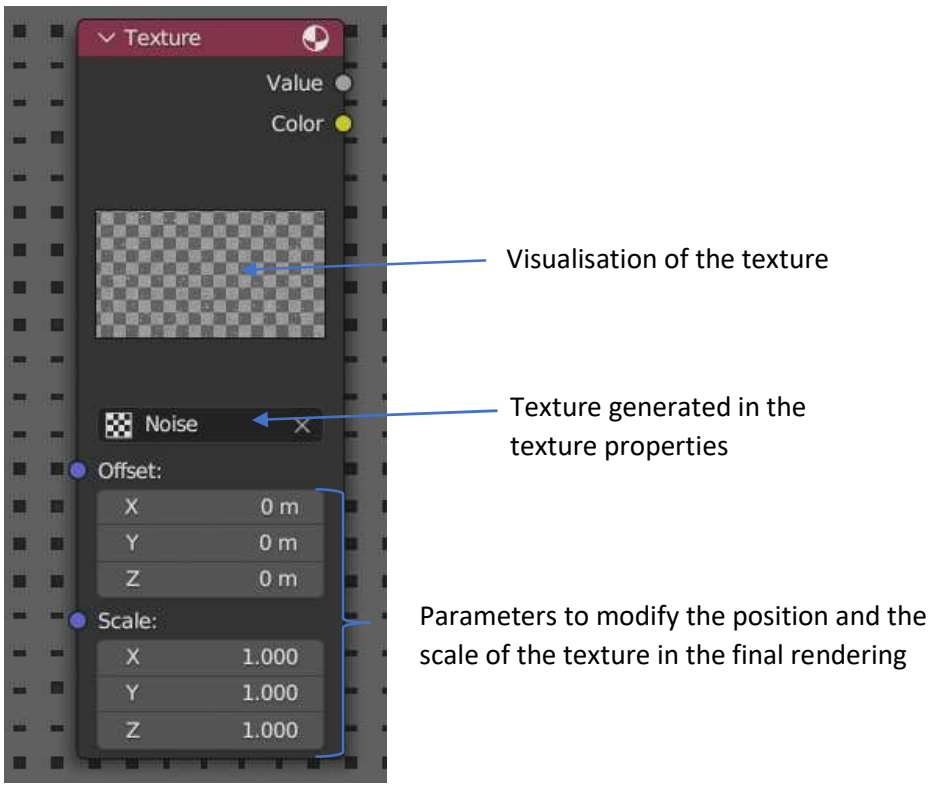


Figure 4.7: Texture node for noise in the Compositing Window (post processing)

In Blender, the image noise is defined as a “node”, generating smoothly varying random values, based on an input value. Four parameters can be adjusted (Scale, Details, Roughness, Distortion), generated 1D, 2D, 3D or 4D noise [182].

4.2.3.Rendering process

The rendering process meets a given need: “Compute the image that a camera would see if it was placed in a given scene”. Its inputs are the components of the full scene, i.e., the geometry, the lights, the materials, and other, and its output, the image or the animation generated.

However, the need of computing an image similar to a photograph means simulating the real world with its complexity. One of the key characteristics is the light, which can be modelled through different models such as [183]:

- Diffraction equations.
- Dispersion equations.
- Relativity model.
- Quantum Mechanics.
- Non-Linear Optics.
- Polarization model.
- Wave-particle duality.

Unfortunately, it is impossible to simulate all of these. Nonetheless, it is not necessarily a requirement to have a perfect result. Consequently, the light modelling can be completed in a straight forward manner using geometric optics [183]:

- Light is emitted somewhere.
- Light travels along a straight line.
- Light bounces off objects.

Moreover, the light models cited above can still be modelled, by importing the equations/formulae which describe them. For example, an effect of dispersion can be simulated by using the geometric optics, if the equation which defining it is known [183].

4.3. Light definition

Light, also known as visible light, is an electromagnetic radiation that is perceived by the human eye [184], and which is generated artificially or naturally such as the LED light units or sunlight. Particles or waves, together or distinct, light can be defined mathematically from different points of view coming from the optical laws of refraction to quantum mechanics with the notion of wave-particle duality.

In the game and film industries, light is not defined as a particle or wave, but as a tool bringing realism to the scene, a bridge between dark and bright part in an image. Its definition in the virtual world is limited as its dimension, type (such as point, sun, rectangle), and power. However, mimicking a real light in a 3D world is a complex process. In Blender, light power is defined depending on the light source. For instance, sunlight is defined in terms of watts per square metre but when considering artificial lighting (spot or area lights) then the lighting is specified in watts noting that this relates to radiant flux and not to electrical energy. Light power can be defined in four different units: watts, the

candela, lumens and lux. They all represent different "light power" or illuminance, and transformations exist between them such as converting lux to Watts.

4.3.1. Light power units

The watts (W) is a define SI related unit of power. It represents the rate of energy consumption in an electrical circuit where the potential difference is one volt and the current is one ampere. It therefore corresponds to the electrical power consumed by a device [185].

The candela (cd) is a primary SI unit and defines the total amount of light emitted in a specific direction [186]. The name candela comes from Latin and can be translated as candlelight.

The lumen (lm) is SI related unit, and also measures light intensity in a similar fashion to the candela but in all directions. It represents to the total amount of visible light perceived by the human eye, and defines the flux intensity, or luminous flux, of the light source 186].

The lux (lux) is SI related unit, and also defines light intensity but in a particular area. As a rule of thumb, 1 lux equals 1 lumen per unit area or 1 lm/m^2 [187]. Table 4.3 gives a definition of each light power units.

Table 4.3: Different light units with their definitions

Watt (W)	Candela (Cd)	Lumen (lm)	Lux (lux)
Electrical power	The intensity of light produced within a specific angle and direction.	The total amount of light that a lamp can produce. Luminous flux.	The amount of light seen on a surface area of 1 square metre from a certain distance.

Relationships exist between all these units. Here the four more important are given:

1. $1\ Candela = 12.57\ lumens$
2. $1\ Lux = 1\ lumens/m^2$
3. $0.9\ lux$ for $1\ W$ for a $1\ W$ LED lamp
4. $1.0\ W/m^2 = 683.0\ lumen/m^2(Lux)$ at the wavelength of $555.0\ nm$ [187]

In Blender, light output is defined in lumens. The four relationships are then used to calculate the Blender light output from the real-world electrical output; and to design a virtual lamp from the variety offered by the software.

4.3.2. Light modelling in Blender

In Blender, light and camera menus are in the same layout and created in the same way.

The light menu (shown in Figure 4.8) is accessible in the property layout (see in the white square in Figure 4.6 in Section 4.2.1), under the icon  .

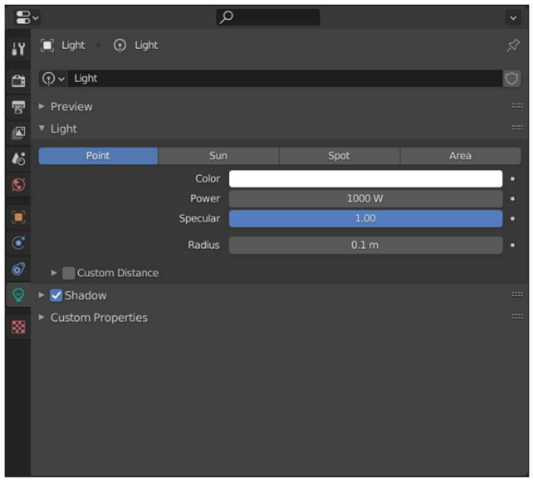


Figure 4.8: Light menu

The type of light (point, sun, spot, area) can be chosen; the power, the colour, the specularity (the reflectivity of the surface), and the size of the light can be set up with a numerical value. In addition, shadow can also be active and defined.

The point light is an omni-directional point of light, emitting the same amount of light in all directions, and it is represented, in Blender by a plain, circular dot. It is used to model simple light sources such as a light bulb [168].

Point light is visualised as a cone-shaped beam, directed in a certain direction. It is not an omnidirectional light, and it emits light in the same direction at a greater or lesser angle to its cone shape [168].

Surface light simulates light from an emitting surface (or similar surface) such as a TV screen, office neon lights, windows or a cloudy sky. Surface light produces shadows with soft edges by sampling light along a grid whose size is defined by the user. This is in direct contrast to artificial point lights which produce sharp edges. The shape of the surface light can be a rectangle, a square, a disk or an ellipse, with adjustable dimensions according to the user preferences [168].

Sun light provides light of constant intensity emitted in a single direction from infinitely far away. It can be useful for a uniform clear daylight open-space illumination. In Blender, sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light. Sunlight is defined in watts per square metre, and not in watts as for the other lights. The sunlight power and size are adjustable according to the user preferences [168].

Light and the object-light interactions can be modelled in Blender. The light interacts in different ways with objects due to their surface texture. The object can reflect, absorb, transmit (the light pass through the object without any variation of its direction), diffract, or refract the light ray. Simulating these different effects is not done through the light panel, but in the shader layout, where texture is added to the object surface.

4.3.3.Textures in Blender

Shading window

Textures are applied in the shading panel available in the window surrounded in green in Figure 4.9. The creation of the objects, lights and cameras is achieved in the layout window, surrounded in red in Figure 4.9.

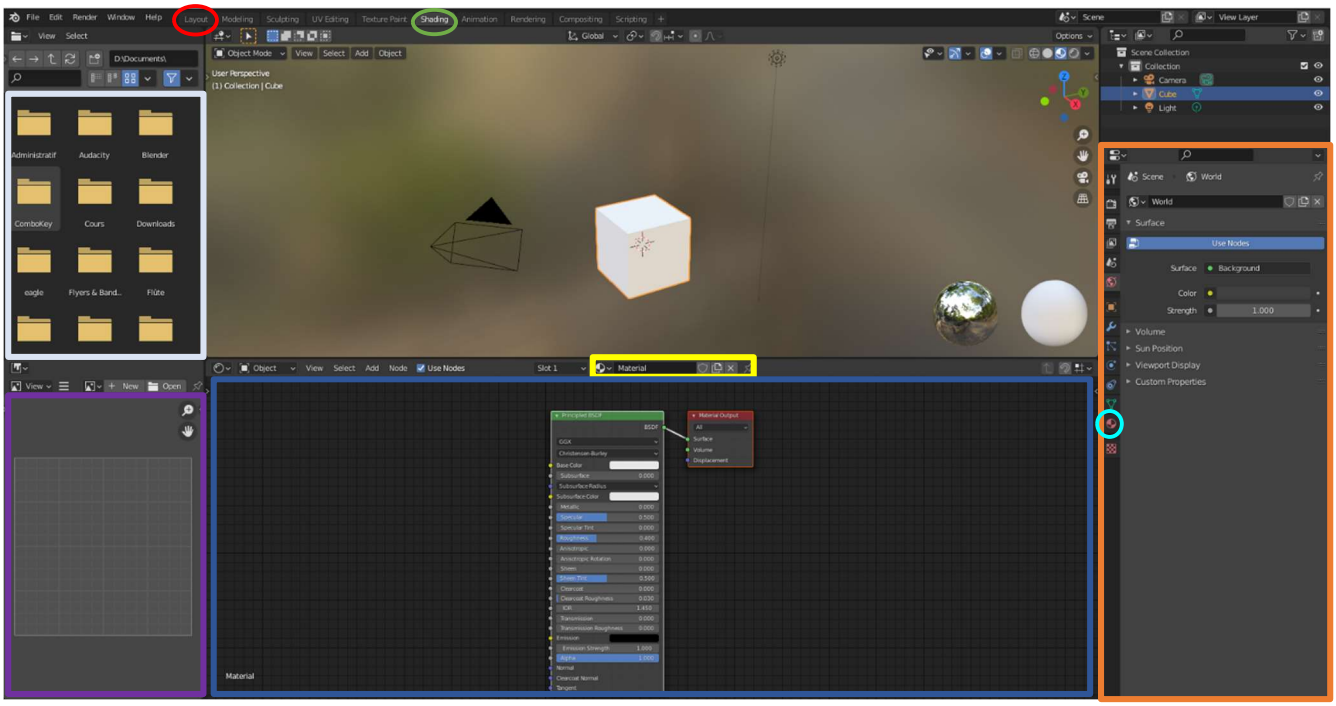


Figure 4.9: Shading window

The shading window works in a similar fashion to the Compositing window explained in Section 4.2.2, with the camera post-processing. Textures are added in the blue rectangle and correspond to a node. Each node is connected to generate the texture, applied on the object through the object panel, accessible in the property panel in orange in Figure 4.9.

In the purple rectangle, the texture can be visualised; and in the yellow rectangle, the name of the texture can be modified, as well as the texture library can be browsed.

In the white rectangle, folders are defined where the texture can be loaded from the computer. Textures are generally images, or maps generated to reproduce the defaults of an object.

The maps correspond to the roughness map, the normal map, the displacement map and the colour map. In 3D computer graphics, these maps are colourless textures (except for the colour map), and are used to simulate the illumination of bumps and dips. In addition, the map method is used to add detail without using more polygons. The normal map corresponds to the defect on the surface of the object, the displacement map, the movement of the texture of the object for a large object such as a forest floor, the roughness map, the irregularities causing light scattering in the surface, and the colour map, the colour of the object as a function of the light (e.g., colour degradation). Texture can also be controlled from the property layout, under the texture icon surrounded in cyan in Figure 4.9.

With the development of Blender 2.9.2, a new texture node called Principled BSDF (bidirectional scattering distribution function) has appeared. This texture combines multiple layers into a single easy to use node, and is based on the Disney [188] principled model, PBR (physically based shading) shader [188]. This innovation allows for the creation of realistic texture, and object-light interaction in an easy way, based on the optical laws.

Definition of the bidirectional scattering distribution function

The BDSF is not a novelty by itself. It has already been used in diverse fields such as the ring laser gyroscopes and telescopes [189], or to characterise material properties [190]. This function radiometrically characterizes the scatter of optical radiation from a surface as a function of the angular positions of the incident and scattered beams, and can be mathematically defined, through radiometric terms, as the scattered surface radiance divided by the incident surface irradiance [168, 189-190].

The scattered surface radiance corresponds to the light flux scattered per unit surface area per unit projected solid angle. The incident surface irradiance refers to the light flux in watts, on the surface per unit of illuminated surface area (not beam cross-sectional area). With these definitions, the BSDF is expressed in Equation 68.

$$BSDF = \frac{\frac{dP_S}{d\omega_S}}{\frac{A.\cos(\theta_S)}{\frac{P_i}{A}}} = \frac{\frac{dP_S}{d\omega_S}}{P_i.\cos(\theta_S)} \approx \frac{\frac{P_S}{\omega_S}}{P_i.\cos(\theta_S)} \qquad \text{Equation 68}$$

Where ω_S is the solid angle defined by the detector aperture; θ_i is the angle of incidence measured from surface normal; θ_S is the angle of detector measured from surface normal; P_i is the incident power irradiating the sample in watts; and P_S is the radiant power measured through the detector aperture in watts as well.

Different types of bidirectional distribution also exist such as the BRDF (bidirectional reflectance distribution function), used for reflected scatter; or BTDF (bidirectional transmittance distribution function) defining scatter transitions through a material [191-193]. BSDF, BRDF and BTDF help to define and characterise the surface texture of an object. Furthermore, based on the laws of optics and physics, they can be modelled virtually to define object surfaces and object-light interactions [188].

Development of the Principled BSDF texture node used in Blender

Disney developed a BRDF model in 2012 and compared it with measured materials to identify its accuracy in terms of modelling the microfacet of the object's surface. The mathematical expression of this BRDF is based on the microfacet model assumption that:
if a surface reflection can occur between a given light vector l and view vector v, then there must exist some portion of the surface (called a microfacet) with a normal aligned halfway between the vectors l and v, $h = \frac{l+v}{\|l+v\|}$. In addition, h can sometimes be referred to as the microsurface normal.

A general form of the microfacet model, for isotropic materials, is given in Equation 69.

$$f(l,v) = diffuse + \frac{D(\theta_h).F(\theta_d).G(\theta_l,\theta_v)}{4.\cos(\theta_l).\cos(\theta_v)} \qquad \text{Equation 69}$$

With the term called *diffuse* is a function of unknown form. In the case of the Lambert diffuser, this term is assumed constant; D is the specular term; F is for the Fresnel reflection coefficient; G is the geometric attenuation or shadowing factor; θ_l and θ_v are the angles of incidence of the vectors l and v with the surface normal; θ_h is the angle between the surface normal and the half vector h defined above; finally, θ_d is the "difference" angle between l and the half vector [188].

However, due to the film industry target market (people watching films), Disney developed an art directable and not necessarily physically correct model. Due to this philosophy, the model developed was called a "principled" model rather than a strictly physical one.

Consequently, the parameters constituting the principled BRDF node resulting from this study are [188]:

- baseColor: the surface colour.
- subsurface: controls diffuse shape using a subsurface approximation.
- metallic: the metallic-ness of the material. An index equal to 0 corresponds to a dielectric, and 1 to metallic. This option is a linear blend between two different models: the dielectric and metallic ones. Noted, that the metallic model has no diffuse component and also has a tinted incident specular, equal to the base colour.
- specular: incident specular amount.
- specularTint: a concession for artistic control that tints the incident specular towards the base colour.
- roughness: surface roughness, controls both diffuse and specular response.
- anisotropic: degree of anisotropy. This controls the aspect ratio of the specular highlight. An index equal to 0 corresponds to an isotropic, and 1 to anisotropic.
- sheen: an additional grazing component, primarily intended for cloth.
- sheenTint: amount to tint sheen towards base colour.
- clearcoat: a second, special-purpose specular lobe.
- clearcoatGloss: controls clearcoat glossiness. An index equal to 0 corresponds to a "satin" appearance, and 1 to a "gloss" appearance.

These parameters are controlled by an index between 0 and 1, called roughness. Its name and principal were inspired by the subsurface BRDF developed by HanrahanKrueger [194], which presents a model for subsurface scattering in layered surfaces in terms of one-dimensional linear transport theory. Moreover, the modelling process (presented into details in Chapter 4 to create a scene in Blender) is based on how the light is reflected on a rough surface.

Thus, the main BRDF index can be seen as defining the complexity of the surface texture in terms of defaults. The higher the number, the more complex the surface is. This results in a kind of change in the perception of the applied texture and then a change in the object-light interaction to achieve the desired aesthetic result.

The principled BRDF allows artists to have access to a wide range of textures without the need to create complicated nodes, resulting in a less realistic result in most cases, because of the inability to easily add and modify each light property defining the surface texture. In addition, the principle BRDF

is physically more accurate by default than other surface texture nodes [195]. However, some options were missing, such as the transmission and emission properties of materials. In 2015, Disney decided to "update" this node, creating the principled BSDF, where specular transmission, subsurface scattering, thin surface approximation, layered shader user interface and material refractive index can be configured [196].

Many commercial and non-commercial rendering engines have developed similar principled BSDF nodes, heavily inspired by the Disney BSDF node. These new nodes are known as Blender's principled BSDF, Autodesk's Standard Surface, Unreal Engine 4's physically based materials, Substance's physically based shaders, and Appleseed standard surface [196].

4.4. Exploration of Blender as a digital model

Very limited evidence [–197-199] exists to show that Blender itself has been considered as a viable metrology compliant environment for multi-camera system modelling. Specifically, there is an opportunity and a need to determine if modelled multi-camera-based systems for object measurement simulated in Blender correlate well with the equivalent real-world multi-camera measurement systems. If such correlation were determined then this would better promote the use of platforms such as Blender for virtual modelling of metrology systems.

This type of platform could then be used as a digital twin to improve decision making, or to assess the costs and limitations of using a robot or a human in a hazardous environment. From a metrology perspective, 3D animation software can be used to simulate different sources of failure for a particular product in a particular environment and help develop solutions.

In this context the investigation of metrology compliance within Blender requires assessment of accuracy and repeatability metrics whilst performing virtual measurements, in addition to considering the potential of the virtual metrology solution. The research presented here considers the initial investigation and suitability of using Blender to model multi-camera measurement systems, with the eventual aim of allowing rapid positional optimisation of in-factory multi-camera measurement systems. The research has specifically two scenarios – these being the measurement of the diameters of three spheres, and the use of a camera verification artefact.

To evaluate Blender, two real-world measurement scenarios were defined, with similar measurement configurations developed and simulated in Blender. The first was the measurement of a sphere diameter by a single camera whilst the second involved the calibration of the camera measurement system based on eight cameras, identical to the one used in the first experiment, using a calibration artefact.

These two experiments considered how to:

- Check whether Blender can be used to design a digital model of the real world or not by using simple and complex configurations.
- Verify whether the laws of optic are respected or not.
- Determine the performance of the virtual environment mimicking the real world.
- Determine the further work needed to improve/correct the answer of the virtual model.

4.4.1.Measurement of a sphere diameter with a single camera

Methodology

The first measurement scenario consisted of the experiment presented in Section 3.4.1 to measure the diameter of a sphere in both the real and virtual worlds. However, in order to check that the sphere was correctly modelled and that the laws of physics were respected, three Raspberry Pi V2 cameras were used instead of one, located on the x, y and z unit axes of the sphere. The diameter of the 0.09 m sphere was determined in the images using the MATLAB algorithm presented in Section 3.3.1.

Real world artefact measurements

The real experiment took place in a controlled environment (illumination intensity control, noise coming only from the cameras, etc.), with two light sources, one coming from the ceiling, and one coming from underneath the sphere to avoid shadows, as shown in Figure 4.10. The top one was a 46 W LED light unit and, the bottom one was a 25 W LED light unit. Both light intensities were measured with a light meter Sekonic C-800-U, and converted into watts using Table 4.3. A dark diffuse background was used to create high contrast between the white diffuse room and the white diffuse sphere.

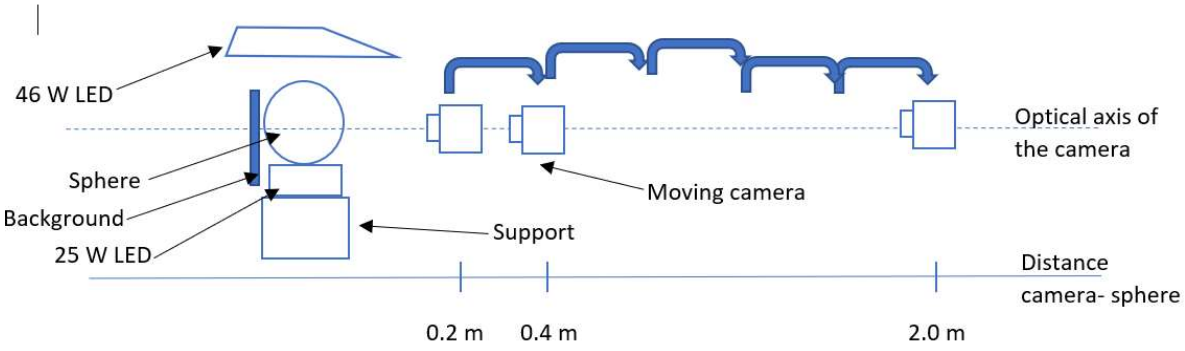


Figure 4.10: Diagram of the real-world experiment

Blender object measurements

The real-world experimentation was recreated in the virtual environment. The experiment again consisted of measuring the diameter and centre of a 0.09 m diameter white virtual sphere from pictures taken by the Blender simulated virtual pinhole camera located at (again) object distances varying from 0.2 m to 2 m, in steps of 0.2 m from the sphere. The whole scene was generated in Blender using a Python script as shown in Figure 4.11. For the purposes of visualisation, Figure 4.11does not include the black background that was normally used to maximise object contrast. Multiple repeats (n = 3) of the experiments were completed to determine any variance in the output.

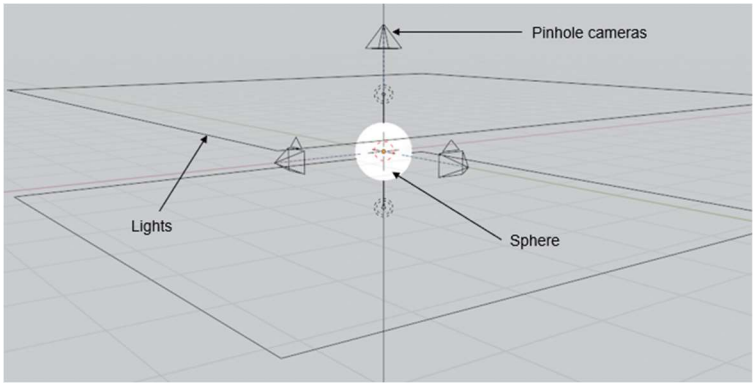


Figure 4.11: Virtual sphere set up in Blender

The parameters of both the real and virtual cameras were focal length of 3.03 mm, image resolution of 1,640 x 1,232 pixels (using 1.12 µm square pixels), and a sensor size of 3.68 mm x 2.76 mm, as defined in the camera datasheet [154]. A dedicated camera model was created in Blender with the following parameters and used to simulate the Raspberry Pi V2 pinhole camera.

- Lens:
  Type: Perspective.
  Focal length: 3.03 mm.
  Lens Unit: Millimetres.
  Shift X and Y: 0.
  Clip Start: 0.1 m.
  Clip End: 1000 m.

- Depth of Field:
  Nothing was set up and/or modified.

- Aperture:
  Nothing was set up and/or modified.

- Camera:
  Sensor Fit: Horizontal.
  Sensor width: 3.68 mm.
  Sensor Height: 2.76 mm.

The 0.09 m sphere diameter was modelled with 32 polygons on a black background, noting that objects created in Blender are all based on a polygon mesh. Hence, the number of polygons represent the resolution of the sphere geometry, consequently the more polygons the sphere is modelled with, then in theory the more “spherical” the sphere will be. Polygon density was considered in a separate series of background experiments with resolutions of 100 and 1,000 polygons. The results of the deviation of the sphere diameter for a resolution of 32, 100 and 1,000 polygons are presented in Figure 4.12 where the x-axis is the camera-sphere distance in metres, and the y-axis is the deviation of the sphere diameter in metres. It is noted that the distance between the camera and the sphere is still between 0.2 m and 2 m with a step of 0.2 m.

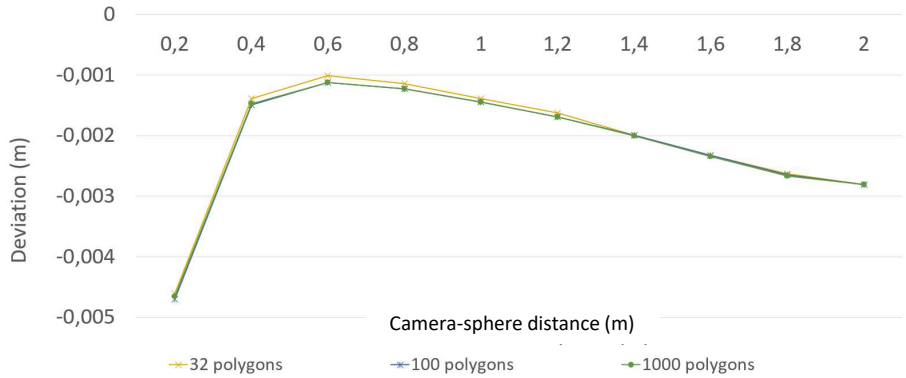


Figure 4.12: Deviation of the sphere for a sphere resolution of 32, 100 and 1000 polygons

Table 4.4: Deviation at 0.4 m, 0.6 m, 0.8 m, 1 m, 1.2 m, and 2 m for a sphere resolution of 32, 100 and 1000 polygons

Sphere resolution	0.4 m (mm)	0.6 m (mm)	0.8 m (mm)	1 m (mm)	1.2 m (mm)	2 m (mm)
32 polygons	-1.4	-1.0	-1.1	-1.4	-1.6	-2.8
100 polygons	-1.5	-1.1	-1.2	-1.4	-1.7	-2.8
1,000 polygons	-1.5	-1.1	-1.2	-1.4	-1.7	-2.8

Regarding Figure 4.12, three sphere resolutions gave similar answers. In term of rendering time, no differences are observable. The same remark is applicable for the computing time between MATLAB and Blender.

Regarding the deviation of the sphere diameter, it is observable that the diameter increased between 0.2 m and 0.6 m due to the geometrical error illustrated in Figure 3.1 of Chapter 3, before decreasing between 0.6 m and 2 m. The three answers are merged on the distance between 0.2 m and 0.4 m; and 1.4 m and 2 m. However, noise is observable on the 32 polygons answers for the other distances.

According to Table 4.4, the difference between the three answers on a distance between 0.4 m and 1.2 m is around -0.1 mm and sometimes -0.001 mm (according to the whole set of data). Consequently, it was shown that the sphere resolution did not have an impact on the deviation of the sphere diameter and was considering as negligible. To save computer power, the resolution of the sphere chosen for the work presented in the thesis was 32 polygons.

As explained in Section 4.3, in Blender, illumination intensity is defined depending on the light source. Consequently, it is difficult to directly correlate illumination intensity between a real-world environment and the equivalent Blender model environment. In this series of experiments, two square area lights with default values initially fixed at 10 kW were placed at 1 m above and below the sphere to illuminate the whole sphere homogeneously. Whilst 10 kW would be a significantly large value in the real-world scenario, within Blender this allowed for even illumination with minimal shadowing.

Through the sphere experiment and the scene construction, it has been identified that in Blender, the illumination set-up and characteristics does not necessarily give the same result as the same illumination in the real world, this partly being a function of the colour transform used in Blender to

generate the rendered image. By default, Blender uses the sRGB colour transform, which was originally designed to approximate the response of a cathode-ray tube monitor [200].

In the Chapters 5, 6 and 7, the virtual scene was rendered with a custom colour configuration called "Filmic Blender", downloaded here [155]. This colour configuration was created by Troy Sobotka, an industry professional and camera enthusiast who wanted to fix this encoder problem. Filmic Blender is a free custom colour configuration, based on a similar colour management configuration to ACES (Academy Color Encoding System), which is the industry standard for VFX [200].

In addition, Figure 4.13 illustrates the difference between these two encoders, by showing the greyscale histogram of a 0.09 m sphere diameter with a resolution of 1000 polygons.

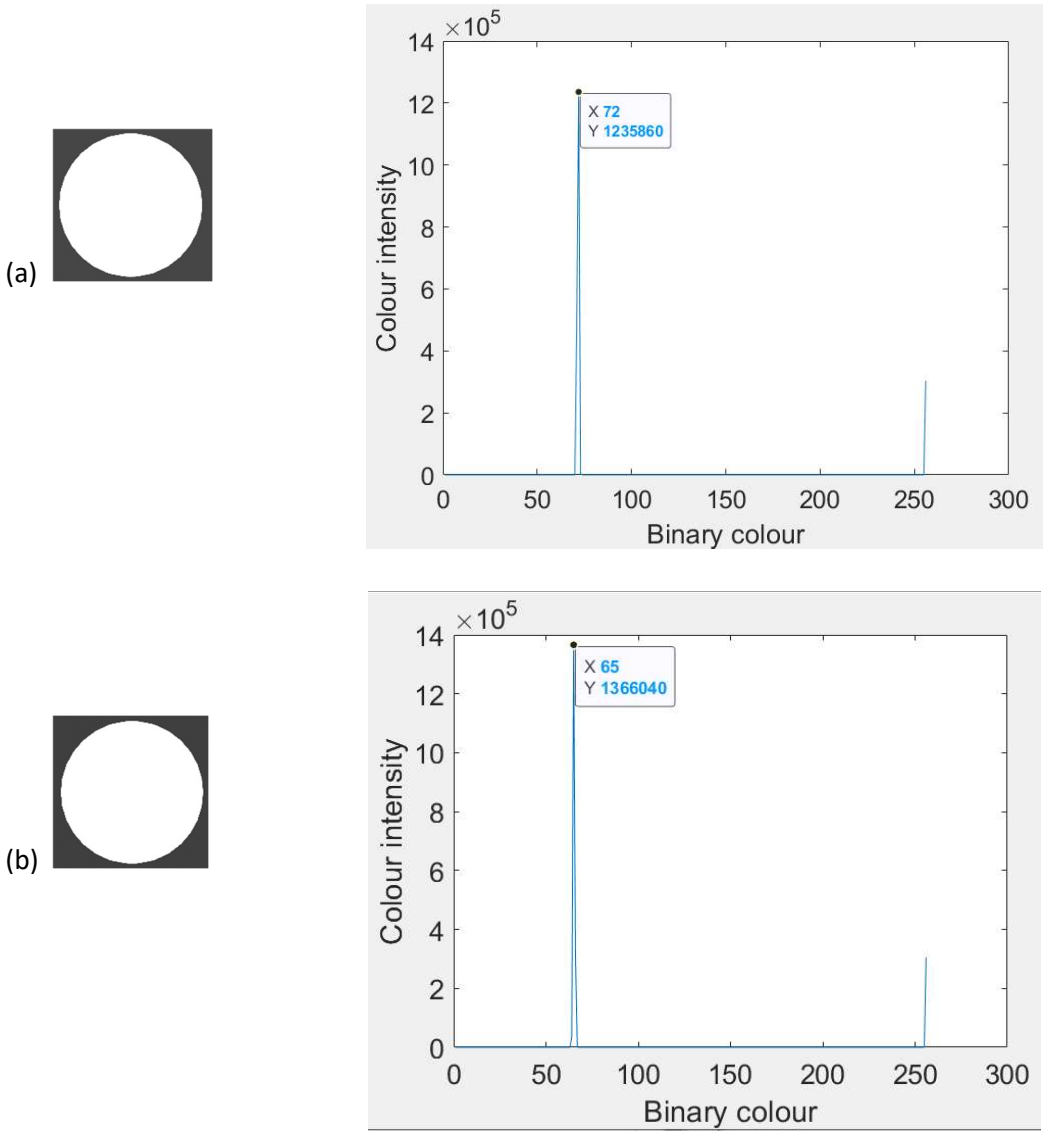


Figure 4.13: a) Filmic colour management, b) sRGB colour management

In Figure 4.13, the x-axis corresponds to a binary colour scale coded between 0 and 256, and the y-axis is the intensity of the colour. The peaks on the two graphs correspond to the colours detected in the two images.

Theoretically, only two peaks at 256 (white) and 0 (black) would have been seen if the two images were pure black and white colours. Two peaks of different intensities can be seen in both images but the first one is different in both images. In Filmic colour management, the peak is at 72; whereas in SRGB colour management, the peak is at 65. The variations in the intensity of the peaks may be due to the colour management chosen. However, the variations are small, implying that the colour management has no real impact on the colour of the image.

The peak at 256 in both images corresponds to a pure white colour, while the other three show that the black colour is not pure black as expected. This variation in black colour could be due to the light power used, exaggerating the details of the sphere, and possibly altering the colours.

The Blender modelling environment was set up to be ideal, namely, no noise or distortion parameters were applied to the environment and the camera model, no surface texture, and light effects were taking into consideration.

The principal render properties of this model were:

- Render: Cycles (Pinhole model render).
- Sampling:
    - Integrator: Branched, Path, Tracing
    - Adaptive sampling: OK
    - Render: 128 samples by pixels.
    - Viewport: 32 samples.
    - Denoising:
        - Render: OpenImageDenoise.
        - ViewPort: Auto.
        - Start sample: 0.

The same MATLAB functions and data processing developed for the real-world scenario were used.

4.4.2.Camera optimisation

Camera networks are becoming more and more popular. Allowing the capture of immobile or mobile targets, they become an important tool in object recognition, object behaviour analyses, the training of artificial intelligent structures, object monitoring processes, and people recognition and tracking [201].

Multi-camera networks allow a larger area to be covered than equivalent mono-camera networks with minimised occlusions; and to perform tracking and 3D reconstruction with high accuracy [202].

However, moving from mono camera system to multi-camera system can be challenging in terms of data processing. Examples of these challenges can be:

- From a hardware point of view, these cameras need to be connected to the same computer, requiring equipment such as an interfacing multiplexer hub physically linked to the computer used.
- From a software point of view, a powerful computer might be needed to store and process the multiple streams of data generated by the system. Noise filters, and optimisation

algorithms might also need to be taken into consideration to denoise the data, and optimise the final result.

In addition, the use of multi-camera systems pose problems in terms of the number of cameras needed and their location to maximise the coverage of the measurement area [202].

This problem of coverage maximization can be compared to the art gallery problem (AGP), also known through its variant the Watchmen Tour Problem (WTP). The AGP is a puzzle, the aim is to determine the optimal number of guards and their route through a polygon, to guarantee the detection of an intruder. However, the algorithms used to solve this problem (AGP and its variant), are unsuitable for the majority of real-world camera placement applications because they are not taking into consideration parameters related to the environment (illumination, light object interaction), and the camera [202].

Over the years, people have tried to solve this camera localisation problem through diverse solutions. These solutions do not use the same methods, or algorithms, and they are not used in the same environment, or for the same purposes. However, they can be gathered into three categories: Generate and Test approaches, Synthesis approaches and expert systems [202].

Generate and Test approaches

Generate and Test approaches are also known as trial-and-error processes. These approaches are based on the simulation of all constraints and models to determine the optimum solution for a given problem. The HEAVEN system [203], and the ICE [204] are examples of the utilisation of this approach. HEAVEN use geometric shapes to simulate the camera sensor, and the environment. The sensor is modelled through a spherical representation, and a geodesic dome is created around the object. Through a subdivision process, the minimum is reached, and the optimum solution is found. ICE works in a similar way with the inclusion of the light environment.

In [205], the optimum number of cameras and their best position are determined through the analysis of the visible point. The visible point analysis technique is based on a hidden point removal approach, and used to detect which surface points on the object are visible from a given camera position. An algorithm is used to find the set of positions that provide optimum surface point density and overlap between views, minimising the total number of camera images required.

Synthesis approaches

Synthesis approaches are another solution, where the constraints are modelled under an analytic function, and the problem is formulated in terms of satisfaction of constraints [202]. This approach is similar to the fine elements problem [202] in mechanics where each requirement generates a geometric constraint, which is satisfied in the 3D domain. Each 3D domain generated per each requirement are then intersected to each other to determine the locations satisfying all constraints simultaneously [202].

This method is illustrated through the work proposed by Erdem and Sclaroff [206], in which camera localisation optimisation is performed through binary optimisation techniques, with the utilisation of polygonal spaces. The first step of this technique is to define the general camera placement problem

by defining assumptions in accordance with the capabilities of real-world camera. The second step is to compute a camera based on certain task-specific constraints, with a minimal camera setup cost. Finally, the minimisation of the problem is achieved through a binary optimisation over a discrete problem space.

In the work presented by Pinto et al [207], a solution approach based on a TOF-based collimated camera protype for online hydrotherapy monitoring is presented. This work was based on Monte Carlo simulations, used the TOF technique for better camera performance, and was composed of three stages. The first addressed a benchmarking of the Monte Carlo tool in order to choose the best physical models. The second focused on the optimisation by the Monte Carlo simulations. Because no parameters really influence the camera performance, multiple random geometries constraints were applied. This strategy was adopted to determine the best performing simulated camera geometry, and to understand which geometric parameters most influenced model performance. The last stage used all of the models simulated as a training data set for a multivariate regression analysis, used to predict camera performances.

Another example can be found in the field of surveillance, where a system has been designed to optimise, with the help of several cameras, the overall observability of all actions performed in a defined area [208]. The system can be used in dynamic and unpredictable environments, and does not attempt to measure or reconstruct surfaces or objects. Here, the gain function, which was minimised, was defined through quality parameters of the view. It considered that each camera's state can be parameterized in terms of an "action". This action corresponded to the modification of state of the camera, moving from the initial coordinate frame, $R0$ (no rotation, and translation of the camera) to a new coordinate frame, $R1$, created by the translation or the rotation of the camera.

Finally, the most common method used to solve camera localisation problems is bundle adjustment. In [209], this algorithm is used for pose estimation, simultaneous self-calibration and reconstruction for multi-camera systems. Based on the Levenberg-Marquardt algorithm [210], it combines the speed of the descent gradient, with the accuracy of the Newton Raphson algorithm to minimise the cost function.

Expert systems

Expert systems address the highest-level aspect of the optimisation problem, the illumination. For instance, these systems can inform whether front or back illumination is more appropriate for different objects, in different situations [202]. These tools are usually used in an advising mode, helping to improve design, simulation, and to understand the impact of the light on the optimisation. An example is shown in [211], where a system was developed to provide advice regarding the best lighting configurations in given circumstances. Its name is LIGHTING ADVISOR, based on a recognition system.

Another example is shown in applications where dynamic objects appear randomly. The aim is to measure the system performance in detecting some user-specified characteristics in this context, and to determine the sensor configurations that would maximize its performance. The optimisation problem addressed is the visibility of occluding objects in the sensor planning. When the sensor view is occluded, object information is lost, requiring the addition of extra sensors. Two techniques were developed to analyse this constraint, a probabilistic approach, where the "average" visibility rates were calculated, and a deterministic approach, to solve worst-case scenarios [212].

A last example of this category is given by Erdem and Sclaroff in their automated camera layout to satisfy task-specific and floor plan-specific coverage requirements [213]. The problem addressed here was similar to the AGP problem explained previously. With a radial sweep algorithm, the visible region for each camera is calculated.

The use of multi-camera networks can be difficult due to optimisation problems, cost, and the choice of technology to be used. Given the nature of this optimisation problem, the digital twin seems to be the most appropriate solution, as it enables a large number of scenarios to be tested in a short space of time, at a reasonable cost in terms of machine power.

Methodology

Real world camera calibration

While the first set of experiments was designed to evaluate Blender's camera performance in the context of measuring objects (spheres) with a single camera, the second set of experiments was designed to evaluate Blender's ability to predict camera positions relative to a camera calibration artefact using photogrammetry techniques. The camera position optimisation experiment in Section 3.4.2 was used for this purpose, with the MATLAB-based algorithm presented in Section 3.3.2 for data analysis - both in the real world and in the Blender virtual environment.

Photogrammetry uses triangulation to reconstruct a 3D scene from several separate images, requiring at least two cameras. Background experiments were carried out with a number of cameras (ranging from three to eight) to determine which combination of cameras was likely to produce lower reprojection errors. The results are presented in Appendix 3 and show that eight cameras gave the best result.

Blender based camera calibration

Within Blender, an equivalent set of eight virtual Raspberry Pi V2 cameras were placed in a triangular set up around the board, to mimic the real-world scenario as shown in Figure 4.14. Camera definitions were those as used for the sphere measurements with data processing following the MATLAB algorithms developed for the real-world scenario (Section 3.3.2).

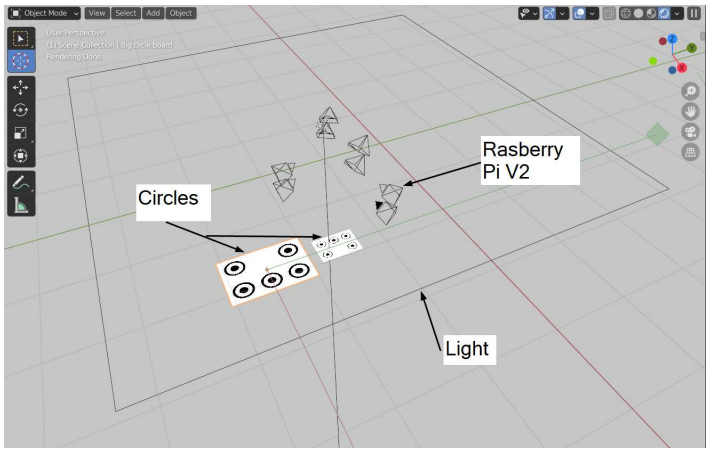


Figure 4.14: Blender scene using the zone circle artefact

4.5. Results

4.5.1.Sphere measurement with a single camera

Real-world sphere analysis

The results from the real single camera measurement of the 0.09 m sphere are shown in Figure 4.15 demonstrating the deviation of the measured sphere diameter from the actual diameter. The x-axis scale represents the distance between the camera position and the sphere, the left side y-axis scale is the difference between the theoretical and measured diameter (deviation or residual), and the right y-axis scale is the area of the sphere within the field of view in camera pixels. The mean value of the multiple repeat results from the three camera positions on the orthogonal sphere axes are shown. The standard deviation of the trial data with respect to the diameter deviation was 0.59 mm, 0.61 mm, and 0.53 mm respectively for Trial 1 to 3, whilst the standard deviation across the trial repeats at each object distance ranged from 0.0002 m to 0.013m, the latter value occurring at 0.4 m object distance and attributed to geometry error as a function of Figure 36 of Chapter 3.

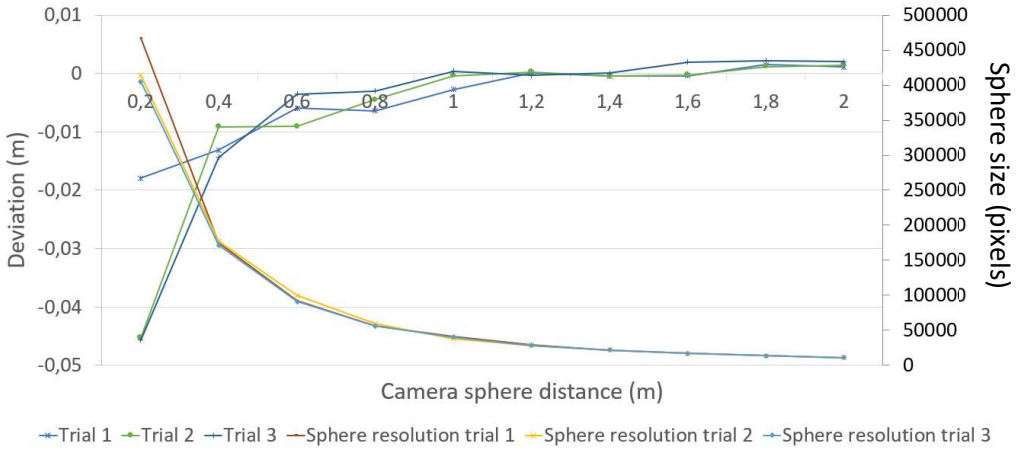


Figure 4.15: Deviation of sphere diameter measurement for the real environment

The data demonstrates that the three trials generated similar results, with similar data trends. The absolute deviation of the sphere diameter decreases with increasing object distance. The initial significant deviation from the true diameter is a function of the camera field of view relationship when viewing a sphere as defined in Figure 3.1 of Chapter 3. The deviation improves with increasing object distance as noted with three sets points in the data summarised in Table 4.5. Here it is noted that there is a discrepancy between the Trial 1 deviation data and that for Trials 2 and 3. This has been attributed to subtle differences of camera set-up between experiments (noting that the three trials were not completed contiguously) and noise elements within the experiment.

Table 4.5: Deviation of real environment sphere diameter and pixel resolution for an object distance of 0.2 m, 0.6 m and 2.0 m

Object Distance	0.2 m	1.0 m	2.0 m
Deviation Trial 1 (mm)	-18	-2.70	1.13
Deviation Trial 2 (mm)	-45	-0.37	1.42
Deviation Trial 3 (mm)	-46	-0.38	2.02
Sphere size Trial 1 (pixels)	467,382	41,115	11,272
Sphere size Trial 2 (pixels)	414,135	38,280	11,232
Sphere size Trial 3 (pixels)	405,048	40,261	10,961

The size of the sphere defines the number of camera pixels that constitute the sphere within the field of view. As expected, and observable in Table 4.5, as the camera object distance increases, the sphere is resolved using fewer pixels thus potentially reducing the accuracy of measurement and hence the accuracy of sphere diameter measurement at longer distances with all three camera positions showing this trend. Moreover, the camera resolution decrease causes increasing image blur as shown in Figure 4.16, likewise affecting the quality of measurement although it is also noted that the V2 version of the Raspberry Pi camera tends to suffer from defocus issues at longer object distances as a function of the original factory build and configuration.

Figure 4.16: 0.9 m sphere diameter showing cropping and blur at 0.2 m, 1 m and 2 m

Blender environment sphere analysis

The results from the Blender modelled single camera measurement of the 0.09 m modelled sphere are shown in Figure 4.17. This likewise demonstrates the deviation of the measured sphere diameter from the actual modelled diameter and decreasing camera pixel resolution as a function of object distance using the same axis descriptions as for Figure 4.15. Unlike the real environment experiments, the Blender analysis did not use three repeats. This is because the algorithmic nature of Blender means that repeating the same analysis multiple times with the same data input leads to negligible variance of data output. This was previously determined through background experimentation. Hence any Blender result graphs will have no errors bars.

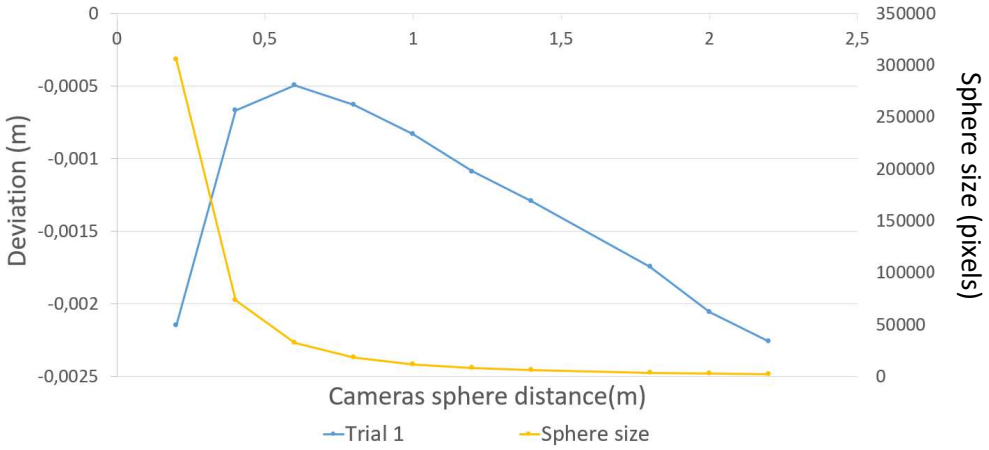


Figure 4.17: Deviation of sphere diameter measurement for the Blender environment

Table 4.6: Deviation of Blender environment sphere diameter and pixel resolution for an object distance of 0.2 m, 1.0 m and 2.0 m

Object Distance	0.2 m	1.0 m	2.0 m
Cam on x (mm)	-2.15	-0.83	-2.06
Cam on y (mm)	-2.15	-0.81	-2.03
Cam on z (mm)	-2.08	-0.74	-2.15
Sphere size (pixels)	305,561	11,870	3,743

Initially, the sphere diameter deviation decreases up to 0.6 m object distance, but then unlike the real-world experiment the deviation negatively increases as the object distance progresses to 2 m. However, it should be noted that the left side y-axis scale factor in Figure 4.17 is an order or magnitude smaller than the equivalent in Figure 4.15, and if plotted on the same scale as Figure 4.15 would approximate to a straight line. In the first instance, the deviation comes from the sphere geometry, and the diameter measurement is function of the camera field of view (see Figure 3.1 of Chapter 3). The subsequent negative increase of diameter deviation is due to the lighting and the resolution of the camera. In addition, it was determined that the same blurring process occurred in the virtual model (similar to Figure 4.16). This is illustrated in Figure 4.18.

It is noted here that during further background experimentation it was determined that too much light within the Blender environment exaggerated the environment of the sphere yet did not seem to saturate the virtual camera image sensor – this being a key limitation of real cameras yet not noted as being a characteristic of the Blender camera model response.

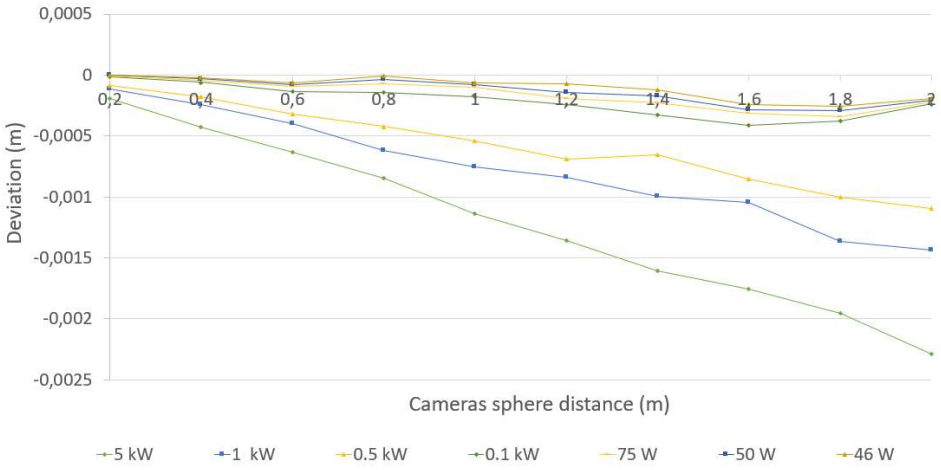


Figure 4.18: Light variation in Blender for a 90 mm circle

Figure 4.18 illustrates this point. Using the Blender setup, and the methodology given in Section 3.4.1, seven light powers (5 kW, 1 kW, 0.5 kW, 0.1 kW, 75 W, 50 W and 46 W) were simulated to determine the impact of light power on the sphere diameter measurement in the virtual environment. However, to further remove the geometry error influence illustrated in Figure 3.1 of Chapter 3, a 0.09 m diameter circle was modelled instead of a sphere - placed on a black background to achieve similar white/black high contrast associated with the previous sphere measurements.

Figure 4.18 shows the diameter deviation is greater for high power illumination (5 kW, 1 kW and 0.5 kW), than for low power (0.1 kW, 75 W, 50 W and 46 W). Thus, the more powerful the light and illumination, the more the details of the circle are exaggerated, resulting in errors in the detection of the circle (and consequently sphere) exacerbated as a function of increasing object distance. This background element of work confirmed that object specific illumination thresholds may exist but can only be determined on a case-by-case basis. The initial choice of 10 kW illumination was originally and correctly justified to maintain even illumination across the entire sphere and remove shadowing but this still generated errors when detecting the diameter of the sphere.

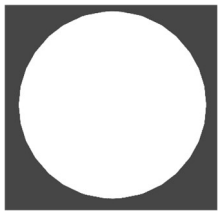
Table 4.6 also reveals that the size of the sphere is not the same between the real and the virtual image even though the camera parameters were set-up as per the manufacturer's specification. For instance, at 0.2 m for Trial 1 (Table 4.5), the size of the sphere in the real image is 467,382 pixels compared to a virtual equivalent size of 305,561 pixels (Table 4.6). As the object distance increases then the difference in pixel count becomes more obvious. This is related to the manner in which Blender generates images, whereby pixel count is linked to the output of the rendered scene rather than necessarily relating it directly to the number of pixels at the camera image plane.

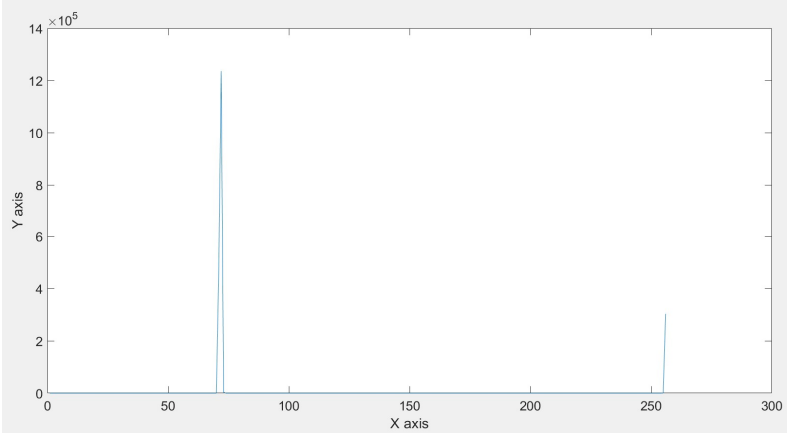
In addition, this difference may be due to the grey scale of the real-world image versus the Blender equivalent. In Blender, the sphere is assumed to be perfectly white [255, 255, 255], on perfectly black background [0, 0, 0], whereas in the real-world case the white and black elements of the image are not perfectly white and black but contain grey scale components of varying values. In MATLAB, to detect the sphere in an image, a binary image process is employed and uses a threshold to determine the grey scale components of the image based on the Otsu method [41]. Due to the different grey scale values in the image, fewer pixels are used to detect the sphere in MATLAB.

This idea is reinforced with the plot of the grey scale histogram of the real and the virtual image in Blender. Figure 4.19 shows the 0.09 m sphere diameter at a camera sphere distance of 0.2 m, in both

environments with their respectively grey scales. Noted here that the modelled resolution of the virtual sphere was 32 polygons.

a)





b)

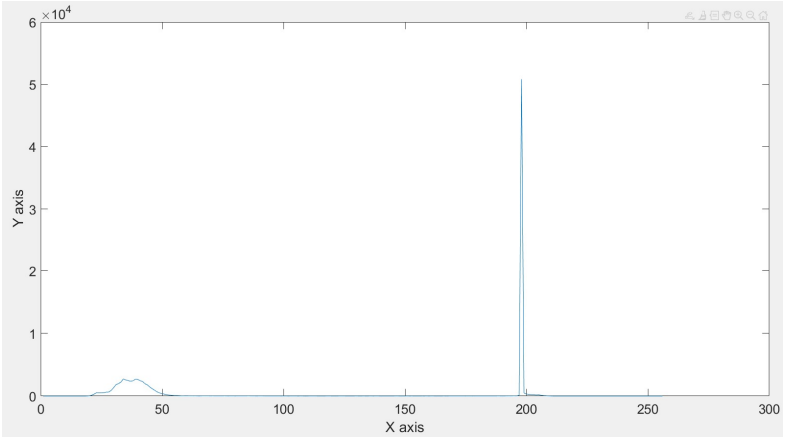


Figure 4.19: Images and greyscale of
a) Virtual sphere with a resolution of 32 polygons, b) Real sphere

According to Figure 4.19, the greyscale of the virtual and real spheres is different. The greyscale of the Blender image is composed of two peaks at 72 and 256, while the real image has its peaks at 37 and 200.

These values correspond to the colour, a pure black will be at 0, and a pure white at 255. In virtual greyscale histogram, it is noticeable that white colour is a pure white, but the black colour is composed of 3 different variations of black. In comparison, the real greyscale histogram shows that the real image was composed of a variety of black and white colours. Hence, the two images are different.

4.5.2.Camera position optimisation

Real-world reprojection error analysis

The results from the real-world multiple camera measurement of the calibration zone circles are shown in Figure 4.20 noting that again for the real-world investigation there were three repeats of the experimentation. The x-axis represents the distance between the cameras and the calibration artefact, and the y-axis represents the reprojection error of the centre of each circle detected. Due to data density as a function of five repeat experiments for each trial, only Trials 1 and 2 are shown here to illustrate both magnitude and trend characteristics.

Two sets of data are presented for each experiment using the two separate zone circle artefacts. The smaller artefact was used for an object distance varying between 0.4 m and 0.8 m (Figure 4.20), the larger artefact was used for object distances varying between 1.0 m and 1.8 m (Figure 4.22).

However, due to the difficulties to read Figure 4.20 and Figure 4.22, only Trial 1 and 2 plotted in Figures Figure 4.21 and Figure 4.23, are used for the analyse. These two trials were chosen because they describe the mean tendencies of the real answer for both artefacts.

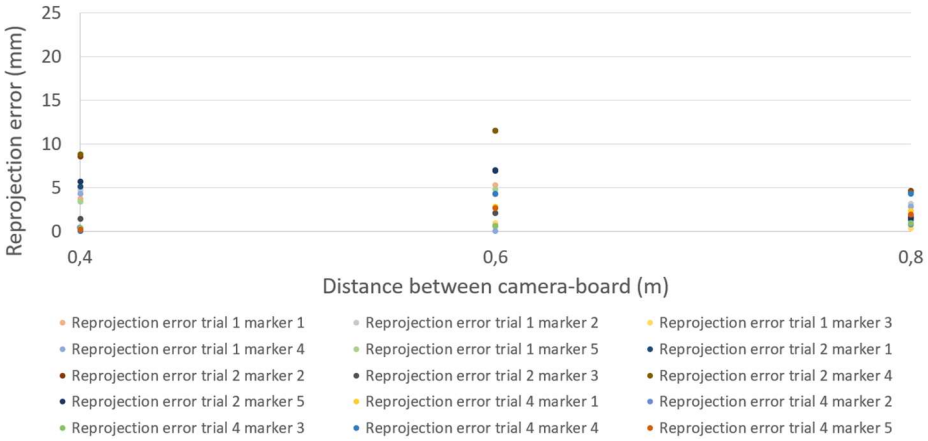


Figure 4.20: Real environment reprojection error (0.4 m to 0.8 m) with the 5 trials

Figure 4.21: Real environment reprojection error (0.4 m to 0.8 m) with the 1 trial and 2 trial

The data (also shown in Tables 4.7 and 4.8 respectively) demonstrates that the two trials generated similar results, with similar data trends for both versions of the artefacts, but not with the same magnitude of results. For instance, for zone circle 1 at 0.4 m, the reprojection error was 5.317 mm for trial 1, 7.002 mm for trial 2 and 0.272 mm for trial 3 respectively. The difference between the results comes from the script used for the detection, and the physical setup. Because the tripod mounted cameras were moved manually from 0.4 m to 1.8 m, it was found to be very challenging to maintain consistent camera rotation and positional parameters (with six translational and rotational degrees of freedom available) during repositioning at each object distance. In addition, due to the variation in camera configuration, the light on the surface of the sphere was not perceived by the camera in the same manner, which led to errors in the detection of the sphere in the image during processing in MATLAB.

With respect to Figure 4.21 and Table 4.7, the reprojection error decreases between 0.4 m and 0.6 m, before increasing. The reprojection error is larger for zone circles 1, 2, 4 and 5, whilst zone circle 3 has the smallest value. This difference is attributed to the fact that the zone circles are paired (1 with 4, 2 with 5) but 3 is an individual zone circle. With increasing object distance, zone circles 1 and 4 start to become more difficult to identify, likewise for 2 and 5. Moreover, zone circle centre detection might not be the true geometric centre (for each zone circle) due to optical changes of perspective and depth of field. In addition, the circles are potentially "ovalized", changing slightly the centre coordinates and the height of the board.

It is noticeable that Trial 2 gives a larger reprojection error than Trial 1 at 0.4 m and 0.8 m, a function of the physical setup, the camera location and orientation, and the lighting condition. In addition, the circle detection script may be sensitive to variation, as a function of incorrect detection of the circles caused by the image contrast and the illumination environment.

Table 4.7: Real-world reprojection error for a distance
between the camera and the sphere from (0.4 m to 0.8 m first and second trials)

Zone Circle (Trial 1)	**0.4 m (mm)**	**0.6 m (mm)**	**0.8 m (mm)**
1	5.32	1.61	4.10
2	4.18	3.15	1.16
3	0.92	0.39	2.17
4	3.95	2.78	1.36
5	4.82	1.71	3.87
Zone Circle (Trial 2)	**0.4 m (mm)**	**0.6 m (mm)**	**0.8 m (mm)**
1	7.00	1.36	12.06
2	11.55	4.66	6.27
3	2.09	1.56	3.95
4	11.50	4.47	6.75
5	6.92	1.49	13.01

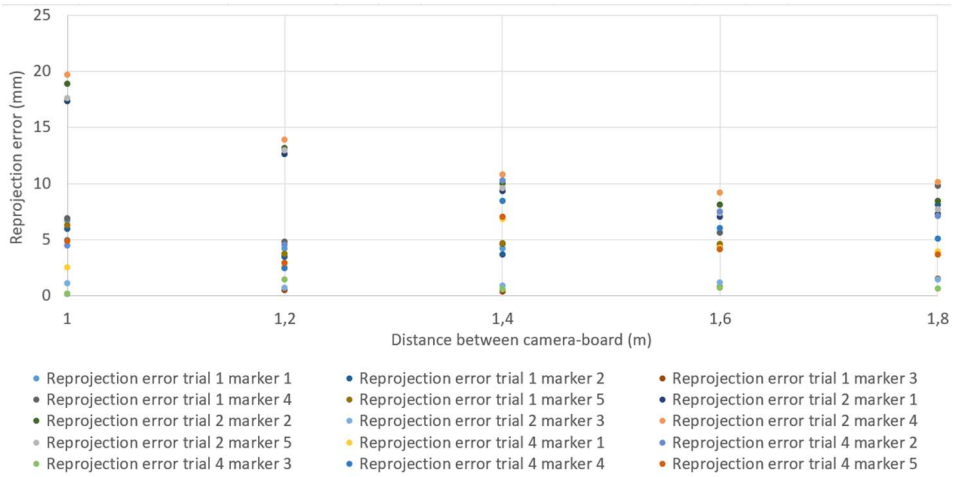


Figure 4.22: Real environment reprojection error (1.0 m to 1.8 m) with the 5 trials

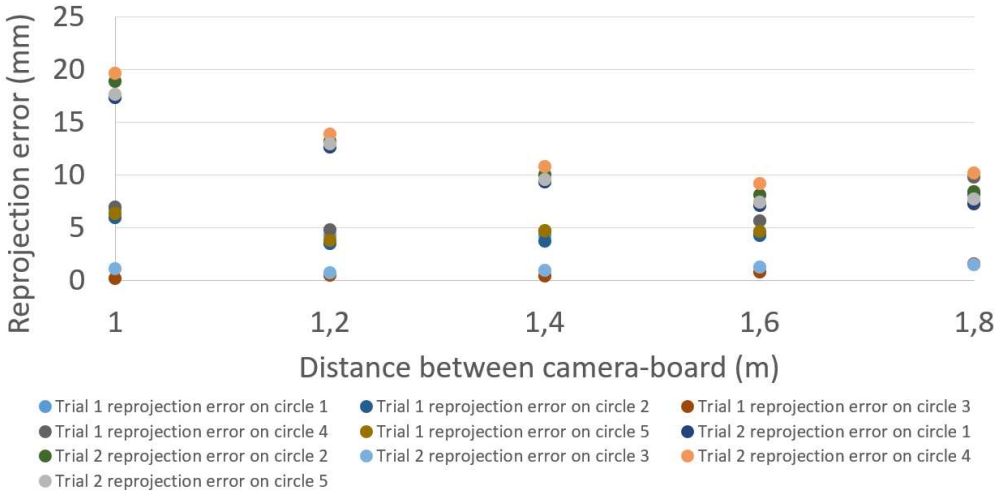


Figure 4.23: Real environment reprojection error (1.0 m to 1.8 m) with the 1 trial and 2 trial

With respect to larger zone circle artefact (Figure 4.23 and Table 4.8) the reprojection error decreases between 1.0 m and 1.6 m, albeit before slightly increasing at 1.8 m for Trial 1 and 2. For Trial 3, the reprojection error increased from 1.2 m to 1.4 m, before slightly decreasing. For zone circles 1, 2, 4 and 5 the reprojection errors were larger, whilst zone circle 3 again had the smallest value.

Table 4.8: Real-world reprojection error for a distance
between the camera and the sphere (1.0 m to 1.8 m – first and second trials)

Zone Circle (Trial 1)	**1.2 m (mm)**	**1.4 m (mm)**	**1.8 m (mm)**
1	4.23	4.21	6.87
2	3.47	3.71	8.16
3	0.50	0.43	1.53
4	4.81	4.65	9.83
5	3.78	4.70	7.31
Zone Circle (Trial 2)	**1.2 m (mm)**	**1.4 m (mm)**	**1.8 m (mm)**
1	12.63	9.35	7.23
2	13.18	10.03	8.44
3	0.72	0.91	1.48
4	13.91	10.80	10.17
5	12.99	9.59	7.71

Similar trends for Trials 1 and 2 are observed across both sets of data shown in Tables 4.7 and 4.8. However, the results for zone circle 3 are of similar values for both trials, and the results for the large artefact analysis shown in Table 4.8 are larger than in Table 4.7 showing that reprojection errors increase with object distance. This is to be expected due to changing pixel resolutions as a function of object distance and orientation of the camera.

In addition, the size of the calibration board seems to not impact the measurement, but the conditions of how the setup is installed appears to be important, and the detection script appears to be sensitive to board size. As explained previously, these sources of errors are the misalignment between the camera and the sphere, the light on the board, and the pixel resolutions. However, the equation used for the conversion of the diameter in pixels into metre is also a potential source of error.

In Equation 70, three parameters can be considered as constant and three as variable.

$$diameter\ (m) = \frac{distance_{camera-sphere}}{focal\ length} \times \frac{\left(sensor_{width} \times diameter\ (pixel)\right)}{Image_{width}} \qquad \text{Equation 70}$$

The three variables are the distance camera-sphere, and the diameter in pixels; while the three constants are the sensor width, the focal length, and the image width.

This equation is also composed of a multiplication of two quotients. The first quotient is the division of the distance camera-sphere by the focal length. The focal length is a small number with a magnitude in the order of millimetres, while the distance camera-sphere varies between small and large distances in metres. Due to its position in the quotient, and its magnitude order (mm), the focal length can be considered as a sensitive parameter, influencing the result. If this value is defined incorrectly, the conversion will be significantly influenced.

The second quotient is the division of the result of the sensor width multiplied by the diameter in pixels, by the Image width. As explained with the first quotient, the image width is also a sensitive parameter, influencing the result in a similar way to the focal length.

Blender reprojection error analysis

The analysis of the reprojection errors determined within the Blender virtual environment is presented in a similar fashion to the real-world analysis within Figure 4.24 (short distance) and Figure 4.25 (long distance), supported by Tables 4.9 and 4.10 respectively.

With respect to both Figures and Tables of data, the reprojection error for zone circles 1, 2, 4 and 5 increases with increasing object distance, whilst zone circle 3 is fairly consistent (and small) in its reprojection values. This is similar overall behaviour to that seen in the real-world experiments. Zone circle 3 may be exhibiting the best results because of its position within the coordinate framework of the artefact and analysis.

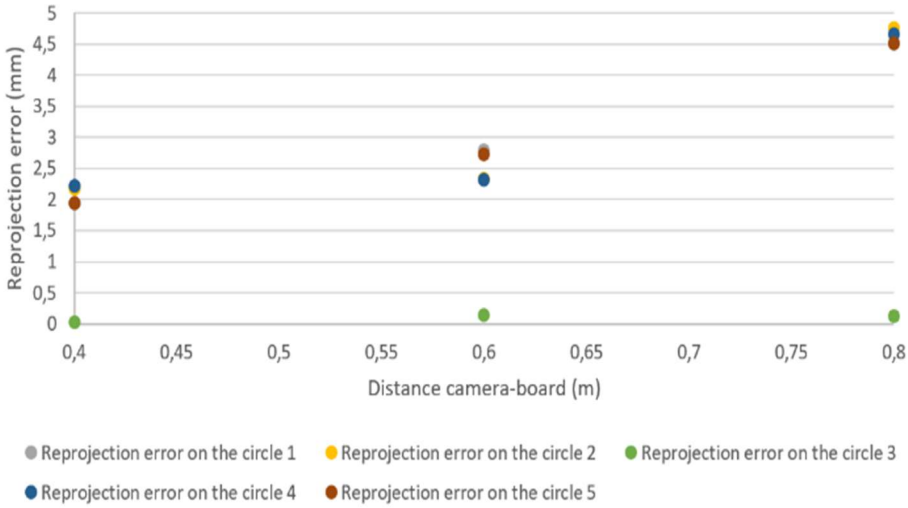


Figure 4.24: Blender environment reprojection error (0.4 m to 0.8 m)

Figure 4.25: Blender environment reprojection error (1.0 m to 1.8 m)

The data presented in Tables 4.9 and 4.10 further demonstrates that the Blender environment reprojection error appears to be influenced by the size of the calibration board. At 0.4 m the reprojection error is between 0.02 mm and 3.0 mm, at 1.2 m, the reprojection error is between 0.05 mm and 0.7 mm.

Table 4.9: Blender reprojection error for a distance between the camera and the board from 0.4 m to 0.8 m

Circle	0.4 m (mm)	0.6 m (mm)	0.8 m (mm)
1	2.64	2.8	4.52
2	2.48	2.34	4.76
3	0.03	0.03	0.13
4	2.54	2.21	4.66
5	2.67	1.95	4.51

Table 4.10: Blender reprojection error for a distance between the camera and the board from 1.0 m to 1.8 m

Circle	1.2 m (mm)	1.4 m (mm)	1.8 m (mm)
1	0.67	0.87	1.64
2	0.77	1.05	1.91
3	0.05	0.02	0.19
4	0.67	0.91	1.81
5	0.67	0.92	1.59

According to Figure 4.24 and Figure 4.25; and Table 4.9 and 4.10, it is also clear that there is a step change in magnitude between the use of the two boards, with the smaller board causing larger reprojection errors. It is anticipated that the small board is more difficult to detect than the larger one due to its size, and consequently there is more potential for the merging of the circles within the camera images as a function of increasing distances and reducing pixel resolution. Clearly, this does not follow the real-world equivalent trends for the two sizes of artefact.

To reinforce these ideas, the following set of images show the evolution of the board pixel resolution for a distance of 0.4 m, 0.6 m, 0.8 m (small board) (Figure 4.26); and 1.2 m, 1.4 m, and 1.8 m (large board) (Figure 4.27) in both environments (first row: Blender; second row: real world).

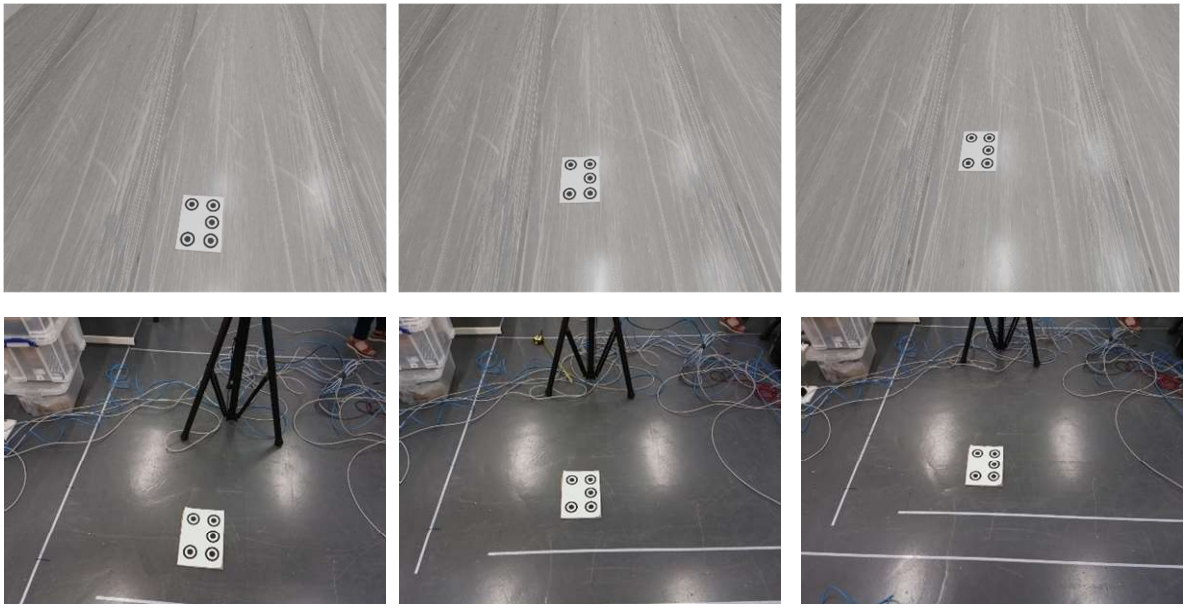


Figure 4.26: Pictures of the small board for a distance of 0.4 m, 0.6 m and 0.8 m in the virtual and real set up

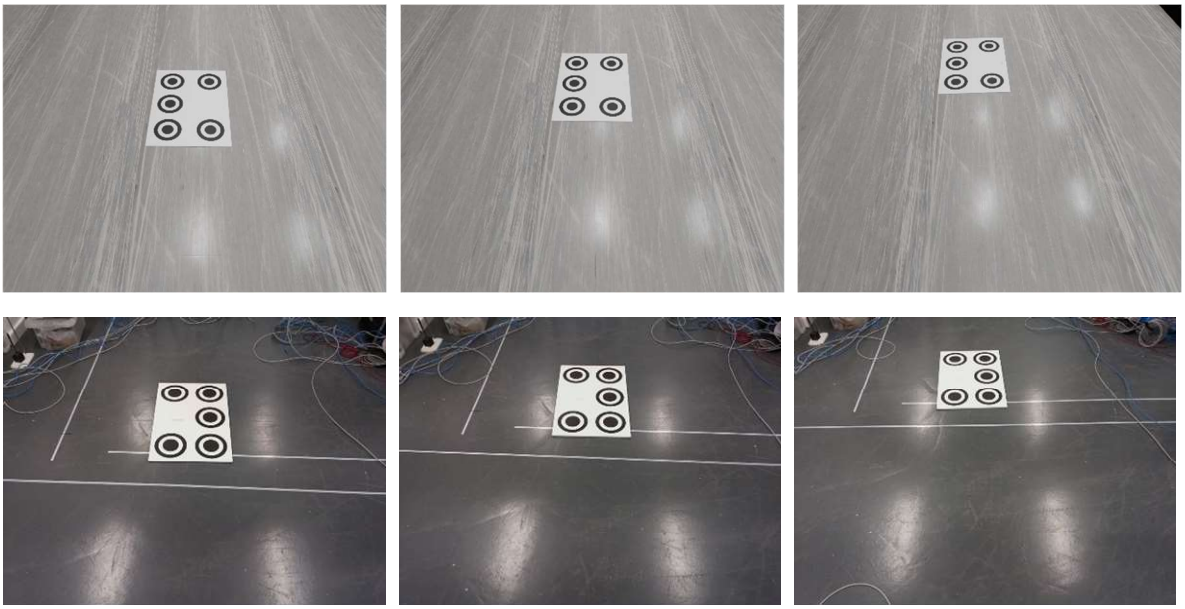


Figure 4.27: Pictures of the large board for a distance of 1.2 m, 1.4 m and 1.8 m in the virtual and real set up

Analysis of the data and also reflecting on the requirements for establishing the virtual models in Blender, reveal that there are (as expected) issues with matching the model to the real-world. For instance, in the real-world – the zone circle artefact is printed on flat laminate board which has a natural element of (diffuse) surface texture – likewise the spheres in the diameter measurements exhibit texture. These real textures can be explicitly measured and quantified in terms of 2D or 3D surface texture parameters such as with the introduction of standards ISO 4287, or ISO 25178 respectively [214-215].

However, in Blender the menu options used in the set-up of the model provide texture descriptors that are considerably more broad ranging in composition and interpretation (based on nodes, light

absorption and reflection properties) that are significantly subjective and qualitative in nature, and do not appear to directly relate to the quantified metrics [216]. Texture adjustment can be achieved with the (Blender) size tool, or with the smart match tool.

4.5.3.Further discussion of the results

With respect to the sphere experiments, the virtual and real-world responses have significant similarities but also differences. Trends wise the two responses behave similarly from 0.2 m to 0.6 m camera-object distance but differ thereafter. The initial error input at very close object distance (visible in both environments), predominantly comes from the camera/sphere geometry relationship shown in Figure 3.1 of Chapter 3. The second error input at large object distance (visible in both environments) predominantly comes from the camera/sphere geometry relationship shown in Figure 3.1 of Chapter 3, and the decreasing image pixel resolution as a function of object distance. This explanation is further facilitated by the size of the sphere being smaller (based on pixel count) in the virtual environment than in the real environment even though geometrically speaking the virtual sphere was the same specification (0.09 m diameter). This means that the virtual sphere in the virtual image, for the same image size and resolution, is composed of fewer pixels than in the real image. This is partly caused by the manner in which image generation is defined in Blender rather than the resolution of a camera.

The real-world scenario and results are less accurate than the virtual environment. This is shown in terms of the order of magnitude difference for the real results (tens of millimetres) compared to the virtual results (millimetres). This disjuncture between the two environments and the data is predominantly a feature of two issues. Firstly, the real-world will contain a range of error sources (e.g., camera parameters, differences between cameras, precision and accuracy of camera location, lighting) which will not have been fully encoded into the virtual model. And secondly, the modelling within the virtual environment is dependent on the menu options provided and the quality of the model boundaries defined by the operator (e.g., light power and design of light sources, illumination wavelengths, object texture simulation, specular and diffuse reflections, camera definitions).

In the camera location and calibration experiments, the results from the virtual and real experimentation are found to be different for similar camera-object definitions. With reference to the data tables, the real world reprojection error results are between 0.43 mm and 14.0 mm, whilst the virtual results are between 0.03 mm and 4.8 mm. Furthermore, the results present differently as a function of object distance. As explained above, the definition and repeatability of the set up in both environments will be sources of error and performance differences for these experiments.

A persistent outcome from these two exemplars has been that (in the majority) the scenes created in Blender gave better results than the real scenes, noting that significant care and attention has been given to both the set-up of the real-world and virtual experiments. It seems that the differences between the real experience and its virtual reproduction are the main sources of error, as are the parameters used to generate the virtual world (such as the power of the light or the camera model).

Whilst differences are clear – overall trend behaviour of the virtual cameras is close enough to the real cameras to be able to conclude that the laws of optics are being respected. An example of this is illustrated by the geometric behaviour of the real and virtual systems incorrectly measuring sphere diameter when too close to the sphere (Figure 3.1 of Chapter 3).

However, whilst Blender has been shown to perform better than reality and shows significant promise as a tool to aid investigative multi-camera metrology systems, this performance capability is in itself a source of concern. The work demonstrates that it is inadvisable to use Blender in a raw or default environment configuration (such as light interactions, texture, camera characterisation) to design a digital twin of reality from an observable point of view, because it is lacking a "realism" that correctly captures the vagaries of the real-world environment.

In other science and engineering modelling scenarios this would be termed as correct/incorrect definition of modelling boundary conditions. Yet aspects of a real-world environment being measured by camera systems are at times exceedingly subjective as opposed to be easily quantified. And this also assumes that the modelling software provides menu options and variable control that have the capability to correctly mimic such real-world variables from a metrology sense, albeit they may model very well from a visual artistic sense. A case in point throughout these two exemplars has been the difficulties found with consistent and corresponding illumination of the two environments, and definitions of surface texture of objects. Surface textures are important because of light-object interactions. As shown in Figure 4.18, light has an impact on the detection of the sphere. It is further anticipated that the surface texture will also have an impact on the measurement and should be taken into account.

4.6. Conclusion

This chapter was initiated to begin to determine whether a 3D virtual reality modelling environment such as Blender can be used to fully model camera-based metrology systems, and potentially support the placement of camera-based measurement systems in industrial environments.

Based on this knowledge of Blender, two examples were used to explore and highlight the potential but also the problems encountered when considering using Blender for the simulation of camera systems: firstly, the measurement of a sphere using three cameras, and secondly, the measurement of a camera calibration artefact used to optimise multiple cameras positions (in this case eight cameras).

Through these experiments, it has been demonstrated that the current Blender environment needs to be more "realistic" whilst at the same time it is also recognised that such work needs to also minimise real-world experimental errors. In addition, the following 5M chart was designed to highlight the main sources of error and to give a basis for work for the next chapters.

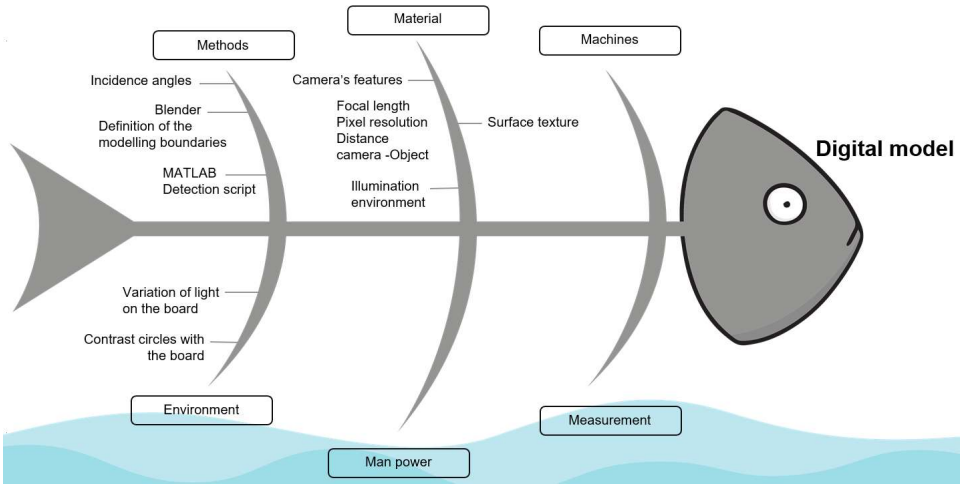


Figure 4.28: Fishbone of the main sources of errors identified in Chapter 4

In addition, the novelty of Chapter 4 can be identified as:

- Highlighting of the lack of methods to determine the optimum cameras and markers positions in a given scene, in literature.
- Characterisation of the response of Blender when used in its default environment, with limited knowledge of it.
- The identification of the main sources of error in the replication of the real environment in the 3D virtual environment (Figure 4.28).
- The metrological use of Blender as a tool and reference (incidence angle experiment, Section 4.5).

In Chapter 5, an artistic motion-based approach to improving the Blender environment is presented. This solution is based on characterising the real and virtual cameras, determining a correlation between the surface texture of real objects and the equivalent virtual objects, and understanding how to correctly reproduce the real lighting design (e.g., wavelength bandwidth, intensity, diffusivity and reflectivity).

Chapter 5

Photorealistic digital twin

Photorealism is an art movement created between the late 1960s and early 1970s in the United States, consisting of reproducing photographs in other media such as paintings or drawings as realistically as possible. This movement evolved from Pop Art, Abstract Expressionism and Minimalist art movements, and challenged the long-held notion that the use of photography in artwork was “cheating” [217].

Nowadays, this notion is most often used in the video game and the cinema industries, to generate “real” rendering for better game experiences with thrill [–218-220], and to create realistic characters and environments, based on computer generated imagery (CGI) [162] such as Disney and Marvel movies [221]. Technologies developed and tested by these industries have become more and more realistic and powerful over time [222-223], and have started spreading in other domains [–224-226].

The marriage between art and technology has made it possible to improve different techniques and to reach new goals. The combination of deep learning and augmented reality with cinematic rendering has helped (for instance) clinicians to interpret medical images for a growing variety of applications [168, 227].

Industry V4.0 has also been influenced by this art movement. Photorealism has been implemented in digital twins to meet different requirements such as in continuous monitoring procedures to detect the level of deterioration of architectural structures [228], or modelling realistic scenes and pictures for geolocation problems [229], connected and automated vehicle [230], and networks training [231].

However multi-camera network digital twins do not use photorealism technologies for their designs. Generally based on creating a whole virtual camera from its sensor to its radiometric characteristics [232-233], they are time-power consuming and not necessarily fully accurate.

In Chapter 4, an alternative to these designs was presented. This digital model was fully developed within Blender 2.9.2, and its performance for single and multi-camera systems in reproducing real scenes was tested in two experiments. It was then shown that (initially) Blender was much better than reality, but also that it was not possible to use it in a raw or default environment (such as no light interactions, no texture, no camera characterisation) to design a digital twin of reality from an observable point of view.

The shortcoming of this model was that it was not photorealistic enough. Blender’s internal rendering engine is based on ray-tracing [234], allowing the modelling of light sources and the interaction of the light with 3D objects in a visually realistic way. However, the conclusion was that it was necessary to characterise the real environment and the camera used.

In this chapter, the notions of photorealism and image fidelity/quality are introduced in section 5.1. These two notions are at the heart of the analysis presented in this chapter because of the need to degrade the perfect Blende environment in order to model the real environment, and to compare the results obtained by the two cameras in order to ensure a metrological comparison.

The method of designing a photorealistic virtual model is also presented as well as the method used to reach that result. Designing a realistic virtual model is not an easy task. Key photorealistic parameters have to be defined. In this work, photorealistic parameters are defined as the elements of the scene such as the illumination or the camera that need to be characterised in the virtual and real environment, in order to obtain a virtual rendering that feels real. Four main parameters are explored: the camera model, the illumination environment, the environment, and the object-light interaction. Their characterisation is presented in Section 5.2, and the new model generated by this process in Section 5.3.

However, the characterisation of the camera cannot be supported by changing only the values of its components (discussed in Chapter 4, Section 4.2). The camera signature and the generated image must be taken into consideration. Section 5.3 is devoted to image evaluation supported by additional literature on this topic, as well as methods used to compare the real and virtual camera signatures. In addition, the accuracy of the rendering process is considered by comparing the quality of the virtual and real images.

In this Chapter, the following challenges are expected to be addressed:

- What are the photorealism parameters needed to design a photorealistic Blender virtual environment?
- How to model a photorealistic illumination environment knowing that Blender used ray tracing to simulate light path?
- Which parameters real camera parameters are needed to design a photorealistic virtual camera?
- What is the impact of the environment and the object textures on the object under measurement?
- Is it possible to design a virtual model exactly identical to the real reference?
- Which metrology tool can be used to quantify “objectively” the difference between the real model and its digital representation?

These questions lead to the following potential innovations:

- Identify the key photorealistic parameters allowing to design a Blender photorealistic virtual environment and characterised their impact on the sphere deviation measurement.
- Designing a photorealistic blender virtual environment.
- Developing a metrology toolbox to quantify “objectively” the difference between the real model and its digital twin based on image fidelity and quality.
- Developing a set of recommendation to design a photorealistic virtual environment based on the work presented in this Chapter and in the previous Chapter.

5.1. Photogrammetry and Image fidelity/quality

5.1.1. Photogrammetry

Photorealism was born in the United States during the late 20th century. Up to that point, other styles existed such as the abstract expressionism which was a dominant art style. This art movement was been developed between 1940s and 1950s by Pollock, Rothko and De Kooning. The term abstract expressionism defined new forms of abstract art, characterised by gestural brush-strokes or mark-marking, and the impression of spontaneity [235]. The emergence of photorealism was motivated by the quest for novelty, and by the desire to demonstrate that it was possible to make photography not only with a camera [236].

The first working camera taking pictures appeared in 1827, created by Niépce, a French inventor. The picture was made of a piece of paper coated with silver chloride, which became darker where it was exposed to light 237]. After this discovery, and over the time, inventors have come and gone to improve the process and the camera.

However, this discovery did not only revolutionize the image world but also the art one. Before the invention of photography, all landscapes and portraits were realised on canvas. With photography, the commission was almost immediately realised, with a degree of precision and detail that had almost never been achieved. From this starting point, a small "war" started to determine if photography could be qualified as an art, and if photography could be exposed as a piece of art. At the same time, art was challenged by of the precision and details in photography and tried to reinvent itself to achieve or exceed photography.

It is in this context, at the end of the 20th century, that photorealism, was created. Challenging previous hierarchies, and prejudices, photorealism is, by itself, a revolution in the art domain, the reconciliation between the tradition art, and the photography, and a diversion of photography to create a type of art.

Photorealism, also known under the name of superrealism, is the duplication of visual information captured by a camera, on another medium such as a painting or a drawing. Usually, artists project photographs onto canvas to reproduce with precision and accuracy the shape of the photography.

The name of this art movement was given in 1969 by Meisel an American author, art dealer and proponent of the photorealist art movement. Two years afterward, he gave a clear definition of what photorealistic art was at the request of Speiser, who wanted to commission a large collection of photorealist art. This collection is known as 'Photo-Realism 1973: The Stuart M. Speiser Collection'. Photorealistic art is defined by [238]:

1. The Photo-Realist uses the camera and photograph to gather information.
2. The Photo-Realist uses a mechanical or semi-mechanical means to transfer the information to the canvas.
3. The Photo-Realist must have the technical ability to make the finished work appear photographic.
4. The artist must have exhibited work as a Photo-Realist by 1972 to be considered one of the central Photo-Realists.
5. The artist must have devoted at least five years to the development and exhibition of Photo-Realist work.

Various artists are now associated with this art movement, such as Picasso with cubism, or Monet with pointillism. Pioneers of this art form, they found ways to capture forms and landscapes in brushwork or sculpture, to deceive observers while forcing them to reflect on notions of realism and reality [236, 217].

Through their work, they have developed techniques and knowledge that make their art photorealistic, and pave the way for the development of other technologies and processes such as virtual reality. To understand the impact of this art movement, and its heritage, the key photorealism artists are listed below, with a short explanation of their work, and techniques:

- Close, who is the first photorealistic artist. He created monumental portraits through exacting realism that utilize scale, colour, and form [239].

- In photorealistic sculpture, Hanson made a life-size woman seated at a cafeteria table, plainly dressed, with her bags and packages by her side. All the objects dressing the sculpture, going from the dress to the belongs are real objects. This art shares a strong message, going against the grain of artists beautifying the female form [240].

- Estes used New York City streets, showcasing an urban aesthetic as models, and composed his paintings on multiple photographs. Due to this process, Estes found a way to reproduce, with a high a degree of reality, the reflection of surfaces. For instance, in his art works, the reflections of storefronts and car windows are reproduced [239].

- Bell did similar work to Estes, but his art was focused on small and insignificant objects, in order to question our culture values, as well as to play with the interplay of high and low culture. One of his most famous and remarkable pieces of art is “Sixteen candles” painted in 1992. This painting represents sixteen glass spheres, put on a glass surface. On this master piece, the reflection of the light on and through each sphere, as well as, their shadows have been represented [240].

Nowadays, the term “photorealism” is most often used in the video game and the cinema industries, to generate “real” rendering for better game experiences with thrill [218 - 220], and to create realistic characters and environments, based on computer generated imagery (CGI) [241] such as Disney and Marvel movies [221]. Technologies developed and tested by these industries became more and more realistic and powerful over the time [222-223], and started spreading in other domains [–224-226].

Photorealism is being explored by industries to solve challenges created by, for instance, the development of the smart factory. 3D animation software is used to set up fluid simulation, such as simulating the interaction of two fluids (air and water), using high-order finite difference discretization methods, designed for physics applications [227], or to train neural networks [242] for diverse purposes such as object detection [243], or cybersecurity [244].

The Industry 4.0 paradigm is also touched by this art movement. Photorealism is implemented in digital twin to meet different requirements such as in continuous monitoring procedures to detect the level of deterioration of architectural structures [228], or modelling realistic scenes and pictures for geolocalisation problems [229], connected and automated vehicle [230], and network training [231].

To develop a digital twin in a 3D animation software, a virtual model reproducing realistically the real object of interest is needed. The “photorealistic process” is composed of:

- **A characterization of the camera:**
  Modifications of the values of the focal length, the width and the height of the camera sensor, the focal distance, the F-stop (ratio of the system's focal length to the diameter of the entrance pupil), and to choose the right type of camera and sensor fit in the camera parameters. Noise and the distortion factors need to be added after rendering, in the post-processing step.
- **The environment characterization:**
  Determination of the type of light (colour, size and light intensity, the light-object interactions, the size of the object and the textures).
- **The render properties:**
  Chosen according to light-object interactions needed to generate the right amount of samples used for the rendering, the number of bounces wanted for each texture (such as glossy, glass), and denoise filters to remove dead pixels (burnt out pixels coming from the light-shadow rendering) in the final rendering.

In Blender, the notion of a sample is related to the way images are generated, in this case with the ray tracing method (explained in Appendix 5). With this method, light paths are projected from the camera to the outside, and the samples will follow the ray along a possible path of deviations to record the information generated (such as object, texture, colour) by the ray on the scene. Then the sample can be considered as noise in Blender. The more samples in the scene, the less noise in the final rendered image [245-246].

All of these guidelines allow the design of a virtual model which can be considered as a possible basis for a digital twin, resulting in rendered images similar to photographs. However, the rendered image is only similar to a picture taken by a camera. The quality assessment of the digital twin output must be done, by quantifying the level of similarities between a photograph, and the equivalent rendered image. This process is called image fidelity.

Several solutions exist from different fields to determine fidelity. The output of the model can be evaluated from an artistic point of view, using the definition given by Meisel [238]. However, basing this criterion solely on human perception is not sufficient. As explained earlier, digital twins can be used in dangerous and delicate situations. From a mathematical point of view, various possibilities exist such as the Discrete Fourier Transform, the degree of blur or sharpness of the image, and the signal-to-noise ratio. In the field of machine learning, image quality assessment algorithms are used for this purpose.

5.1.2.Image fidelity/ quality

Image fidelity and image quality can be mistaken. Image quality is defined as *"the weighted combination of all of the visually significant attributes of an image"* [247-248]; whereas image fidelity is the level of similarities between two images. In this research the similarities between a CMOS image and a rendered image are quantified.

From an artist point of view, the definition given by Meisel [238] is enough to assess the fidelity of the two images, because, by definition, a photorealistic work is strongly similar to a photograph. If the rendered image from a 3D animation software is considered as a photorealistic work, three of the five points, identified in 5.1.1, are then validated:

- The Photorealist uses the camera and photograph to gather information.
- The Photorealist uses a mechanical or semi-mechanical means to transfer information to the canvas.
- The Photorealist must have the technical ability to give the finished work a photographic appearance.

Noted that the second point can be interpreted differently depending on the current use of technology. Nowadays, people use different technologies such as computers, tablets, to do 3D modelling. Regarding this evolution, point 2 can be rephrased as follow: The Photorealist uses a mechanical, semi-mechanical or digital means to transfer information to the canvas.

The two other points are about the artist itself and are irrelevant in the image fidelity study. However, this work is also part of the design of a virtual model based of a digital twin. Due to the diverse applications of the digital twin, the justification from an artistic point of view is not enough.

The comparison of image similarities can be done through image quality and fidelity processes. For instance, the signature of the camera in an image can be analysed and used to determine if the real photograph, and the virtual photograph are taken with the same camera. In 3D animation software, it is possible to change the camera, and to modify the camera parameters such as the focal length or the size of the aperture. If the virtual image is similar, or identical to the real photograph, the imprint of both cameras must be similar or the same. This camera imprint can be visualised with the Discrete Fourier Transform techniques as illustrated in [249-250].

Nevertheless, the characterisation of the image does not stop there. An image is also defined through its degree of sharpness or blur, depending on the preferences of the user, as well as signal-to-noise ratio generated by all the elements in the photography process. These two components (blur and signal-to-noise ratio) provide key elements for constructing comparison matrices.

Blurring in an image is a natural process and is the opposite of sharpness in an image. Blur defines all elements outside the camera's field of view, where details are less sharp [251]. It also happens when the camera is moved when it was about to take a picture. This phenomenon is quite interesting because it corresponds to a high-pass filter that smooths out the details [251]. Conversely, sharpness defines all elements within the camera's field of view, where details are perceived. This phenomenon can be considered as a low-pass filter [252].

In many engineering applications, signal-to-noise ratio is often defined as the ratio of signal power to noise power. However, in some cases, such as imaging, an alternative definition can be found where

signal-to-noise ratio (SNR in Equation 71) is given as the ratio of the mean of the signal to the standard deviation of the noise in the image [253]. Its expressing is defined in Equation 71:

$$SNR = 10 * \log\left(\frac{M_{signal}}{\sigma_{noise}}\right) \quad \text{Equation 71}$$

Where M_{signal} corresponds to the mean of the signal. In that constant, σ_{noise} correspond to the standard deviation of the noise in the image.

Finally, the quality of both the images (virtual and real) can be individually assessed with Image Quality Assessment algorithms [254]. These algorithms take an arbitrary image as input and output a quality score. This score is the visualisation of the degree of degradation of the image. These degradations are caused by the presence of, for instance, noise, blocking artifacts, blurring, fading. They are introduced during image acquisition, compression, storage, transmission, decompression, display or even printing [255].

Image Quality Assessment algorithms are an automatic way to evaluate the image quality through a computing process, and may be more reliable than the human eye or other parameters (blurriness, the sharpness of the image, or the brightness), and is divided into three categories [256-257]:

- Full-reference Image Quality Assessment algorithms:
  A perfect reference image is used as a reference and compared with the test image.
- Reduced reference Image Quality Assessment algorithms:
  As the reference image is not available, an image containing partial information from the reference image is used for comparison with the evaluated image.
- Objective Blind or no-reference Image Quality Assessment algorithms:
  The only input is the image requiring quality assessment.

Various algorithms exist for each category with different degrees of complexity. For instance, MATLAB propose image quality metrics [258]:

- Full-reference Image Quality Assessment algorithms based on:
  - Mean-squared error.
  - Peak signal-to-noise ratio.
  - Structural similarity.
  - Multi-scale structural similarity.

- No-reference Image Quality Assessment algorithms using:
  - Blind/Referenceless Image Spatial Quality Evaluator.
  - Natural Image Quality Evaluator.
  - Perception based Image Quality Evaluator.

5.2. Photorealistic parameters characterisation

The results of Chapter 4 experiments have shown that (in the immediate sense) the real-word environment is less accurate than the virtual one. These differences between the two setups appear to come for two sources of errors:

- The lack of “realism” of the virtual model where parameters such as camera parameters, differences between cameras signature, precision and accuracy of camera location, and lighting are not fully taken into consideration, and not modelled.

- The constraints of modelling defined by the software itself with its menu options and the user itself with the identification and delimitation of the model boundaries (e.g., light power and design of light sources, light wavelengths, object texture simulation, specular and diffuse reflections, camera definitions).

Adding "realism" to the Blender scene, considered "perfect", means deliberately degrading the scene to better mimic the real world. This degradation process can be achieved using a photorealistic approach. The environment configuration has to be modified by the definition of the model boundaries, to generate an output whose results are of the same order of magnitude of the real output.

5.2.1. Key elements for each photorealistic parameter

The characterisation of the four key parameters identified were completed according to the following processes. Noted that in the way Blender has been designed, light and environment are separated and no overlaping eixst.

1. **Environment characterisation:**
- Identifying the key objects defining the experimental environment.
- Identifying object-object interactions.
- Measuring the size and characteristics of the objects.

2. **Light characterisation:**
- Putting the scene in a virtual volume cube with appropriate size that mimics the real world room (defined in the environment characterisation) to generate ray bounces.
- Measuring and calculating the illumination light power.
- Measuring the light dimensions.
  Identifying the light-object interactions for the purpose of performing object interaction characterisation on them.

It is noted here that the light characterisation and object-light interactions characterisation was done separately due to ambiguity existing between the notion of surface texture in the real world and in Blender highlighted in Section 4.3.

However, as explained in Section 4.3, the light power in Blender is defined depending on the light source, rather than with reference to the electrical power. Due to this difference, in the real world, a light meter (Sekonic C-800-U) was used to measure the radiant flux of the light in lumens.

The conversion of luminous power was done by combining the relashionships 2 and 3 between watts and lumens, given in Section 4.3.1. The result obtained was then multiplied by the luminous area to obtain the correct conversion from lumens to watts, expressed in Equation 72.

$$1\ (W) = 683\ (lumen/m^2) * Light_{surface} \qquad \text{Equation 72}$$

3. Object-light interactions characterisation:

The process of measuring all surface textures of the real objects is presented in Appendix 4. The surface texture values were measured by calculated R_a, the average value of the roughness profile over 3 repetitions and using a Gaussian filter (details about R_a, and the cutt off are given in Appendix 4). The value of R_a for each object used in this chapter is :

- Large zone circles board:
  Black circles : 2.71 μm
  White circles : 1.47 μm

- Small zone circles board:
  Black circles: 2.90 μm
  White circles: 2.78 μm

- Large checkerboard:
  Black squares : 3.29 μm
  White squares : 0.26 μm

- Small checkerboard:
  Black squares : 3.48 μm
  White squares : 0.30 μm

- 0.09 m sphere: 10.68 μm

4. Camera characterisation:

The characterisation of the virtual camera was based on the datasheet of the real camera, and the parameters available in Blender:

- F-stop.
- Sensor size.
- Image resolution.
- Pixel size.
- Depth of focus (distance at which the object starts to be out of focus).
- Shape of the aperture with the number of aperture blades.
- Level of noise and distortion.

The characterisation of each parameter was achieved by measuring their impacts on the system output. The 0.09 m sphere diameter from Chapter 4 was again used as the artefact for the photorealism experimentations.

5.2.2.Methodology

5.2.2.1. Camera models

Two virtual camera models are used in this Chapter.

The basic model is the camera model used in Chapter 4, and was defined as:

- Lens:
  Type: Perspective.
  Focal length: 3.04 mm.
  Lens Unit: Millimetres.

- Depth of Field:
  Nothing was set up and/or modified.

- Aperture:
  Nothing was set up and/or modified.

- Camera:
  Sensor Fit: Horizontal.
  Sensor width: 3.68 mm.
  Sensor Height: 2.76 mm.

- Post Processing:
  No noise and distortion applied.

The final model was the camera model designed in its chapter after characterisation, and was defined as:

- Lens:
  Type: Perspective.
  Focal length: 3.04 mm.
  Lens Unit: Millimetres.

- Depth of Field:
  Distance: 0.3 m.
- Aperture:
  F-stop: 2.
  Blades: 3.

Blades rotation: 0°.

- Camera:
  Sensor Fit: Horizontal.
  Sensor width: 3.68 mm.
  Sensor Height: 2.76 mm.

- Post Processing:
  Noise level: 0.02 dB
  Distortion level: 0.001

For both cameras model, the pixels were square with a size of 1.12 μm, and the image resolution was 1,640 x 1,232 pixels.

5.2.2.2. Definition of the virtual scenes

Two virtual models were used for the characterisation of the photorealistic parameters. The first virtual model was the model used in Chapter 4, referred as the basic model. This model is illustrated in Figure 5.1 and was composed of a 0.09 m diameter white diffuse sphere modelled with 32 polygons on a black background. Two square area lights with default values fixed at 10 kW were placed at 1 m above and below the sphere to illuminate the whole sphere homogeneously. Three pinhole cameras were set up on the x, y, and z axes to view the sphere from different points of view. The Blender modelling environment was set up to be perfect (parameters such as no noise, distortion, surface texture were introduced).

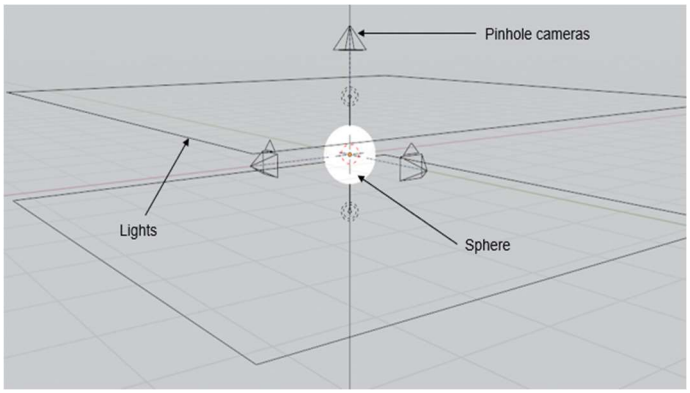


Figure 5.1: Basic virtual model

The second virtual model, referred as the final model, is illustrated in Figure 5.2, and explained later in Section 5.2.2.5.

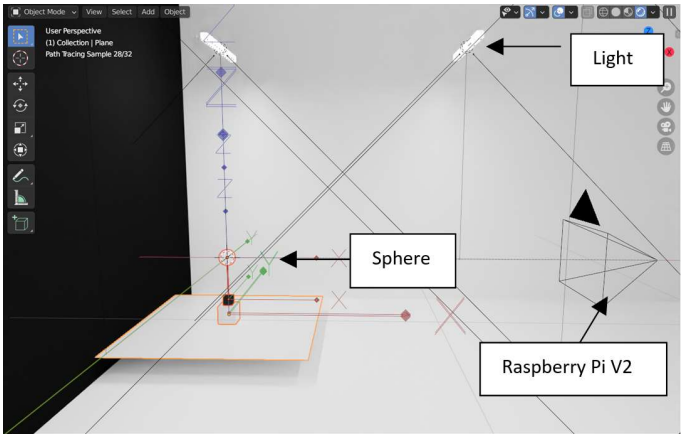


Figure 5.2: Final virtual model

The main differences between the two models were the introduction of surface textures, the correct modelling of the real light in Blender with equivalent electrical intensity power, dimensions, and position, as well as the modelling of objects belonging to the sphere environment, e.g., they are not used in the experiment, but they can impact the measurement by bringing noise and errors.

5.2.2.3. Characterisation methodology

For each parameter, the deviation of the sphere diameter was measured to determine the impact. The detection of the sphere diameter in the image, and the conversion of its diameter from pixels to millimetres was achieved using the MATLAB toolbox presented in Section 3.3.1, of Chapter 3.

Environment

The impact of the environment was measured by removing elements from the final virtual model (Figure 59) to reconstruct the basic virtual model (Figure 58).The virtual camera model used was the final camera model defined in Section 5.2.2.1, and was placed at 0.5 m from the sphere.

Light

The impact of the light was assessed in the new set up by changing the light intensity, with the basic virtual camera model. The light power was set to 5 W, 10 W, 15 W, 20 W and 39 W (actual light power measured with relationship 4 of Table 4.3). Light output greater than 1 kW was too high due to low contrast between the black background and the sphere; and light output lower than 5 W was too low causing a similar problem. The 39 W was chosen as the biggest value of the set because it was the measured light power of the real lights.

To reinforce this affirmation, Figure 5.3 shows two detections examples for an illimination light power at 1 kW and 2 W, at a camera-sphere distance of 0.55 m, 1.36 m and 2.36 m. Note that these images were taken with the basic virtual camera model.

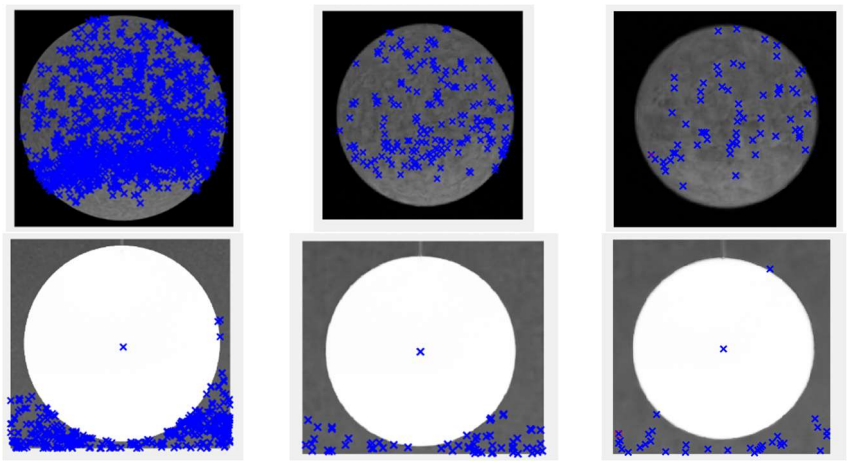


Figure 5.3: First row: illumination power at 2 W; Second row: illumination power at 1 kW.

In Figure 5.3, the first row corresponds to the sphere illuminated at 2 W, while the second row corresponds to the same sphere but illuminated at 1 kW. The first observation is that the spheres in the first row are darker than the spheres in the second row, due to the difference of a power of 10 between the illuminances. The second observation comes from the distribution of the blue points. These points correspond to the detection of the centre of the sphere. Due to the illumination used, the sphere is either too dark or too light to be detected correctly. This leads to the detection of micro-circles in the sphere and in the background due to poor contraction between the sphere alone and between the sphere and the background.

Noted that similar work was completed in Chapter 4 but with the basic camera model and basic virtual scene. The results of this current experiment were compared to the results of the previous experiment in Chapter 4 to determine the impact of the modification of the camera and scene models on the sphere deviation.

Object-light interactions

The impact of object-light interactions was measured by applying different textures on the sphere, and measuring the impact of the sphere diameter. The textures used were defined in Blender as:

- No texture (default object).
- Glossy with a roughness of 0.5.
- Glossy with a roughness of 1.0.
- Glass with a roughness of 0.5.
- Glass with a roughness of 1.0.
- Emission with a roughness of 0.5.
- Emission with a roughness of 1.0.
- Normal map.
- Height map.
- Colour map.

As explained in Chapter 4, Section 4.3.3, the principal Blender tool to generate a texture, which

specifies all the object characteristics (such as roughness, diffuse, reflection) is the *principled BSDF* shader. This is shown in Figure 5.4.

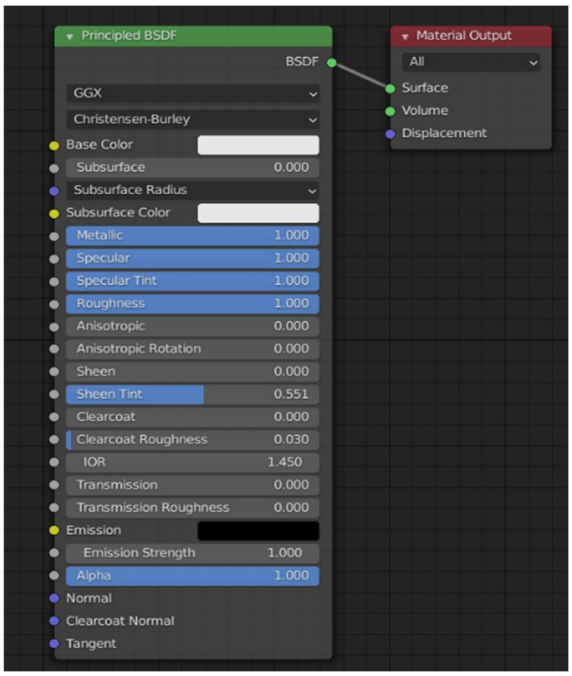


Figure 5.4: Principled BSDF shader in Blender

However, this tool does not necessarily model the physics correctly due to the developers' choice to target an artistic market. The options for the principled BSDF defining the object surface characterisations are controlled by an index between 0 and 1; except for the refractive index of the surface (IOR in Figure 5.4), which can be set to the real-world value.

This index is called roughness and its principal was developed by HanrahanKrueger [105]. In this research, a model is presented for subsurface scattering in layered surfaces in terms of one-dimensional linear transport theory. In addition, the modelling process is based on how the light is reflected on a rough surface.

The use of this “roughness” index affects the interaction between the object and light, as well as the camera's perception of the object and the image rendering process, making the direct correlation between the real-world surface definition, and the equivalent environment in the Blender model difficult. Furthermore, this value is totally dependent on the designer’s perception of the subject, and hence is very subjective.

Camera

The impact of modifying the virtual camera model was measured, in the new virtual model, by observing the fluctuation of the sphere deviation as a function of adding complexity to the camera parameters. This complexity consisted of setting up successively the “true” numerical value, to the basic model, of the F-stop, the depth of focus, the noise level, the lens distortion level, and the aperture shape (blades). In addition to correlating the geometry output from the camera

measurement systems, characterisation was also achieved by a process of image fidelity and assessment of blur for each variation of camera parameter, presented in Section 5.3.

5.2.2.4. Real environment

Environment definition

The first real-world experiment was the sphere diameter measurement introduced in Section 3.4.1 of Chapter 3. The measurements were repeated three times out of sequence in order to understand the potential variance as a function of camera placement.

The experimentation took place in a controlled environment (parameters such as light intensity control, noise coming only from the cameras were known), with two rows of three 39 W LED diffuser units mounted on the ceiling, located on a grid basis 1.30 m from each other. The sphere was mounted on a 5 mm diameter black steel rod attached to two blocks (black and white respectively) located on a white table (~1 m^2) with a black background. The white table and cube were used to reflect incident light up to mitigate shadowing of the bottom of the sphere. A dark background was used to create high contrast between the white room and the white sphere.

The different elements of the practical set-up were judged to have the following Blender related texture descriptions:

- White polystyrene sphere: Specular.
- Background: Diffuse.
- White wall: Specular.
- Table: Specular.
- White cube: Specular.
- Black cube: Diffuse.

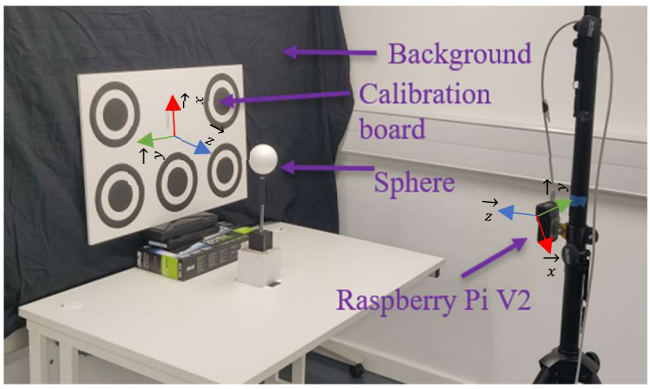


Figure 5.5: Real-world experimental set up

Calculation of the distance between the camera and the sphere, and the camera parameters

The distance between the camera and the sphere were measured by using the zone circle board presented in Chapter 3, Section 3.1.2. The zones circles were 0.1 m, 0.15 m, and 0.2 m respectively in diameter on a substrate measuring 0.5 m x 0.75 m x 0.025 m.

The board was placed at the location of the sphere, in the same environment, with its centre aligned with the camera optical centre. The camera used in the experiment took a picture of the board, and the MATLAB toolbox, presented in Section 3.3.1. of Chapter 3, was used to detect the centre of the circles, and calculate the camera extrinsic parameters. The true distance between the camera and the sphere was equal to the camera translation on the z-axis minus the radius of the sphere (because the zone circle board was rotated 90 degrees on the x-axis to face the camera).

The focal length and the lens distortion of the camera were measured with the MATLAB toolbox presented in Section 3.3.3, using the black and white checkerboard pattern with 30 mm squares introduced in Section 3.1.2. The noise level in the real image was used to define the noise level in the virtual environment. This choice was made because, as explained in Section 4.2.1, noise in Blender is added after the rendering process, in the post-rendering process, by directly modifying the image.

The noise level was calculated with the signal-to-noise ratio, using the MATLAB toolbox developed in Section 3.3.6.

5.2.2.5. Blender environment

The real-world experimentation was recreated in the virtual environment, generated in Blender using a Python script as shown in Figure 5.2, in Section 5.2.2.2. The same MATLAB functions and data processing developed for the real-world scenario were also used to process the virtual reality generated data.

The Blender model was composed of the same elements as the real environment. The white glossy sphere (0.09 m diameter) was modelled with 32 polygons on a black background. The lighting units mimicked the real-world luminaires. Two lights were created for each “real” light, and oriented at + 45 and – 45 degrees with respect to the z-axis to recreate the light spreading in the real environment. However, it is noted that light power is defined in Blender depending on the light source, thus making it challenging to directly correlate the light intensity between with the real-world equivalent texture in Blender environment.

5.2.2.6. Results

Real-world analysis

The results from the real single camera measurement of the 0.09 m sphere are shown in Figure 5.6 demonstrating the deviation of the camera measured sphere diameter from the actual diameter based on three repeats, and further illustrated in Table 5.1. The x-axis scale represents the object

distance, and the y-axis scale is the difference between the theoretical and measured sphere diameter.

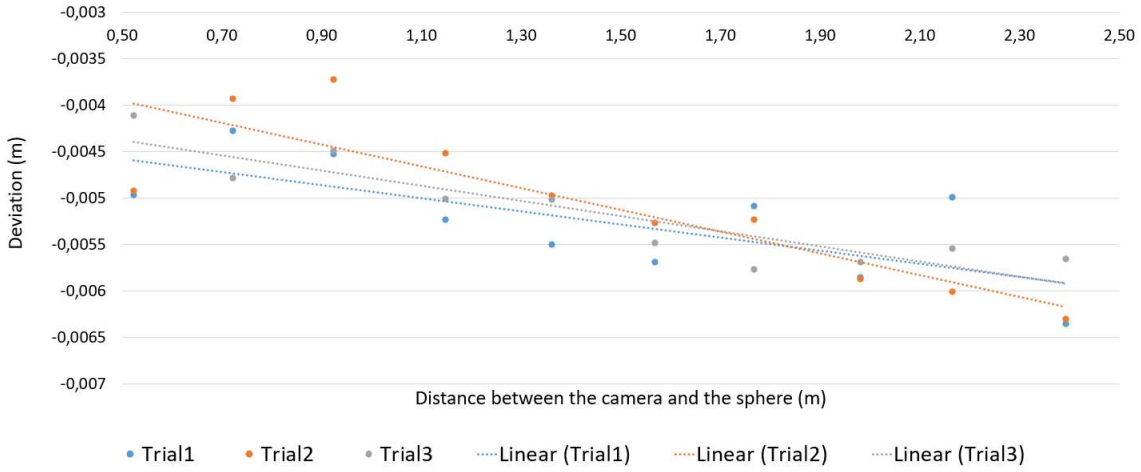


Figure 5.6: Deviation from theoretical observable sphere diameter

In Figure 5.6, the dotted lines correspond to the trend line for each trial and are used for the data analysis plotted in this Figure. They were drawn in Excel, using a linear model. From these lines, the data shows that the three repeated trials generated similar results and trends. The (negative) deviation of the sphere diameter increased slightly with distance from the object. Noted that if the deviation was 0 mm, the theoretical diameter was measured without any error.

Table 5.1: Deviation on the sphere diameters for a distance between the camera and the sphere of 0.5 m,1.36 m and 2.17 m.

	0.5 m (mm)	**1.36 m (mm)**	**2.17 m (mm)**
Trial 1	-4.97	-5.50	-4.99
Trial 2	-4.25	-4.98	-5.54
Trial 3	-4.11	-5.00	-6.01

Table 5.1 reinforces what was observable in Figure 5.6. The deviation between the three trials decreases with increasing camera-sphere distance, with variation between each answer observable. This variation potentially comes from the movement of the camera, and the difficulty to perfectly align the camera optical centre with the sphere centre. The diameter deviation increase is linked to the resolution of the camera, because as the object distance increases the sphere is resolved using fewer pixels thus potentially reducing the accuracy of detecting the sphere in the image at longer distances.

Due to the similarities of the three real answers, the mean of value result was used as reference for the comparison with the virtual results. In addition, the standard deviation of the mean of the deviation on the sphere diameter was 0.58 mm.

Characterisation of the Blender model

Environment characterisation

Figure 5.7 shows the evolution of the sphere diameter measurement as a function of the reconstruction of the basic virtual model from the final virtual model, step by step, with the basic virtual model of the camera at 0.5 m from the sphere.

These steps of deconstruction were:

- Remove 4 lights out of 6 (2 lights kept).
- Remove the stick on which the ball was supported.
- Remove one of the cubes used as a support.
- Remove the two cubes used as a support.
- Remove the denoise.
- Remove the table.
- Locate at 1 m, the two lights above and under the sphere.
- Remove the colour of the background (default white background).
- Remove the cube volume used to simulate the room.
- Change the illumination light power from 39 W to 10 kW.

The x-axis of Figure 5.7 corresponds to the steps of deconstruction, illustrated by icons to show the starting and final points, and the element removed, and the y-axis is the sphere diameter in metres.

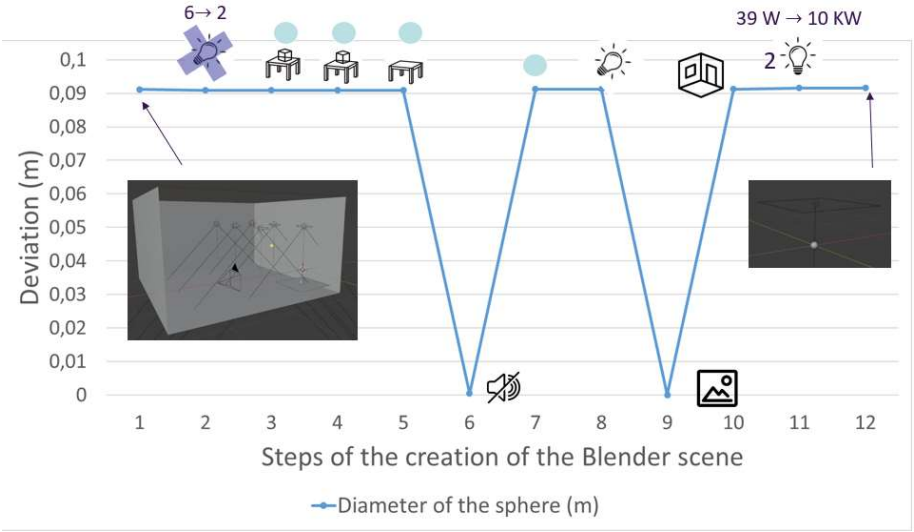


Figure 5.7: Impact of each environment element on the sphere diameter measurement

The results demonstrate that two elements impact the sphere detection. The first element is the contrast between the sphere and the background, illustrated in Figure 5.7 by icon, step 9 of the creation of the Blender scene. In Blender, the default colour object is white. Due to the utilisation of a white sphere in this experiment, using a white background makes the detection impossible (white sphere on a white background).

In addition, the necessity to have a high contrast between the sphere and the background has already been observed in Figure 5.3, with illumination light power at 2 W. In the three pictures, the sphere detection was altered by the grey colour of the colour map of the sphere, highlighting defects in the sphere's surface. Therefore, the contrast between the sphere and the background has to be set up carefully.

The second element is the usefulness of denoising, and the evolution of the complexity in light calculation of the scene with the utilisation of a cube to generate the room. Both elements are illustrated in Figure 5.7 by  and  icons, steps 6 and 10 of the creation of the Blender scene. Blender, in its scene menu, offers the ability to apply a denoising filter to the scene created before the rendering process. Due to the utilisation of Cycles as the render engine, the ray-tracing method is used. As explained in Appendix 5, rays are connected from the camera to the light source. Some of the light rays hit objects, and record information such as their colours or textures. This information is used afterwards to generate the final rendering image of the scene. Consequently, the more rays sent to the scene, the better the image accuracy will be.

In Blender, the number of rays sent into the scene is controlled by the parameter *samples*, in the rendering properties, in the properties menu (Appendix 1). If not, enough samples are used, not enough data will be collected, and white spots, "dead pixels" (also called fireflies in the Blender community), will appear in the image. The denoising filter will adjust the colours of each pixel, smooth out inconsistencies and remove dead pixels (represented by white dots on the picture) also known as noise [259].

Light characterisation

Figure 5.8 shows the evolution of the sphere deviation using the basic camera model. The x-axis scale represents the object distance in meters, and the y-axis scale is the difference between the deviation on the diameter observable in meters.

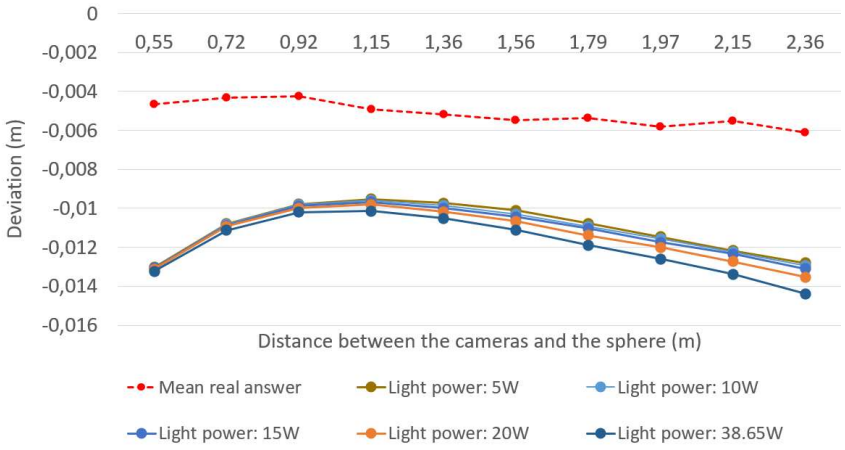


Figure 5.8: Light variations

In Figure 5.8, the red dotted line is the mean of the three real answers from Figure 5.6, used as a reference, and the solid data lines are the virtual responses for different light powers.

The real response appears to increase slightly with increasing camera-to-sphere distance; while the virtual responses behave similarly, with on initial decrease of the deviation of the diameter between 0.5 m and 1.15 m, followed by an increase between 1.15 m and 2.36 m.

A similar experiment was performed in Chapter 4, giving similar results (Figure 4.18) on the distances between 1 m and 2 m. It was concluded that the error was caused by the light power, exaggerating the details of the sphere, and the camera pixels resolution. In addition, as demonstrated in Chapter 4, Figure 4.15, the error between 0.55 m and 1.15 m came from the geometry of the sphere. This error was not observed in Figure 4.18 of Chapter 4 due to the use of a circle.

However, the comparison between the two results (Figure 4.18, Chapter 4; and Figure 5.8, Chapter 5) shows that the starting point is not the same, as well as the slope of the curves. In addition, the real and virtual answers in these two Figures have similar behaviour, but the virtual answers are shifted by approximately -10 mm from the real answer. These differences might come from the addition of realism in the virtual set up, and in the virtual camera model.

Regarding results presented in Figure 5.8, the light power does not seem to be a preponderant source of error. However, due to the utilisation of an optical system, and the definition of digital twin, the virtual light power was set up at the 39 W, which corresponds to the real light electrical power.

Object-light interactions characterisation

The impact of this parameter was measured using the methodology described in Section 5.2.2.4, using the final virtual scene (Figure 5.2), and the final camera model (described in Section 5.2.2.1). However, instead of changing the illumination power, the texture of the sphere was changed. The results are presented in Figure 5.9 and Figure 5.10.

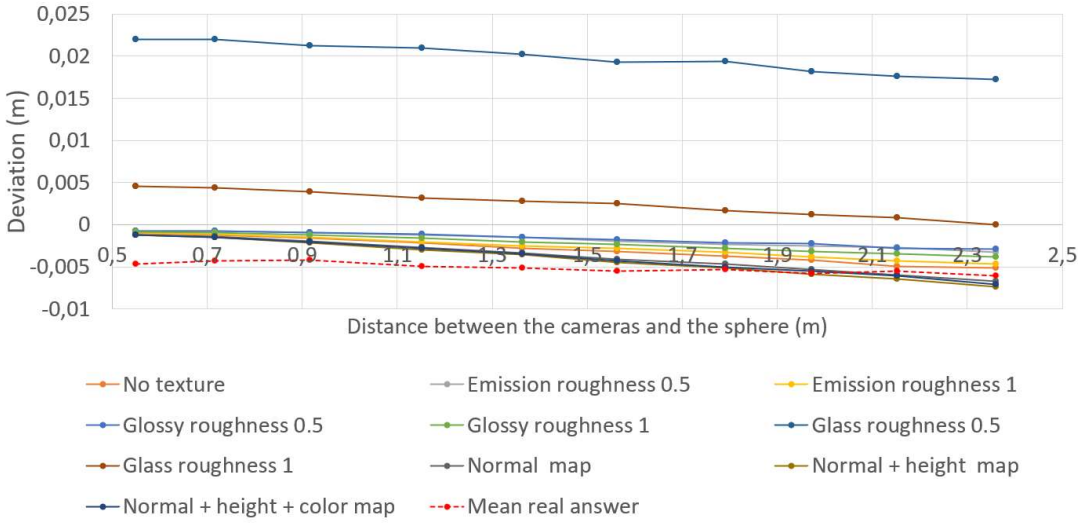


Figure 5.9: Texture variations

As in Figure 5.8, the red dotted line is the reference, and the other lines are the virtual responses for different sphere textures. The response of the glass texture with a roughness of 0.5 has been removed, between Figure 5.9 and Figure 5.10, due to the magnitude of the deviation, the other responses are unreadable due to scale compression.

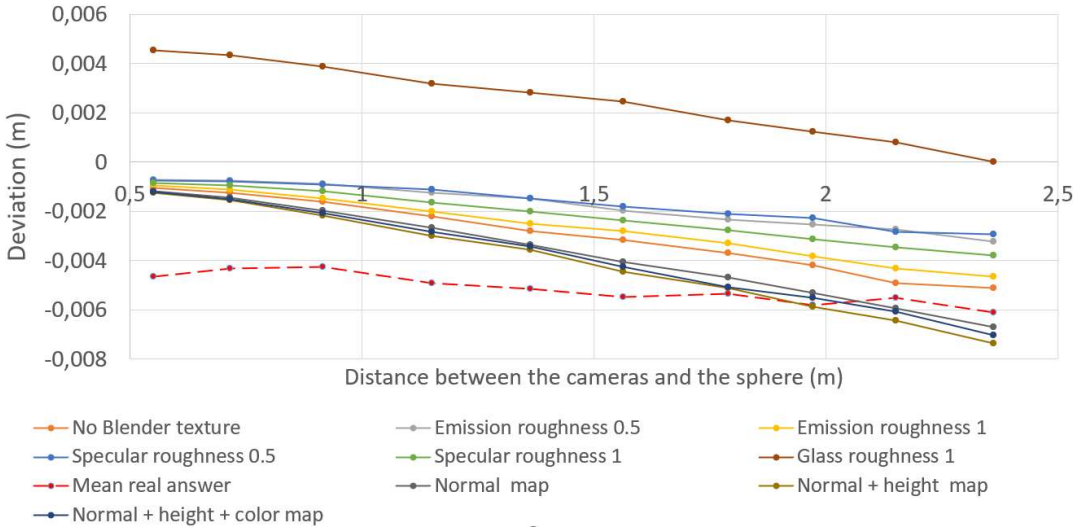


Figure 5.10: Texture variations without the answer for a glass texture with a roughness of 0.5

In the legend of Figure 5.9 and Figure 5.10, the specular, glass and emission texture are defined with a "roughness" index of 0.5 and 1. As explained in Section 4.3.3 of Chapter 4, this index defines the surface roughness, and varies between 0 (surface roughness at 0%) and 1 (surface roughness at 100%). For instance, applying a glass texture with a roughness of 0.5 to an object means that the object roughness is set up at 0.5, and the final surface texture rendering is perceived as half-mirror.

In the legend of these Figures, a "no Blender texture" plot is shown. This plot was generated by putting no texture generated in Blender on it (texture can be specular, glass, emission for instance).

Regarding Figure 5.10 and Table 5.2, all of the virtual answers behave similarly, e.g., the deviation on the sphere diameter increases as a function of increasing distance sphere-cameras. In addition, two groups are observable on both sides of the no Blender texture answer.

Table 5.2: Deviation on the sphere diameters for a distance sphere-camera between 0.5 m and 1.79 m for different sphere textures

Experiment	0.5 m (mm)	1.15 m (mm)	1.79 m (mm)
Mean real answer	-4.97	-5.24	-5.08
No Blender texture	-1.07	-2.20	-3.71
Emission roughness 0.5	-0.73	-1.24	-2.33
Emission roughness 1	-0.97	-2.03	-3.31
Specular roughness 0.5	-0.75	-1.13	-2.12
Specular roughness 1	-0.86	-1.64	-2.76
Glass roughness 1	4.53	3.18	1.70
Normal map	-1.19	-2.69	-4.69
Normal + height map	-1.27	-3.00	-5.14
Normal + height + colour map	-1.22	-2.83	-5.10

The first group is composed of the emission, glass, and specular surface textures. Their virtual answer increases, but the deviation is still lower than the real and the no Blender texture ones. The second group is composed of the results obtained by adding defaults to the sphere surface (with a normal map (micro imperfections), a height map (macro imperfections) and colour map.

As explained in Section 4.3.3 of Chapter 4, the term “normal map” defines a mapping technique used for faking the lighting of bumps and dents. The colour map is a set of values that are associated with colours, and the height map causes an effect where the actual geometric position of points over the textured surface are displaced.

Their answers gave:

- a better result than the reference between 0 m and 1.56 m.
- a similar result between 1.79 m and 1.97 m.
- a poorer result between 1.97 m and 2.36 m.

In addition, it is noticeable that the index of roughness has an impact on the measurement. For instance, at a distance camera-sphere of 0.5 m, an emission texture with a roughness index of 0.5, has a deviation of -0.73 mm, while the same texture with a roughness index of 1, has a deviation of -0.97 mm.

It seems that the use of maps (normal, height and colour) to generate the surface texture of the sphere give more realistic results than using surface texture mimicking of a specific light property such as diffusion. In addition, if the Blender no texture (default mode) is used as a reference for the virtual answers, it is observable that using a surface texture mimicking a specific light property (glass, emission or specular) improves the results.

This conclusion can be explained by the fact that the physics is biased due to commercial choice, but also by the fact that the surface texture imitating a specific light property does not affect the structure of the object. Thus, as explained in Section 4.3.3 of chapter 4, there is no objective rough value. Furthermore, as outlined in this section as well, the roughness value in Blender is determined by an index (between 0 and 1) that modifies the roughness of the texture to achieve the desired light properties. For example, a glass texture will have an impact on the light properties of the object to which it is applied, so that light will be reflected perpendicular to its angle of incidence.

A normal map will alter the surface of the object physically by itself. So, a glass texture merged with a normal map, will not have the same light properties as an object with just a glass texture on. However, as for the textures mimicking the light properties, the physics is biased, and it is not possible to use the real values because they are configured with images (png, jpg, ...) converted into maps, with indexes that the user modifies to obtain the desired visual result.

Camera characterisation

Figure 5.11 shows the evolution of the sphere deviation as a function of the complexity of the camera model. The first camera model, referred to as the base model in the legend to Figure 5.8 and Figure 5.10, consists of the basic elements (such as focal length and lens characteristics) needed to define a virtual camera and is presented in Section 5.2.2.1. The final model of the virtual camera (referred to as the aperture shape in the legend to Figure 5.8 and Figure 5.10) was generated from this base model. However, in order to understand how each new camera parameter (such as depth of field, noise and distortion) affected the camera model, the deviation on the sphere diameter was measured for each new parameter added to the previous camera model until the final camera model had been built.

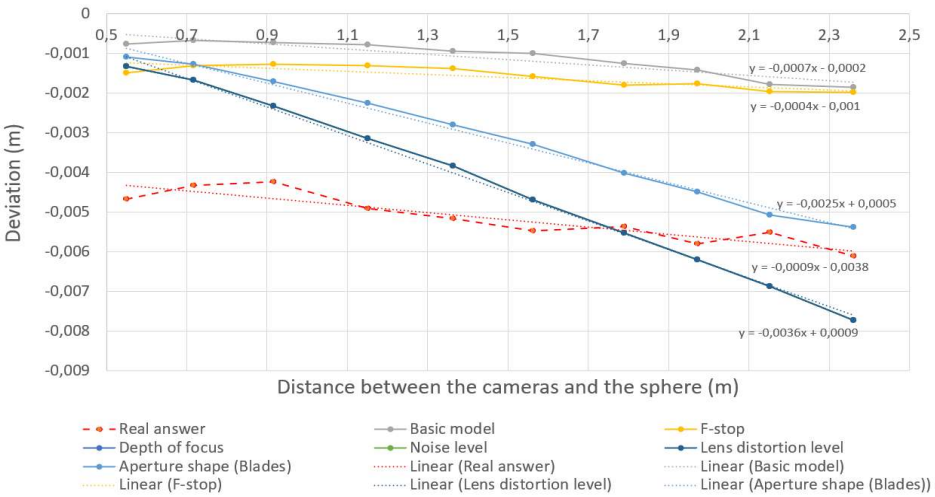


Figure 5.11: Camera parameters variation

As in Figure 5.8 and Figure 5.10, the red dotted line is used as the reference, and the other lines are the virtual responses corresponding to the modification of the basic camera model by setting the "true" numerical value of each camera component until the final virtual camera model is obtained.

From Figure 5.11, two behaviours can be observed. The first behaviour is described by the basic virtual model and the basic virtual model with the F-stop setting. These two virtual responses describe the

same trend as the real camera model but are shifted upwards by -2 mm and -1 mm respectively. These offsets can be explained by the composition of the camera model. The basic camera model is composed only of the focal length and the sensor size. The -2 mm difference between its answer, the real and the basic virtual ones seems to come from the incomplete definition of the camera model. In addition, this difference can also be interpreted as the basic difference between the real and virtual models before any photorealistic approach was considered.

The second behaviour is described by the fixed depth of focus, the level of lens distortion, the level of noise, and the shape of the aperture. The addition of depth of focus to the previous model generated an almost constantly increasing curve. Its slope is equal to -3.6 mm, while it is equal to -9 mm for the real response. Furthermore, it seems that adding lens distortion and noise level to this camera model (F-stop and depth of focus fixed) does not affect the result, as the three responses are superimposed. However, the level of lens distortion and noise level were quite low (0.02 dB and 0.0013 dB respectively), and high levels might impact the results.

The modification of the aperture shape (triangular) changes the slope and shifts the curve upwards. In the previous camera model (F-stop, depth of focus, lens distortion, and noise level), the slope is equal to -3.5 mm, while in the final camera model (F-stop, depth of focus, lens distortion, noise level, and aperture shape fixed) is equal to - 2.5 mm. This difference comes from the modification of the perfect pinhole model with a perfect circular opening, into a camera model with a triangular opening due to the number of the blades, defined by default at 3 in Blender.

From the analysis of Figure 5.11, it is noticeable that the basic virtual camera system and the virtual camera model with only the F-stop set up are given the worse answer. In the other hand, setting the F-stop, the depth of field, or the F-stop, the depth of field and the aperture shape have an impact on the sphere deviation. However, it seems that these parameters are not enough to generate a model identical or similar that the real one.

To complete the characterisation of the camera, a closer look was taken at the effect of the aperture on the measurement. To determine the effect of the aperture, the same experiment was performed as for the other camera modifications, but only with the number of blades changing from 3 (minimum in Blender) to 16 (maximum in Blender), with a step of 1 blade. The results are shown in Figure 5.12.

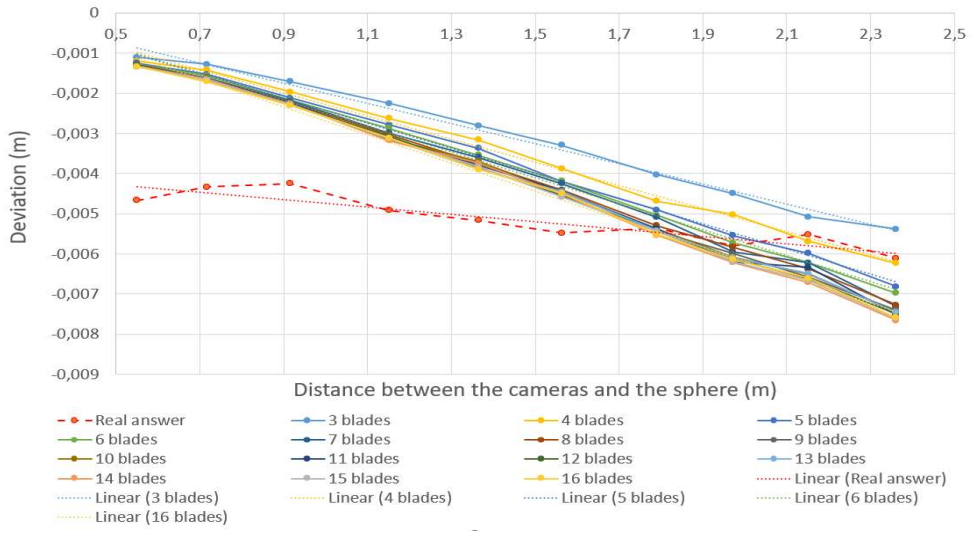


Figure 5.12: Blades variations

According to Figure 5.12, the change in the number of blades has an impact on the deviation. A strong change in the slope of the data trend appears for 3, 4, 5 and 16 blades. For example, at 2.3 m, the deviation is at 5.38 mm for 3 blades, at 6.23 mm for 4 blades, at 6.81 mm for 5 blades and at 7.59 mm for 16 blades. According to these results, the number of blades was set up at 4 for subsequent experiments.

Finally, the comparison between these two final virtual camera models (with a perfect circular and with a lozenge aperture shape), with the real camera model shows that:

- The virtual model with a perfect circular aperture shape gave a better result than the real one over the distance range 0.5 m and 1.7 m, but a worse result after 1.79 m.
- The virtual model with a lozenge aperture shape gave a better result than the real one over the all-distance range.
- According to Table 5.3, the virtual answers are better than the real one when the sphere is closer to the camera.
- According to Table 5.3, the virtual results are closer to the real ones, when the sphere is far away from the camera.

Table 5.3: Comparisons between the final camera model with a perfect circular aperture and a lozenge aperture with the real answer for an object distance of 0.5 m, 1.36 m, 1.56 m, 1.97, and 2.15 m

Experiment	0.5 m (mm)	1.36 m (mm)	1.56 m (mm)	1.97 m (mm)	2.15 m (mm)
Mean real answer	-4.67	-5.17	-5.48	-5.80	-5.52
Circular aperture	-1.33	-3.83	-4.69	-6.20	-6.87
Lozenge aperture	-1.19	-3.15	-3.87	-5.02	-5.08

The differences between the virtual and the real camera model come from the parameters used as well as their numerical values. However, the camera characterisation presented in this Section, show the impact of these different parameters on the sphere detection. The output does not take into consideration the image generation.

The aim of the next section (Section 5.3) is to give a deeper camera characterisation by using an image fidelity process. This process is based on:

- The comparison of the Discreet Fourier Transform of the different camera models (real and virtual) used in the experiment presented in Figure 5.11.
- The comparison of the blur percentage in the image generated by the real camera model, and the final virtual camera model based on a reduced reference image quality assessment algorithm.

5.3. Image assessment

Image assessment is usually achieved by looking at the image fidelity and image quality, which are often confused due to their similarity. Image quality refers to the level of accuracy with which different imaging systems capture, process, store, compress, transmit and display the signals that form an image [260-261], whereas Image fidelity describes the ability of a process to render an image accurately [262-263], without visible distortion or loss of information. The difference between the two can be summarised as follows: one focuses on the characteristics of signal processing in different imaging systems, while the other focuses on the perceptual evaluations that make an image pleasing to human observers.

In this study, the interest in using fidelity and image quality methods was to determine tools that could quantify the level of similarity of the virtual camera to the real camera through the image, using the camera signature; and also, to quantify the accuracy of the rendering process through the comparison of two image qualities.

5.3.1. Methods definition

From an artistic point a review, and in a photorealistic context, image assessment is characterised by a rendered image similar to a photograph.

An artistic work is considered photorealistic when [238]:

1. The Photo-Realist uses the camera and photograph to gather information.
2. The Photo-Realist uses a mechanical or semi-mechanical means to transfer the information to the canvas.
3. The Photo-Realist must have the technical ability to make the finished work appear photographic.
4. The artist must have exhibited work as a Photo-Realist by 1972 to be considered one of the central Photo-Realists.

5. The artist must have devoted at least five years to the development and exhibition of Photo-Realist work.

However, due to the non-artistic value of the result of this work, only the first three points will be considered. Therefore, if the result of the model was compared to an artistic work, and only if the result is like a photograph, it would be evaluated as photorealistic. Digital twins are used in smart manufacturing to connect physical and virtual spaces [264], build autonomous manufacturing cells [265], achieve high productivity and support decision making [266]. The quality assessment of the model output cannot be only based on human perception.

Traditionally, the Discreet Fourier Transform is the solution considered for image fidelity because it decomposes an image into its frequency components that make it up. The DFT is calculated with the Fast Fourier Transform algorithm [267-268], and belongs to the Fourier Transforms family.

The Fourier transform was first mentioned in 1822 [269]. In this work, Fourier claimed that any function of a variable, whether continuous or discontinuous, can be expanded in a series of sines of multiples of the variable, and provided the foundation for the current Fourier transform.

The Fourier transform, also called Continuous Fourier transform, of an integrable function $f: \mathbb{R} \to \mathbb{C}$, is mathematically expressed, in 1D, as (Equation 73):

$$F(u) = \int_{-\infty}^{\infty} f(x).\, e^{-i.2\pi.xu}\, dx \qquad \text{Equation 73}$$

Where $f(x)$ is the spatial function, and u, a given frequency.

The inverse Fourier transform for the same function, at the same frequency, in 1D, is (Equation 74):

$$f(x) = \int_{-\infty}^{\infty} F(u).\, e^{i.2\pi.xu}\, du \qquad \text{Equation 74}$$

The discrete Fourier transform can be defined as the Continuous Fast Fourier transform applied to discreate function. A function is defined as discrete when it is only defined for a set of numbers, called samples, that can be listed.

The Discreet Fourier Transform is expressed as (Equation 75):

$$F(p) = \sum_{m=0}^{M-1} f(m).e^{-i.2\pi.\frac{pm}{M}} \quad \text{and } p = 0 \ldots\ldots M-1$$ Equation 75

Where $f(m)$ is the spatial function, M, the number of samples and m, a given frequency.

The inverse Discrete Fourier transform is (Equation 76):

$$f(m) = \frac{1}{M}.\sum_{p=0}^{M-1} F(p).e^{i.2\pi.\frac{pm}{M}} \quad \text{and } m = 0 \ldots\ldots M-1$$ Equation 76

Equation 73, 74, 75 and 76 are expressed in 1D. However, an image is a 2D object. Therefore Equation 75 becomes Equation 77, and Equation 76 becomes Equation 78:

$$\begin{cases} F(u,v) = \iint_{-\infty}^{\infty} f(x,y).e^{-i.2\pi.(ux+vy)}\,dx\,dy \\ f(x,y) = \iint_{-\infty}^{\infty} F(u,v).e^{i.2\pi.(ux+vy)}\,du\,dv \end{cases}$$ Equation 77

Where $f(x,y)$ is the image function with x, y as spatial coordinates, and u, v, the two frequencies defining the image, where u is along x and v is along y axes.

$$\begin{cases} F(p,q) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n).e^{-i.2\pi.\frac{pm}{M}}.e^{-i.2\pi.\frac{qn}{N}} \\ f(m,n) = \frac{1}{MN}.\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(p,q).e^{i.2\pi.\frac{pm}{M}}.e^{i.2\pi.\frac{qn}{N}} \end{cases}$$ Equation 78

Where $f(m,n)$ is the image function with m, n as spatial coordinates, and p, q, the two frequencies defining the image, where p is along m and q is along n.

Graphically, the Fast Fourier Transform of a 2D image corresponds to the plot of its frequencies. Figure 5.13, shown an example, used as reference, of the Fast Fourier Transform of a circle. The creation of the image, and its Fast Fourier Transform was done and calculated in MATLAB. Note that the logarithm of the absolute of the Fast Fourier Transform was used to visualise the whole frequency spectrum, whereas the Fast Fourier Transform only allows the fundamental frequency to be observed.

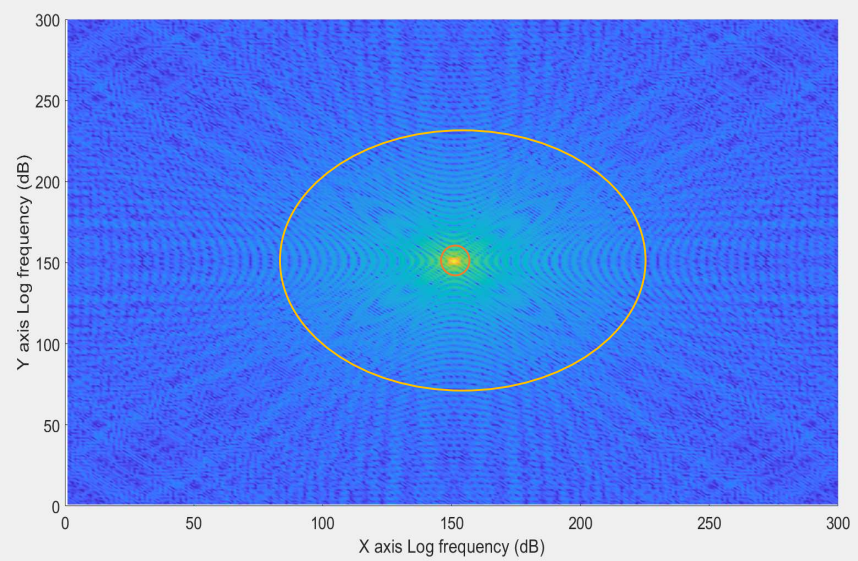


Figure 5.13: FFT for a circle

The sphere was chosen because its Fast Fourier Transform is characteristic, and simple to analyse. As explained in Section 4.2.3, in Chapter 4, an image is the result of the information obtained by the light beams hitting objects in a scene.

When the light rays hit a circle, three similar distinctive patterns are observable in the frequency domain, as a function of the object orientation. The first pattern comes from the high frequencies hitting the edges of the circle, generating a repetition of a rotational pattern (outside of the yellow and orange circles in Figure 5.13). The second pattern corresponds to low frequencies hitting the surrounding of the circle and correspond to a rotational pattern (in the yellow circle in Figure 5.13). The last pattern is a circle in the middle (in the orange circle in Figure 5.13) of the image, representing again the zero-frequency value, and corresponding to the inside of the circle.

The Fast Fourier Transform is widely used for image analysis, image filtering, image reconstruction and image compression [270], where its magnitude is used to determine the intensity of each frequency, and its phase, to visualise the shape of the information contained in the image as illustrated in Figure 5.13[271-272].

The Discrete Fast Fourier Transform is used as a transform from the pixel-domain into the frequency-domain [273], to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers [274]. It can be used in a blind image quality assessment of Gaussian blurred images [275], hide images in an image for steganography [276] and safety purposes [277], as well as in Image Quality Measure by measuring the changes in phase and magnitude between the reference image and distorted images to compute the quality score [278].

In this work, the Fast Fourier Transform and the Discrete Fast Fourier Transform are used to performed image quality measurement, coupled with the measurement of the degree of blur or sharpness of the image [279-280] by utilisation of image quality assessment algorithms, to construct comparison matrices.

5.3.2.Methodology

The same methodology as the one used in the camera characterisation experiment (Section 5.2.2.4) was used to perform the image fidelity.

Fast Fourier Transform

The frequency spectrum of the real model and each step in creating the final virtual camera model from the basic camera model was visualised using the Discret Fast Fourier Trasnform calculated with the MATLAB function *fft2*. The images used were generated in the final virtual model by taking a picture of the 0.09 m white polyester sphere on a black background, placing the camera at 0.5 m and 2.5 m from the object. Only these two distances were used to understand the impact of the modification of the camera components, as well as the impact of distance. Furthermore, it was concluded with the results of the camera characterisation in Section 5.2.2.6 that the final camera model was better than the real camera model when the camera was close to the object, and a similar response when the camera was far from the object.

All image processing was completed in MATLAB using the procedure described in Section 3.3.4 of Chapter 3, and the logarithm of the Discret Fas Fourier Trasnform was used in the calculation. This choice was made because it is difficult to visualise the whole frequency spectrum with the Discret Fas Fourier Trasnform, due to a scaling factor problem caused by the large difference in amplitude between the fundamental frequency and the other frequencies.

Blur percentage

The blur percentage in the real and virtual images were quantified by using the MATLAB process described in Section 3.3.5.

5.3.3.Results

All Fast Fourier Transforms and their logarithms, for both sphere-camera distances, were plotted only for the basic virtual camera model and the real camera model due to the high similarity between all plots.

Fast Fourier Transform (sphere-camera distance: 0.5 m)

Figure 5.14 and Figure 5.15 present the MATLAB plot of the Fast Fourier Transform for the basic virtual camera model and the real camera model, and they were taken with the higher pixel resolution possible, but they are still limited. The x and y axes represent the image frequencies in Hz, and the z axis, the magnitude [281]. Noted that, in MATLAB, the magnitude refers here at the strength of the frequency components relative to other components [282].

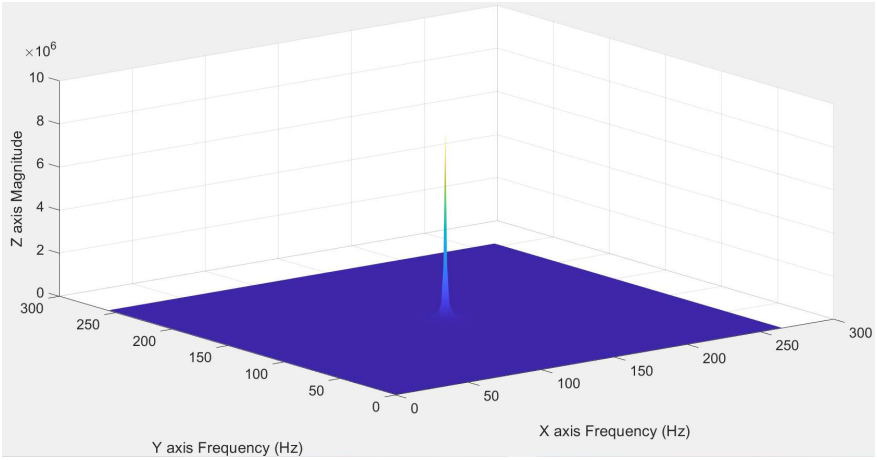


Figure 5.14: Fast Fourier Transform of the basic virtual camera model (0.5 m)

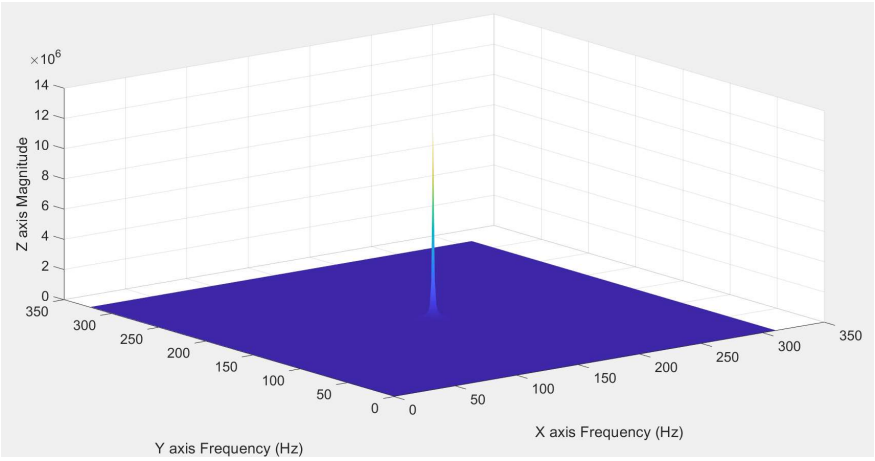


Figure 5.15: Fast Fourier Transform of the real camera model (0.5 m)

On the basis of the two Figures, the Fast Fourier Transform of the basic virtual camera model and the real camera model are very similar in shape, but both magnitudes and the frequencies are different. The fundamental of the basic virtual camera model is at 133 Hz on the x-axis, and 129 Hz on the y-axis with a magnitude of 8,755,191, while the fundamental of the real camera model is at 156 Hz on the x-axis, and 162 Hz on the y-axis with a magnitude of 12,494,169.

Table 5.4 shows the fundamental frequencies, and the fundamental magnitude for each modification of the basic camera model components until the final virtual camera model is obtained.

Table 5.4: Fundamental frequencies and amplitudes for each camera modification in Reality and in Blender (0.5 m)

Environment	Camera modifications	Frequency (Hz) x-axis	Frequency (Hz) y-axis	Magnitude
Real	Real model	156	162	12,494,169
Blender	Basic model	133	129	8,755,191
	F-stop 2	131	127	8,829,073
	Depth of focus	142	132	8,904,545
	Noise level	132	126	8,808,700
	Lens distortion	134	130	8,847,550
	Aperture shape (Blades) (Final model)	146	138	8,943,001

According to Table 5.4, the real camera model has its fundamental at higher frequencies than all of the Blender camera models; with a magnitude 50 percent higher.

Among the Blender camera models, it seems that the modification of the basic camera model components does not really impact the magnitude of the fundamental (between 8,829,000 and 8,943,000). Furthermore, the same conclusion is drawn as in the case of the camera characterisation experiment (Section 5.1.2.6): the depth of focus and the aperture shape (moving from a perfect circle to lozenge shape) have the greatest impact on the camera frequency spectrum. The impact of noise level and lens distortion is also better visualised than in Section 5.1.2.6, showing that they decrease the amplitude of the fundamental and vary its frequencies (frequencies (x, y) with the depth of focus: 142 dB, 132 Hz; with the noise level: 132 Hz, 126 Hz, with the lens distortion: 134 Hz, 130 Hz).

However, in Figure 5.14 and Figure 5.15, only the fundamentals are observable due to the scale factors caused by the difference of magnitude between the frequencies. For instance, in the basic virtual camera model Fast Fourier Transform, the magnitude of the fundamental is equal to 8,755,191; while the magnitude of the low and high frequencies is typically between 0 and 6,100.

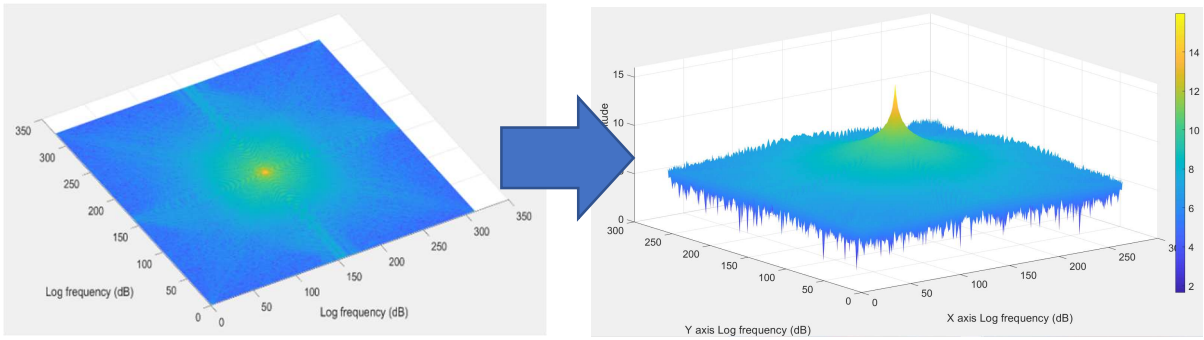


Figure 5.16: 2D and 3D plot of the Fast Fourier Transform logarithm

The colour legend of Figure 5.16 and for other Fast Fourier Transform logarithm plots is:

- Dark blue: High frequencies
- Cyan: Low frequencies

- Green: Low frequencies
- Yellow: Zero-frequency values

For a better reading of the camera frequency spectrum, the logarithm of the Fast Fourier Transform was plotted in Figure 5.17 and Figure 5.18 for the basic virtual camera model, and the real camera model. The axes of these Figures were the log of the image frequencies in dB on the x and y axes, and the magniture on the z axis.

Figure 5.17 and Figure 5.18 were plotted with the MATLAB function *surf* in 3D. The link between the explanation of how to read the Fast Fourier Transform logarithm in a 2D in Figure 5.13, and this plot is illustrated in Figure 5.16.

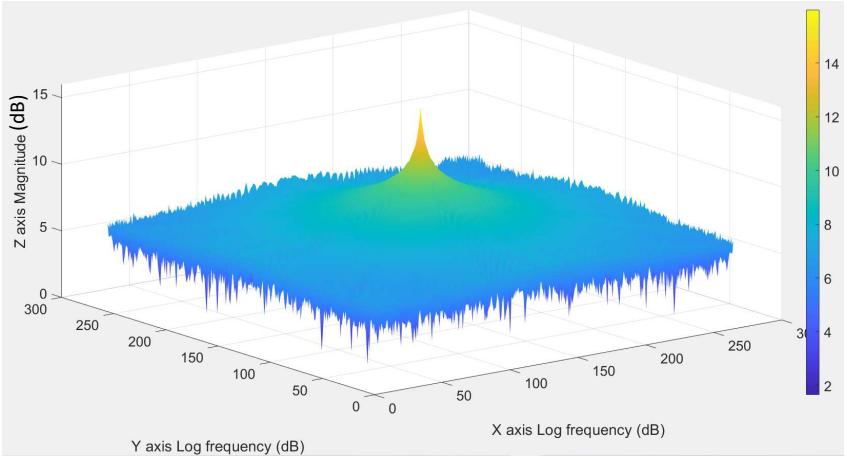


Figure 5.17: Log(Fast Fourier Transform) of the basic virtual camera model

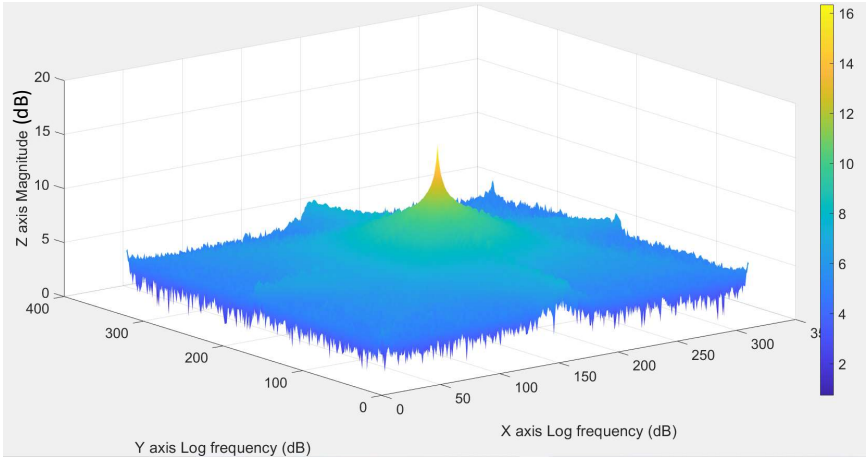


Figure 5.18: Log(Fast Fourier Transform) of the real camera model

According to Figure 5.17 and Figure 5.18, the only graphical result observable are the difference of magnitudes between the fundamental frequency and harmonies of the basic virtual camera model and the real camera model (Figure 5.14 and Figure 5.15).

To provide a more in-depth analysis of the camera's frequency spectrum and the impact of changing the camera's components on it, nine frequencies were chosen to give a simple visualisation of the magnitude spectrum of the camera models. These frequencies were chosen randomly around the fundamental point, as shown in Figure 5.19, to give four low frequencies and four high frequencies on either side of the fundamental point, which was also chosen as the centre point of the set. Their coordinates are given in Table 5.5.

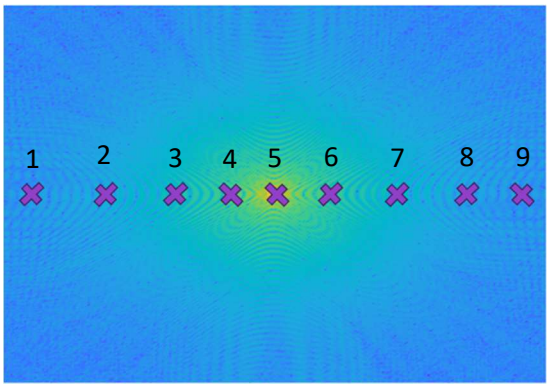


Figure 5.19: Visualisation of the frequencies chosen

Table 5.5: Chosen frequency coordinates

Frequency number (Figure 5.19)	**Coordinates (x, y)**
1	(7, 37)
2	(19, 51)
3	(30, 48)
4	(38, 46)
5	$(\frac{Image_{height}}{2}, \frac{Image_{width}}{2})$
6	(49, 45)
7	(54, 40)
8	(63, 60)
9	(75, 45)

In this experiment, the magnitude of the frequencies chosen has the test to quantify the impact of the complexity of the virtual camera model, with the results presented in Figure 5.20 and Table 5.6, for a 0.5 m object-camera distance. In Figure 5.20 , the x-axis scale represents frequencies chosen in Figure 5.19, and the y-axis scale, the magnitude of the logarithm of the Fast Fourier Transform. In Table 5.6, the points referred to the frequency number given in Figure 5.19 and Table 5.5.

Furthermore, the results of this experiment presented in Figure 5.20 and Figure 5.26, are plotted under a XY scatter plots for a better readability.

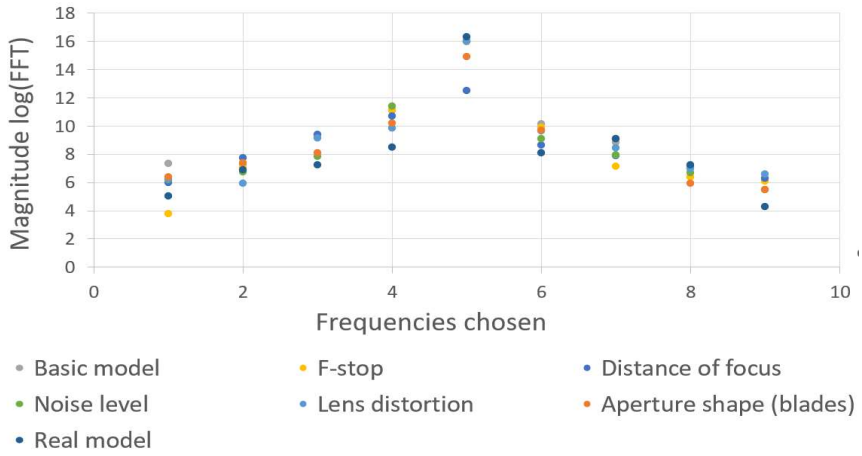


Figure 5.20: Magnitude of the logarithm of the Fast Fourier Transform (FFT) for a camera-sphere distance of 0.5 m

Regarding Figure 5.20, the data for each configuration of camera model describes similar frequency profiles, a curve increasing until the fundamental, and decreasing afterwards. Furthermore, variations are observed in the frequency profile of each camera model, meaning that they do not have the same frequency spectrum.

Table 5.6: Logarithm of the Fast Fourier Transform for the chosen frequencies 1, 3, 5, 7 and 9, at an object distance of 0.5 m

Camera model	Point 1 (dB)	Point 3 (dB)	Point 5 (dB)	Point 7 (dB)	Point 9 (dB)
Real model	5.09	5.24	16.34	9.10	4.26
Basic model	7.33	9.30	15.99	8.84	6.15
F-stop 2	3.76	9.20	15.99	7.15	6.06
Depth of focus	5.98	9.42	12.52	7.87	6.27
Noise level	6.16	7.85	15.99	7.93	5.47
Lens distortion	6.22	9.14	15.99	8.43	6.57
Aperture shape (Blades) (Final model)	6.37	8.11	14.89	9.07	5.47

According to Table 5.6, the addition of complexity to the virtual camera model modifies the distribution of the frequency across the spectrum. For example, the magnitude of first low frequency (Point 1) in the virtual camera model (F-stop of 2, and a depth of focus infinity) is at 3.76, while in the same model with a depth of focus equal to 0.3 m, it is equal to 5.98. In addition, the fundamental frequency
(Point 5) is not the same between the real camera (16.34), and the final virtual camera model (14.89).

Finally, it is noticeable that the complexity added to the camera model has a different impact on the camera spectrum (identified as the camera signature in the image). The depth of focus has an impact on the low frequencies (points 1-3); the level noise and the lens distortion, on the fundamental (point

5), the high (points 6-7) and the low frequencies (points 1-3); and the aperture shape (Blades) on the fundamental (point 5) and high frequencies (points 1-3).

Fast Fourier transform (sphere-camera distance: 2.5 m)

In this Section, the results are presented in a similar manner to the 0.5 m sphere-camera distance. It is noted that with the distance two phenomenum are observed:

- The image is blurred as illustrated in Figure 5.21, and leads to a reduction in the magnitude and frequencies of the camera spectrum.
- The camera pixel resolution decreases with increasing camera-sphere distance, making the sphere detection in the image more difficult.

Figure 5.21: Evolution of the real sphere cropping blur for an object distance of 0.2 m, 1 m and 2.5 m

As previously explained, only the Fast Fourier Transform of the basic virtual camera model, and of the real camera model are presented in Figure 5.22 and Figure 5.23. In both Figures, the x and y axis are the image frequency in Hz, and the z axis is the magnitude.

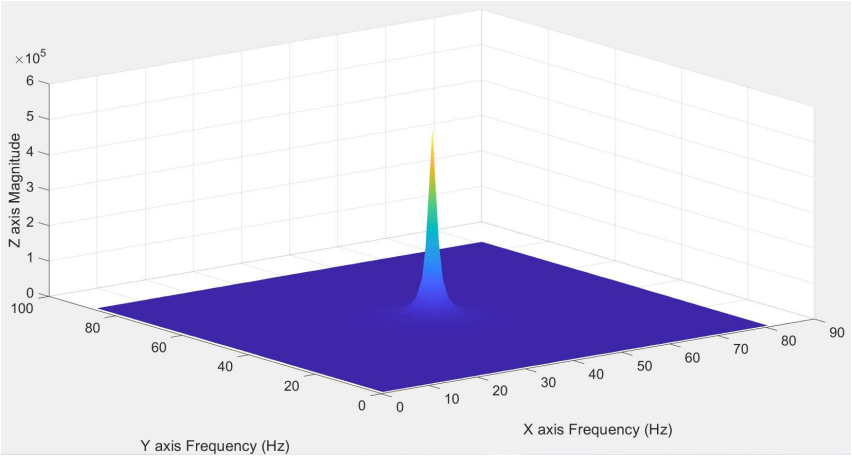


Figure 5.22: Fast Fourier Transform of the basic virtual camera model (2.5 m)

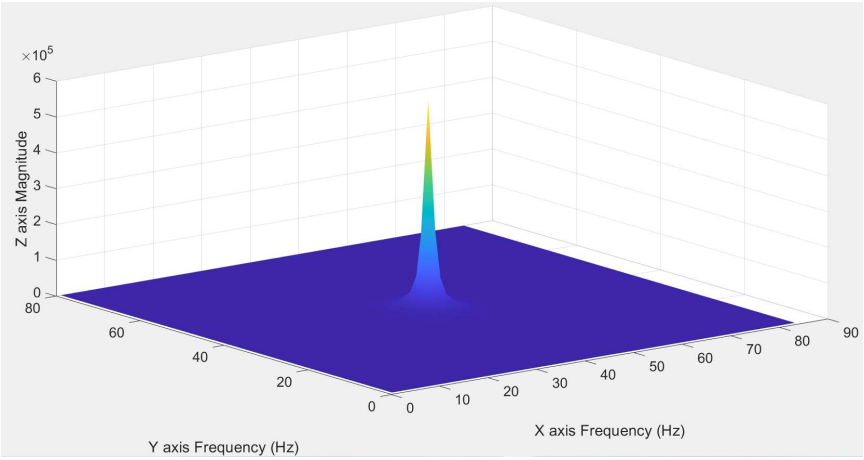


Figure 5.23: Fast Fourier Transform of the real camera model (2.5 m)

Similar to the analysis of the 0.5 m data, the Fast Fourier Transform of the virtual camera base model and that of the real camera model appear to have a similar shape. However, their fundamental frequencies are different. For the virtual base model, this frequency has a magnitude of 529,101, and is located on the x and y axes at 41 Hz and 44 Hz; whereas for the real model camera, its magnitude is 582,253; located at 43 Hz, and 41 Hz.

In addition, the impact of the object distance is clearly observable. When the camera-sphere distance was at 0.5 m, the fundamental of the basic virtual camera model was at 133 Hz on the x-axis, and 129 Hz on the y-axis with a magnitude of 8,755,191; whilst at 2.5 m, the fundamental is at 41 Hz on the x-axis, and 44 Hz on the y-axis with a magnitude of 529,101.

Table 5.7: Fundamental frequencies and amplitude for each camera modification in Reality and in Blender (2.5 m)

Environment	Camera modifications	Frequency (Hz) x-axis	Frequency (Hz) y-axis	Magnitude
Real	Real model	43	41	582,253
Blender	Basic model	41	44	529,101
	F-stop 2	41	41	524,856
	Depth of focus	42	40	565,467
	Noise level	41	40	563,474
	Lens distortion	40	43	567,611
	Aperture shape (Blades)	42	42	554,169

According to Table 5.7, the same conclusions as detailed for Table 5.4 are evident. However, the impact of the depth of focus, the noise level, the lens distortion, and the aperture shape are less apparent than for the images taken at 0.5 m. This observation is due to the blurring of the image, the addition of a low-pass filter and the smoothing of the frequency spectrum, making the contribution of each parameter more difficult to identify.

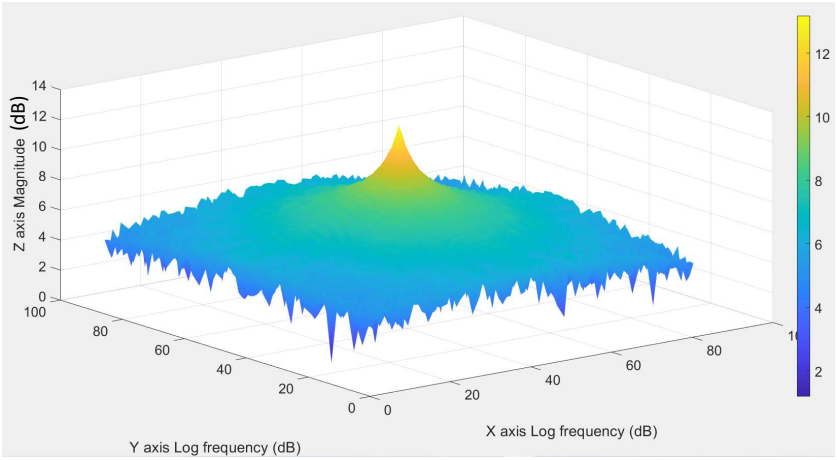


Figure 5.24: Log(Fast Fourier Transform) of the basic virtual camera model (2.5 m)

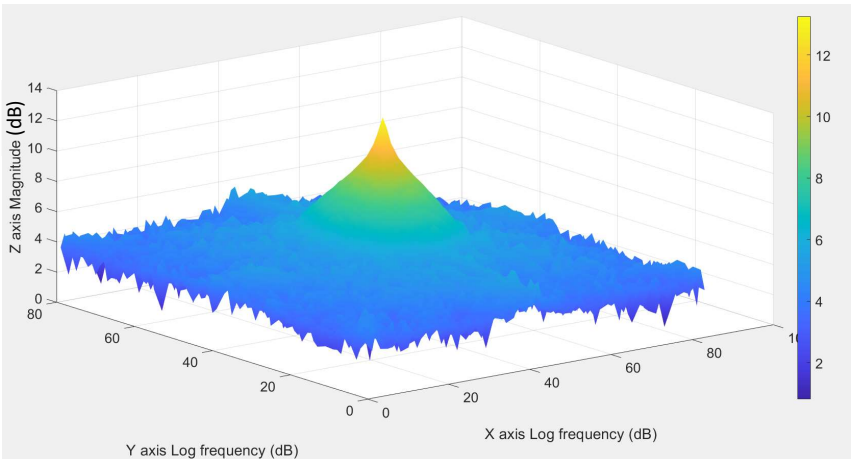


Figure 5.25: Log(Fast Fourier Transform) of the real camera model (2.5 m)

A graphical analysis of Figure 5.24 and Figure 5.25 shows that the distribution of colours is different. Figure 5.24 seems to be composed of less dark blue colours than Figure 5.25. As explained in Figure 5.16 of this chapter, the high frequencies are in blue colour. This colour difference could therefore indicate that the spectrum of the basic virtual camera at 2.5 m from the object is composed of less high frequencies than the real camera model for a similar camera distance. In addition, the colour bar in Figure 5.17 and Figure 5.18 ranges from 0 to 16, whilst the colour bar in Figure 5.24 and Figure 5.25 ranges from 0 to 12. It seems that the magnitude of the fundamental (the peak with the largest magnitude, hence the maximum of the colour bar) is affected by distance, and that its magnitude decreases. However, the graphical analysis is not sufficient, and these results are verified by Figure 5.26, and Table 5.8.

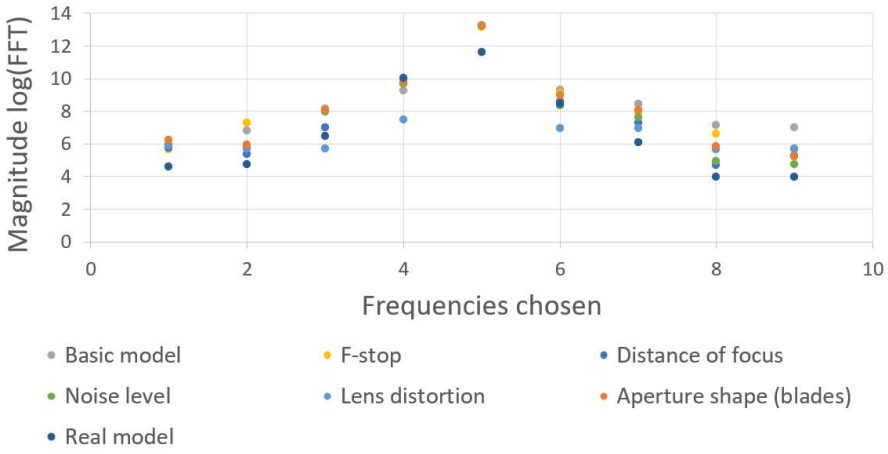


Figure 5.26: Magnitude of the logarithm of the Fast Fourier Transform for a camera-sphere distance of 2.5 m

In Figure 5.26, the x-axis scale represents the logarithm of the Fast Fourier Transform, and the y-axis scale represents the positions of the frequencies chosen in Figure 5.19. The data for each configuration has a similar profile to the data shown in Figure 5.20. However, in Figure 5.20, the magnitude of the y-axis is between 0 dB and 16 dB, whilst it is between 0 dB and 13 dB in Figure 5.26. The fundamental frequency is lowest when the distance is large between the camera and the sphere, and higher when the camera is closest to the sphere.

This magnitude difference of 3 is due to the distance behaving as a filter. As explained in [–283-285] in the generation of hybrid images (a single image can be interpreted alternatively as two different types of information, with the modulation of the viewing distance), high frequencies are better perceived at short distances, and low frequencies are better perceived at high distances.

Table 5.8: Distance 2.5 m, logarithm of the Fast Fourier Transform

Camera model	Point 1 (dB)	Point 3 (dB)	Point 5 (dB)	Point 7 (dB)	Point 9 (dB)
Real model	4.6	6.45	11.62	6.07	3.98
Basic model	5.95	8.17	13.18	8.43	6.98
F-stop 2	5.66	8.06	13.17	7.95	5.66
Depth of focus	5.75	7.02	13.25	7.30	5.29
Noise level	5.96	7.95	13.24	7.60	4.73
Lens distortion	5.88	5.72	13.25	6.96	5.71
Aperture shape (Blades)	6.22	8.03	13.23	8.06	5.21

In Table 5.8 the final virtual camera model and the real model do not have the same frequency profile. The virtual camera always has higher frequencies than the real camera. Furthermore, the same conclusion as in Figure 5.26 is observable, the fundamental magnitude is lower in Table 5.8, than in Table 5.6, and the impact of the modification added to the model is smoother, as already observed in Table 5.7. For example, in Table 5.8, the first low frequency (Point 1) for a virtual camera model with a F-stop of 2, is 5.66 dB, and for the same model with a distance focus of 0.3 m, the first low frequency is 5.75 db. This difference comes from the distance causing a low-pass filter effect on the image.

Blur percentage

To complete the image fidelity process, the blur percentages between the reference image and the real and virtual camera models were compared and plotted in Figure 5.27.

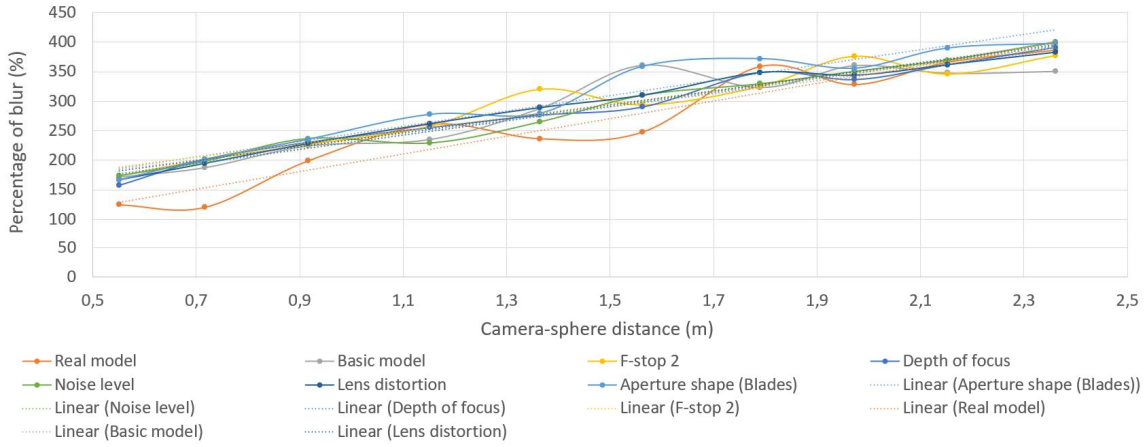


Figure 5.27: Percentage of blurriness for the real and virtual models

Regarding Figure 5.27, the results give similar data trends, with variations coming from the difference between the camera models (real and virtual camera model) used. In addition, the starting point of the real model, and the virtual models is different, and an almost constant difference between them is visible.

Table 5.9: Percentage of blurriness and linear coefficient of the trend lines for object distances of 0.5 m, 1.15 m, and 1.79 m

Camera model	0.5 m (%)	1.15 m (%)	1.79 m (%)	R^2
Real model	124.85	261.22	359.30	0.9199
Basic model	171.69	235.50	323.60	0.8655
F-stop 2	173.56	257.57	325.26	0.9259
Depth of focus	158.19	255.67	348.71	0.9746
Noise level	173.58	229.17	329.52	0.9828
Lens distortion	166.98	262.15	348.54	0.9745
Aperture shape (Blades)	166.96	277.73	371.97	0.9371

According to Table 5.9, the real model seems to be less blurred than the virtual model when the camera is closer to the sphere (0.5 m). In addition, the modification of the depth of focus seems to give the second less blurred image when the camera is closer to the scene (0.5 m), while the basic model gives the least blurred results when the camera is far away from the sphere (1.79 m).

The model with the modification of the F-stop, the noise level, the lens distortion, and the aperture shape blur more the image than the other models at 0.5 m. However, the aperture shape model gives the highest blur results over the whole distance range.

From these results, similar conclusions as previously are reached. However, it seems that the best virtual model defined in Section 5.1.2.6 is the more blurred, generating a difference of 44% at 0.5 m, and 12% at 1.79 m with the real model.

5.3.3.1. Discussion

Three approaches and sets of characteristics have been considered in this element of the research; lighting intensity, object texture, and camera parameters. With respect to the lighting conditions the intensity of illumination has been the key test with values ranging from 5 W to 39 W when illuminating a virtual sphere. Whilst nominally straightforward to complete both in the real-world setting and the virtual model setting, a clear correlation of measurement performance has been demonstrated with significantly better agreement of the magnitude of sphere diameter deviation calculations (compared to the default Blender settings), albeit not in absolute agreement.

This element of the analysis has also identified that lighting parameters used in the modelling environment are somewhat different to the real-world measurement of light intensity and significant care is required to correctly define equivalent virtual lighting settings but may also have an impact. It is also noted here that changes of colour spectrum were not within the remit of the current work.

A key challenge has been to consider object (and even environment) texture. In this context the use of 2D profile and 3D areal surface texture parameters (ISO 4287, ISO 25178-2) would be regarded as the most common meaningful way to measure the surface texture of real-world surfaces. However, Blender does not provide the facility to model the true surface texture values of an object in this manner. Instead, textures are defined as having a value between 0 and 1, representing the percentage of the texture applied to the object. This is a subjective parameter which the user needs to visually optimise.

The texture experimentation has shown that having the right texture with the right amount of surface imperfections is important and will impact the final performance capability of the camera-based measurement system. The results demonstrate that changing the texture, changes the measurement of the virtual sphere and through careful selection reasonable agreement can be achieved with the real-world data.

With respect to the light characterisation, it was reinforced that brighter the illumination is, the more details will be revealed and exaggerated. In addition, the darker the illumination, the better the measurement is, until it is impossible to detect the sphere, because it is too dark.

Finally, the parameters that define the camera system within Blender have been explored based on a Raspberry Pi V2. These parameters were the depth of focus, the number of aperture blades, the aperture shape, the distortion, and the noise. It has been shown that the depth of focus and the aperture shape are important parameters, directly influencing the quality of sphere diameter measurement. The work has also revealed that it is very difficult to exactly model the real camera with the parameters available in Blender, but useful toolsets are available that can quantify camera performance and allow close agreement of the virtual camera with the real-world camera.

Analysis of the 2D frequency components using Discrete Fourier Transforms allows for frequency signatures to be mapped within the virtual images and then compared to the real-world equivalents. Likewise, it has been demonstrated that the blurriness percentage of the virtual images can be quantified and correlated to the real-world equivalent images. With specific optimisation of the virtual camera model (F-stop 2 + depth of focus 0.3 + noise level + lens distortion + aperture shapes) then Blender performed better than reality for the short object distances but tended to agree better with the real-world data for longer object distances.

The research has demonstrated that photorealistic concepts can be used to proactively optimise (in this instance deliberately degrade) the default Blender environment to better model the real-world equivalent measurement scenario (camera measurement system measuring sphere diameters). The level of agreement depends on key parameters, in this instance; light characteristics, object texture, and enhanced camera definitions. Likewise, the ability to refine the correlation between the Blender modelled measurement system and the real measurement system is limited by the extent of Blender software toolsets and the subjective interpretation of the operator.

The results demonstrate that on an individual basis, parameter optimisation can be achieved that leads to equivalent camera system measurement performance in the Blender virtual environment. The challenge now is to extend this work to more complex objects, more complex environments, and the simultaneous use of multiple cameras in a photogrammetry context.

5.4. Conclusion

This element of research has explored how photorealism concepts can be used to better develop digital models in a 3D virtual reality modelling environment such as Blender. Specifically, this work has

been considering the simulation of metrology-based camera measurement systems, with the eventual aim of being able to model such systems in an industrial context and provide rapid camera system optimisation thus avoiding the need to persistently relocate cameras in the real-world environment.

Previous research, in Chapter 4, identified that the lack of realism of the virtual scene compared to the real-world scenario (e.g., refined camera parameters, light-objects interactions), and modelling constraints within the software itself (software selectable options) were two significant issues that caused mismatch of up to an order of magnitude between the virtual measurement system and the real-world measurement system. It was shown that in its native and default configuration settings – Blender could simulate metrology-based camera measurements too well and produce erroneously "perfect" measurement data (based on sphere diameter measurements).

Chapter 5 has proactively explored and optimised various model parameters (boundary conditions) to better mimic the real-world equivalent. It has further exposed the subjective nature of the Blender modelling environment which clearly has been developed and is predicated on artistically driven film and animation outputs rather than a metrologically correct measurement environment.

Through the characterisations of the environment, the illumination environment, the object-light interactions, and the camera model the following results were found:

- Real and virtual cameras are intrinsically different, and both behave like filters.
- The surface textures are artistically defined, and it is not possible to use the real values of the measured surfaces.

This work does not provide a set of rules for setting up a virtual environment in Blender, due to the context-dependent nature. It is almost a philosophy, a path for the user to follow to identify, step by step, the key factors required to generate a photorealistic virtual reproduction, and how these factors should be modified. However, recommendations can be given, as this work can be used as a basis and guide to start designing a photorealistic virtual reproduction of a physical model.

Thus, the following list can be generated:

- A basic knowledge on how Blender is advised, before starting any complex work.
- Due to the utilisation of ray tracing to generate illumination, the virtual reproduction should be placed in a cube, enough large to encompass the model created, and to have the light bouncing as wanted to generate the global illumination environment.
- As a remind, the light units in Blender are in light illumination and not in electrical power. A conversion between the real light power (electrical power) to the Blender units should be done.
- The virtual reproduction is not generated from the point of view of a camera, like an environment photographed. To generate a photorealistic virtual reproduction, the user shall think about how the final rendering, i.e., the photograph, is constructed, and looks like, to identify the key parameters needed to be modified in Blender.
- Keep in mind that the light is important because it brings the real environment to life. However, it is a difficult parameter to set up, and only a trial-and-error strategy can be advised.
- Due to the necessity to "build an image", the environment (such as objects, textures) around the object under measurement should be considered.

- The camera model is also important factor. It is recommended to have a model as much as complex as possible; as shown in the camera characterisation section of this chapter. Keep in mind, that the real and the virtual cameras are intrinsically different as shown in this Chapter. So, it is impossible to recreate the real camera in Blender.
- The object texture management is physic bias, and artistic oriented. It then depends on the global illumination environment, and the objects composing the environment. Generation of benchmarks as a guid to find the right texture, and a trial-and-error approach were used in this work.

5.4.1. Virtual photorealistic model definition

Through the characterisation of the main photorealistic parameters (environment, light, surface textures, and camera), a virtual photorealistic model of the real scene (Figure 5.2) has been designed. In the following section, the final camera model, the final virtual environment, with the final texture used, constituting the final virtual photorealistic model will be presented.

The final virtual camera model was:

- Lens:
  - Type: Perspective.
  - Focal length: 3.04 mm.
  - Lens Unit: Millimetres.
  - Shift X and Y: 0.
  - Clip Start: 0.1 m.
  - Clip End: 1000 m.

- Depth of Field:
  - Distance: 0.3 m.

- Aperture:
  - F-stop: 2.
  - Blades: 4.
  - Rotation: 0°.

- Camera:
  - Sensor Fit: Horizontal.
  - Sensor width: 3.68 mm.
  - Sensor Height: 2.76 mm.

- Post Processing:
  - Noise level: 0.02 dB
  - Distortion level: 0.001

And the final virtual scene imitating the control room (virtual scene used in Chapter 6, and in two experiments in Chapter 7), is presented in Figure 5.28:

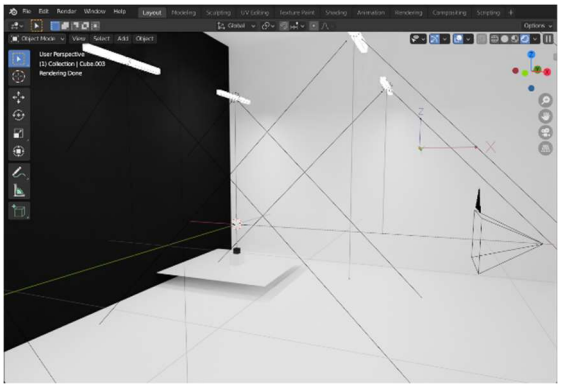


Figure 5.28: Photorealistic blender model of the real scene in Figure 5.5

The Blender model was composed of the same elements as the real environment (two cubes, the table, and the bar). The number of polygons defining the rectangles and cubes cannot be modified and is not identified in Blender. However, as explained in Section 4.4.1, the object resolution does not affect the measurement. The sphere was then modelled with 32 polygons.

The sphere was on a black background and are all based on a polygon mesh. The lights were rectangular lights placed at 1.12 m above the sphere, with a power of 38.65 W. Two lights were created for each "real" light, and oriented at + 45 and – 45 degrees to recreate the light spreading in the real environment.

It is noted that light power is defined depending on the light source, which makes it difficult to directly correlate the light intensity between a real-world environment and the equivalent environment in the Blender model. The textures were applied to the virtual objects by using the BSDF. The surface textures of the different elements constituting the experiment were identified as:

- White plastic cube / Printing cube or circle on a black background: Diffuse and reflected / Diffuse and reflected.
- Transparent cube: Transparent.
- Background: Diffuse.
- White wall: Diffuse and reflected.
- Table: Diffuse and reflected.
- White cube: Diffuse and reflected.
- Black cube: Diffuse.

In summary, the novelty of Chapter 5 can be identified as being:

- In the literature, photorealism is known as an art movement but has not been investigated as a possible solution to increase realism in digital twin applications.
- The exploration of the design of a photorealistic virtual model.
- The identification of the limits of the virtual model and its differentiation from the real model.
- The use of the Discret Fourier Transform and the blurriness index on the Blender images.
- The development of a photorealistic approach to degrade the virtual model and make it more realistic.
- Set of recommendations to design a photorealistic virtual reproduction of a physical asset was generated.
- The design of a photorealistic model, presented in Section 5.4.1, and used in Chapters 6 and 7, as the virtual model under test.

Chapter 6 will evaluate the robustness of the virtual system defined in this chapter, composed of the elements listed in Section 5.4.1. This study will achieve with three experiments where the response of the virtual environment is compared to the real response.

Chapter 6

System robustness

In the previous chapters, a photorealistic digital twin for multi-camera measurement systems has been designed in Blender. The robustness of this proposed system now needs to be investigated to determine whether it is stable, and to characterise its response. The experiments used, in Chapter 6, are similar to previous Chapters, e.g., the measurement of the size of various simple objects, in the same controlled environment, with the same equipment. A checkerboard and a zone circles board of two different sizes were used as calibration objects to identify the fluctuation in the calibration process using the reprojection error.

The experiments were carried out in the real and virtual environment, and MATLAB functions were again used to determine the size of the objects from the images taken. The photogrammetry method was used to calibrate the single camera and to calculate the reprojection error.

In this Chapter, Section 6.1 discusses the methodology of the experiments, and Section 6.2 presents the results obtained.

In addition, the following challenges are expected to be met:

- Has the photorealism process improved the virtual results? (Comparison between the results obtained in Chapter 3 with the results of Chapter 6).
- Have other sources of error been identified in relation to the parameter not considered in Chapter 5?
- Is there a correlation between real and virtual results?
- Is the photorealistic virtual model robust?

These questions lead to the following potential innovations:

- The new virtual model is robust.
- A correlation between real and virtual results has been established.
- New sources of errors have been identified.

6.1. Experiments presentation

To assess the robustness of the system developed in Section 5.4.1, three experiments were developed and simulated in Blender, for the purpose of:

1. Determining the performance of the photorealistic virtual environment as the mimic of simple and complex real-world scenarios.
2. Characterising the answer of the photorealistic virtual environment in the simulations of simple and complex objects.
3. Validating the digital model generated.
4. Identifying and investigating possible new sources of errors.
5. Quantifying the difference between the real world and its virtual replications in simple and complex object simulations.

The two first experiments are similar to those performed in Chapter 4 about measuring simple objects using individual cameras, and the position of a multi-camera measurement system with respect to calibration artefacts. The same methodologies are used; however, more objects are measured with the individual cameras, and the two boards presented in Chapter 5 are used to determine the position of the multi-camera measurement system.

The third experiment is similar to the first experiment in Chapter 3 (Section 3.4.1) and also consists of measuring the diameter of a sphere with a single camera. However, spheres of different diameters were used here, and the aim of this experiment was to make virtual predictions of the evolution of the deviation of the diameter by creating a benchmark of spheres.

Experiments from Chapters 4 and 5 were re-used because:

- The error budgets had already been explored.
- It allowed a comparison of the results between the two virtual systems developed respectively in Chapters 4 and 5, as well as to visualise the impact of the photorealism approach used.

6.1.1. Measurement of object dimensions with individual cameras

Methodology

The first measurement scenario was based on the sphere diameter measurement experiment presented in Chapter 3, Section 3.4.1. However, instead of measuring the diameter of a 0.09 mm sphere, the following objects were used:

- Two diffuse white cubes of 0.1 m and 0.2 m sides.
- Two diffuse white circles of 0.07 m and 0.09 m diameter also printed on a black background.

These objects were presented in Chapter 3, Section 3.1.1.

Measurements were non-sequentially repeated three times to provide understanding of potential variance as a function of camera placement; and the same MATLAB based set of algorithms as the ones presented in Chapter 3, Section 3.3.1 was used.

Real world objects measurements

This first experiment took place in a controlled environment (light intensity control, noise coming only from the cameras, etc.), with two rows of three 39 W LED, located at 1.30 m from each other on the x and y axes. As illustrated in Figure 6.1, the cubes of 0.1 m and 0.2 m sides were put on the top of a transparent cube of 0.3 m sides. Subsequently, as shown in Figure 6.2, the printed circles were attached to one side of the 0.2 m cube, whilst still situated on top of the transparent cube. These three set ups (cubes and 2D circles) used the same alignment criteria as defined Chapter 3, in order to maintain consistent experiments, and to avoid bringing more sources of error into the configuration by changing the camera height or its parameters.

The entire system was on a white table of 1 m side, placed 1.12 m below one of the lights. A white wall was on the edge of the set up to reflect the light, and a black background behind the objects to maximise image contrast.

The white table and the white cube were used to reflect the light to minimise shadows on the bottom of each object. The distance between the camera and the objects were measured in a similar way as explained in Chapter 5, Section 5.2.2.4, and the true distance between the camera and the objects were equal to the camera translation on the z-axis minus half of the object dimension. For instance, the distance between the camera and the 0.1 m cube was equal to the camera translation on the z-axis minus 0.05 m.

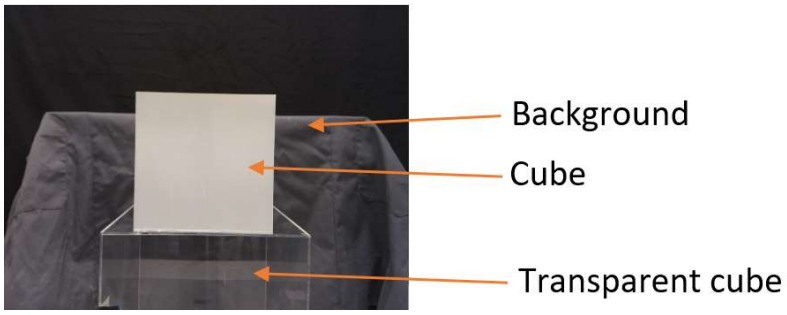


Figure 6.1: Cubes set up view from the camera

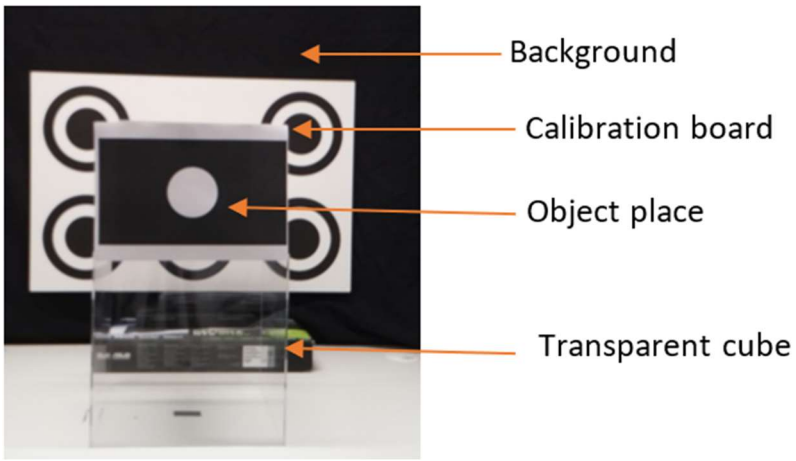


Figure 6.2: Circles printed set up view from the camera

Blender objects measurements

The real-world experimentation was then recreated in the virtual environment. The experiment again consisted of measuring the diameter and centre of a 0.09 m diameter white virtual sphere from pictures taken by the Blender simulated virtual pinhole camera located at (again) object distances varying from 0.5 m to 2 m, in steps of 0.2 m from the sphere. The whole scene was generated in Blender using a Python script as shown in Figure 6.3. For the purposes of visualisation within this chapter, Figure 6.3 does not include the black background that was used to maximise object contrast. Multiple repeats (n = 3) of the experiments were completed to determine any variance in the output.

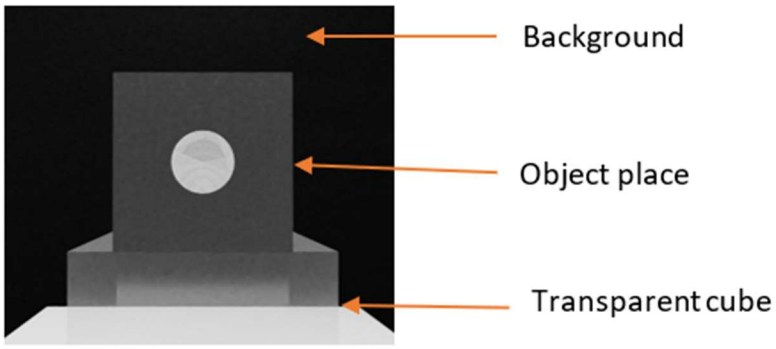


Figure 6.3: Virtual set up from the camera view

The final virtual set up, and the final camera model used to simulate the Raspberry Pi V2 were the models developed and presented in Chapter 5, Section 5.4.1. The same MATLAB functions and data processing developed for the real-world scenario were used.

6.1.2.Camera location relative to a calibration artefact

Methodology

Real world multi-camera location

The second measurement scenario compared the reprojection errors as exposed in Section 3.3.2 of Chapter 3, about cameras position optimisation. However instead of using only the zone circles boards, the checkerboard was also considered. This experiment was reproduced in both the real-world environment and the Blender virtual environment. The reprojection error was calculated with Equation 62, in Chapter 2, and the object distance varied from 0.5 m to 1.8 m with incrementing steps of 0.2 m.

As a reminder, two sizes of each artefact were used due to the field of view of the camera, with the smaller artefact between 0.2 m and 0.8 m, and the larger between 1 m and 1.8 m. In addition, the size of the square of the two checkerboards changed as well with the size of the board.

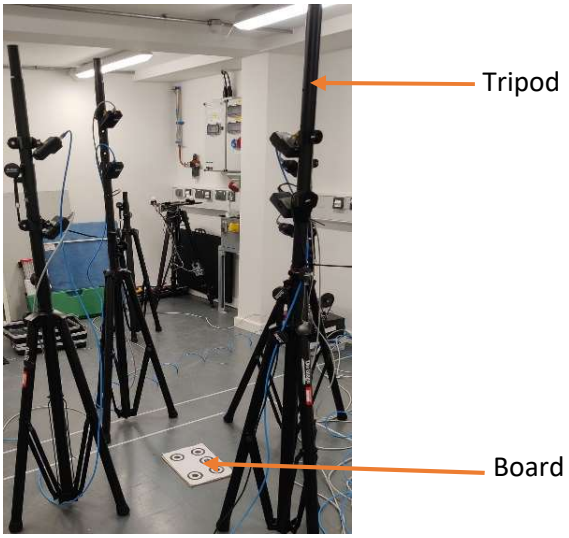


Figure 6.4: Real set up

Blender based multi-camera location

Within Blender, an equivalent set of eight virtual Raspberry Pi V2 cameras were placed in a square set up around the board, to mimic the real-world scenario as shown in Figure 6.4. Camera definitions were those as used for the sphere measurements with data processing following the MATLAB algorithms developed for the real-world scenario (Section 6.1.1).

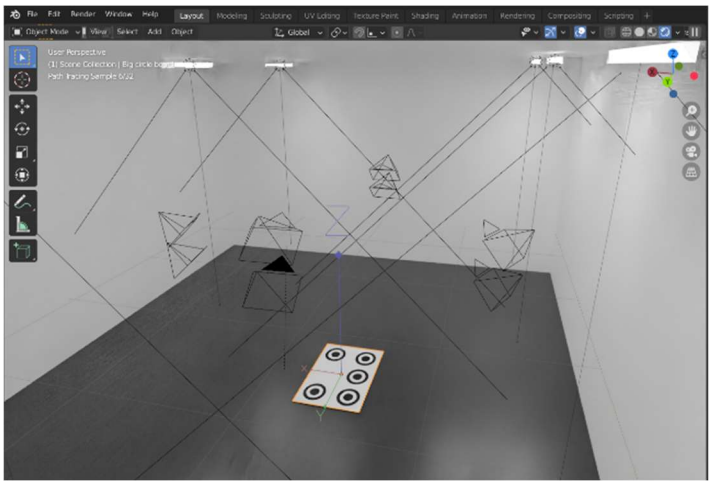


Figure 6.5: Virtual set up

6.1.3. Sphere benchmark

As explained previously, this experiment was similar to the sphere experiment in Section 3.3.2 of Chapter 3, and also consisted of measuring the diameter of a sphere with a single camera. However, the camera-sphere distance was varied between 0.5 m and 2 m with a step of 0.2 m; and spheres of 0.09 m, 0.15 m, 0.20 m, 0.25 m, and 0.30 m diameters, presented in Section 3.1.1 of Chapter 3, were used. In addition, the aim was not to validate the capacity of Blender to simulate simple object, but was to:

- Design a sphere benchmark.
- Validate the virtual prediction of the evolution of the deviation on the diameter of the spheres related to the increase of the camera-sphere distance.

The setup of this experiment, in the real and virtual environments, were the same as in Figure 5.2 (Section 5.2.2.2), and Figure 5.28 (Section 5.4.1), and the virtual camera model used was the one described in Section 5.4.1. Measurements were non-sequentially repeated three times to provide understanding of potential variance as a function of camera placement.

In Blender, the diameters of the five spheres were measured, while in the real-world equivalent only the 0.09 m, 0.20 m and 0.30 m were measured. This difference between the two experiments came from the non-availability of the 0.15 m and 0.25 m diameter real spheres.

The same MATLAB set of algorithms as used in Section 3.3.1 of Chapter 3, were used to measure the observable sphere diameter.

6.2. Results

6.2.1. Measurement of object dimension with individual cameras

6.2.1.1. Real-world objects analysis

Circles

The results from the real single camera measurement of the 0.09 m and 0.07 m circles are shown in Figure 6.6 demonstrating the deviation of the camera measured circles diameter from the actual diameter. The x-axis scale represents the distance between the camera position and the circles, and the y-axis scale is the mean value of the deviation on the circle diameter.

The mean value of each circle has been plotted with the error bar for each value. This error bar was calculated using the standard error of the mean for all values, which shows how far the sample mean is likely to deviate from the population mean. In addition, the mean value was calculated over three trials for the 0.07 m circle, whereas it was calculated over five trials for the 0.09 m circle due to the presence of an outlier at 1.32 m. This outlier appears to be due to background noise, illumination and the low resolution of the image at this distance.

The standard deviation of the 0.09 m circle answer was 0.44 mm, and the 0.07 m circle answer was 0.92 mm.

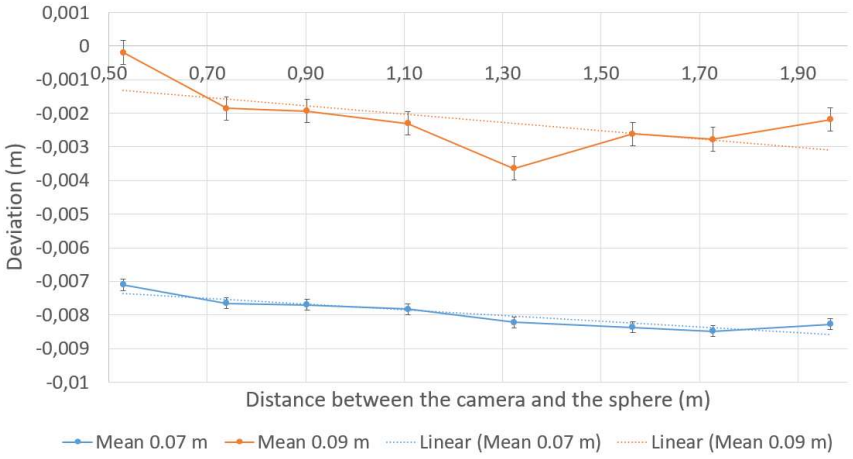


Figure 6.6: Deviation on the diameter of circles

The data demonstrates that for both answers, the deviation on the circle diameter increases with increasing object distance. In addition, the answer of the 0.07 m circle seems to increase almost constantly, while a peak is visible at 1.32 m (identified as the aberrant value) for the 0.09 m circle coming from aberration values in two trials of the set of five. The variations in the two answers come from the pixel resolution and the difficulty in keeping the set-up constant for each trial, and each position, as explained in Section 4.2.2.1.

Table 6.1: Deviation of circle diameters for object distances between 0.5 m and 2 m

Object distance	**0.53 m**	**0.90 m**	**1.30 m**	**1.72 m**
0.07 m Circle (mm)	-7.10	-7.70	-8.22	-8.48
0.09 m Circle (mm)	-1.21	-1.93	-3.64	-2.77

Regarding Table 6.1, a persistent difference of approximately 6 mm is noticeable between the 0.07 m circle and the 0.09 m circle. It seems that the smaller the object, the more difficult it is to detect in the image, due to the reduction of the pixel density and the effect of the distance on the circle resolution. To reinforce this idea, Figure 6.7 shows the pixel resolution of both circles (y-axis) as a function of increasing camera-sphere distance (x-axis).

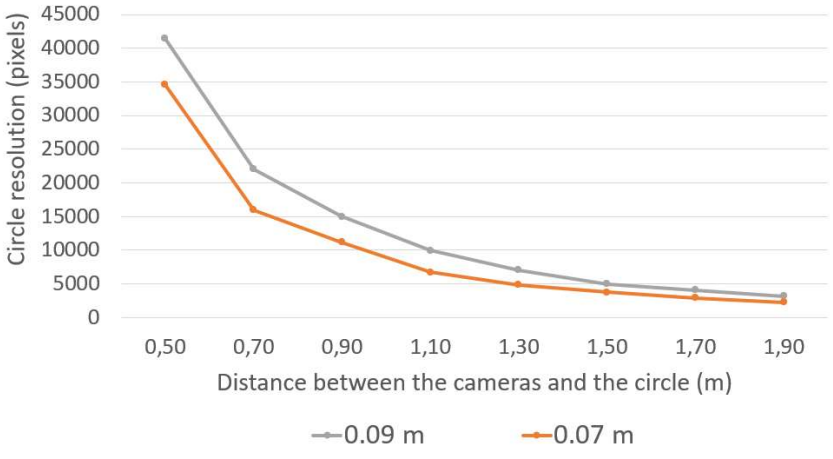


Figure 6.7: Circles resolution

The difference between the resolution of the two circles decreases with increasing of distance between the circles and the cameras (moving from a difference of 6870 pixels to 3150 pixels).

Cube of 0.1 m edges (small cube)

The results for the 0.1 m cube are presented in Figure 6.8. The x-axis scale represents the distance between the camera position and the cube, and the y-axis scale is the deviation of the height and the width of the cube. These two dimensions are theoretically equal but were dependent on the blurring of the image caused by the camera's field of view, the decrease in pixel resolution with increasing camera-to-object distance, and possible noise from the environment. Both dimensions were measured in MATLAB to check if the measurements were correct and to visualise the impact of these potential sources of error on the cube detection.

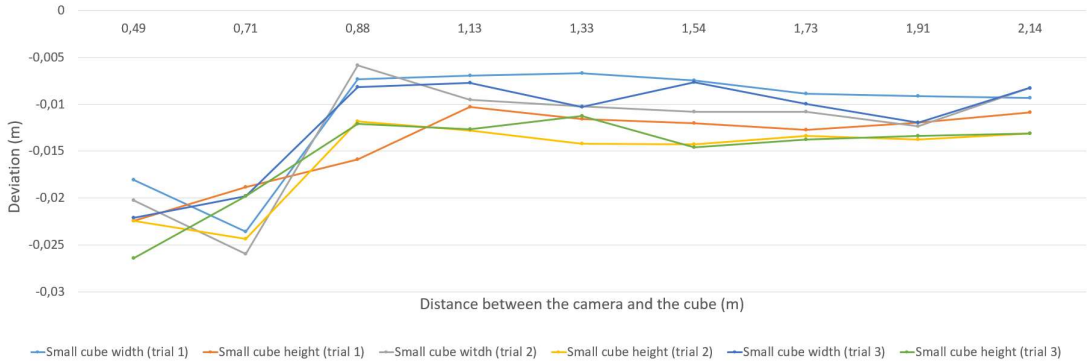


Figure 6.8: Deviation on the height and width of the 0.1 m cube

The data and Table 6.2 highlight that the three trials generated similar results, with similar data trends. For instance, at 0.5 m, the height deviation is between – 22 mm and – 16 mm, while the deviation width is between – 18 mm and – 22 mm.

Table 6.2: Dimensions deviation of a 0.1 m cube for an object distance between 0.5 m and 2 m

Object distance	**0.5 m**	**0.9 m**	**1.33 m**	**1.73 m**
Height Trial 1 (mm)	-22	-16	-12	-13
Height Trial 2 (mm)	-22	-11	-14	-13
Height Trial 3 (mm)	-26	-12	-11	-14
Width Trial 1 (mm)	-18	-7.3	-6.7	-8.8
Width Trial 2 (mm)	-20	-5.8	-10	-11
Width Trial 3 (mm)	-22	-8.2	-10	-10

According to Figure 6.8 and Table 6.2, two patterns are observable, between 0.49 m and 0.9 m; and between 0.9 m and 2 m. The first pattern corresponds to an error in the cube detection. When the

cube was purchased, it was transparent. To keep the results consistent and to contrast with the background used, the cube was painted white. Cardboard was used as a paint base and unfortunately some of it stuck to the cube when it dried, generating a detection error when the camera was close to the cube, as shown in Figure 6.9, highlighted in red. In addition, lines in the background were also detected, and caused detection errors. However, when the camera was far enough away from the cube (cube out of focus), the image started to blur and the carboard was no longer detected as illustrated in Figure 6.10.

The second pattern seems to represent the real answer. However, 10% - 20% of error is observable. According to Figure 6.10, it seems that this error comes from the detection of the background elements due to a lack of contrast as illustrated in Figure 6.9.

Figure 6.9: Cube with cardboard underneath

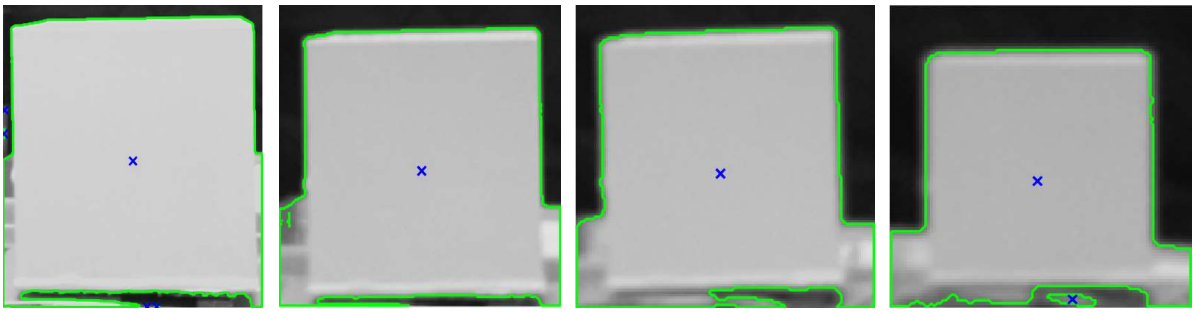


Figure 6.10: Detection of the cube at 0.5 m, 0.9 m, 1.33 m, and 1.73 m

Cube of 0.2 m edges (Medium cube)

The results for the 0.2 m cube are presented in Figure 6.11, and Table 6.3. The x-axis scale represents the distance between the camera position and the circles, and the y-axis scale is the mean value of the deviation on the height and width of the cube. The standard deviation on the cube height was 1.83 mm, and on the cube, width was 2.79 mm.

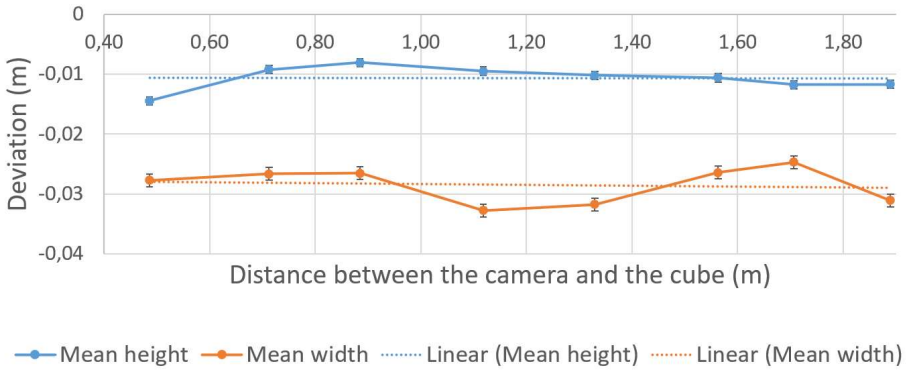


Figure 6.11: Deviation on the height and width of the 0.2 m cube

Table 6.3: Dimensions deviation of a 0.2 m cube for on object distance between 0.5 m and 2 m

Object distance	**0.50 m**	**0.90 m**	**1.33 m**	**1.71 m**
Height (mm)	-14	-8.1	-10	-12
Width (mm)	-28	-27	-32	-25

Regarding Figure 6.11 and Table 6.3, the data demonstrates that for both answers, the deviation on the cube dimensions increases with increasing of the object distance. However, the height and the width are not detected similarly. In addition, variations in both answers are observable.

These detection problems are caused by inconsistent detection of the height and width of the cubes in the image due to background elements, as shown in Figure 6.12 and Figure 6.13. The line defining the bottom of the cube, created during the painting process, and the edges of the rectangle used as a support were detected (circled in red in Figure 6.12), and were considered as part of the width of the cube by the MATLAB algorithm. However, the impact of the background elements is limited by the cropping process. Furthermore, as shown in Figure 6.13, when the camera was closest, the details were clearer and the bottom line of the cube was detected.

In addition, the 5 % to 10 % of error of the cube dimensions come from the same source of error (background elements) as for the 0.1 m cube analysis. However, the error contributions are smaller than in the 0.1 m cube case due to the size of the cube, hiding some of the background elements visible in the 0.1 m cube image.

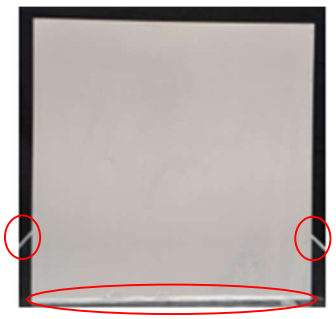


Figure 6.12: Medium cube

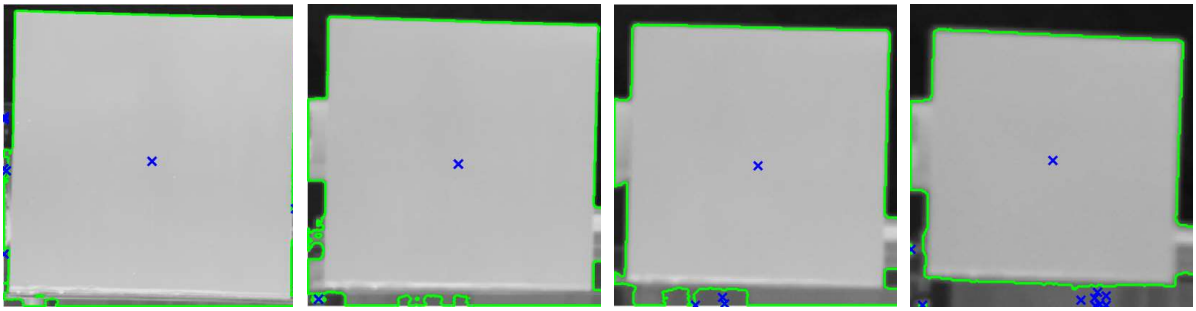


Figure 6.13: Detection of the cube at 0.5 m, 0.9 m, 1.33 m, and 1.73 m

6.2.1.2. Blender objects analysis

Virtual circles

The results from the virtual measurement of the 0.09 m and 0.07 m circles are shown in Figure 6.14 demonstrating the deviation of the camera measured circle diameters from the actual diameter. The mean values of the real answers were also plotted for comparison. The x-axis scale represents the distance between the camera position and the circles, and the y-axis scale is the difference between the theoretical and measured diameter observable.

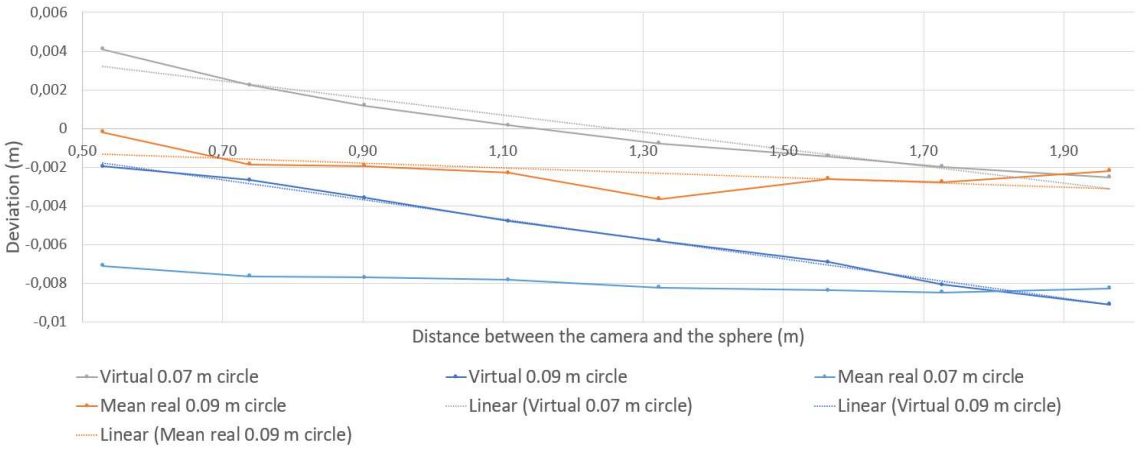


Figure 6.14: Deviation on the diameter of circles

According to Figure 6.14, the virtual results demonstrate similar results for both circles a difference of approximately 5 mm. In both cases, the error increases with increasing object distance. In addition, the real and virtual answers are different but a persistent gap between real-real, and virtual-virtual 0.07 m and 0.09 m circles is present of approximately 6 mm.

For both real and virtual data, the shape of the curve is determined by the camera parameters used. As explained in Chapter 5, the depth of field and the shape of the aperture (blades) of the camera have an impact on the slope of the results curve. Furthermore, the real and virtual data behave in the same way as the results of the sphere experiment in Chapter 5, (the virtual and real data were almost the same around 1.70 m as observable in Figure 6.14 of this experiment) which shows that the definition of the virtual camera model designed leaves a distinct "mark" on the data.

Regarding the virtual data only, and as shown in Figure 6.6, the 5 mm difference between the answers will also be a function of the pixel resolution of the image, as well as the pixel density of the circle in the image. These two parameters are directly related to the size of the circles, and the camera chosen.

Virtual cube of 0.1 m edges (Small cube)

The results for the virtual 0.1 m cube are presented in Figure 6.15. As in the circle experiment, the mean of the real data, with a standard deviation of -4.05 mm on the cube height, and -5.43 mm on the cube width; was plotted on the same graph as the virtual ones for comparison purposes. The x-axis scale represents the object distance, and the y-axis scale is the deviation on the height and the width of the cube.

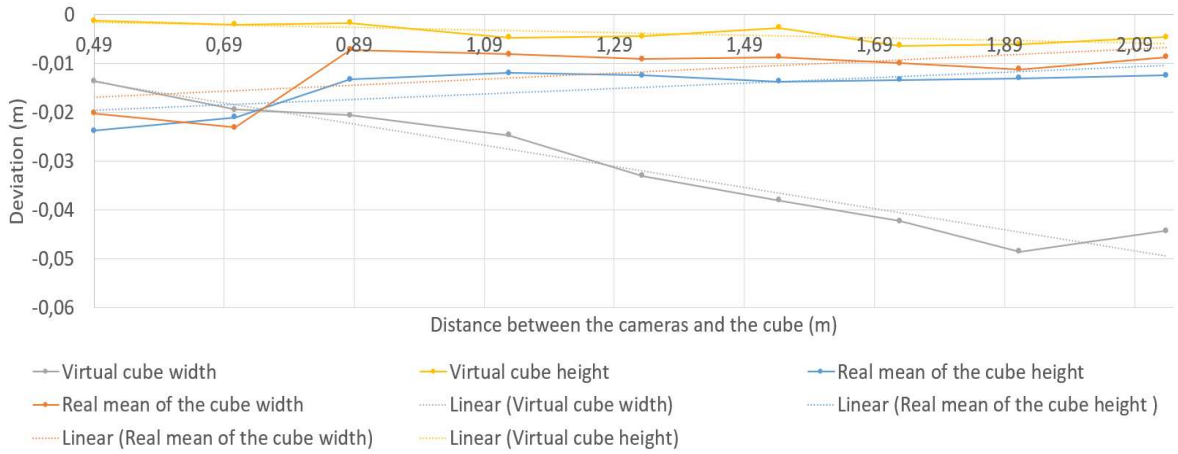


Figure 6.15: Deviation on the height and width of the 0.1 m edges cube

Table 6.4: Dimension deviation of a 0.1 m cube for an object distance between 0.5 m and 2 m

Object distance	**0.50 m**	**0.90 m**	**1.33 m**	**1.71 m**
Height (mm)	-1.2	-1.7	-4.4	-6.3
Width (mm)	-14	-21	-33	-42

Regarding to Figure 6.15 and Table 6.4, the virtual height and width of the cube are not detected similarly. It is noted that the same trend was observable for the real data in Figure 6.8. In addition, the deviation on both dimensions of the virtual cube seems to regularly increase with increasing object distance.

As explained in the 0.1 m real cube experiment and illustrated in Figure 6.16, the deviation on the width of the cube comes from the same source of error as in the real experiment, e.g., the detection of background elements.

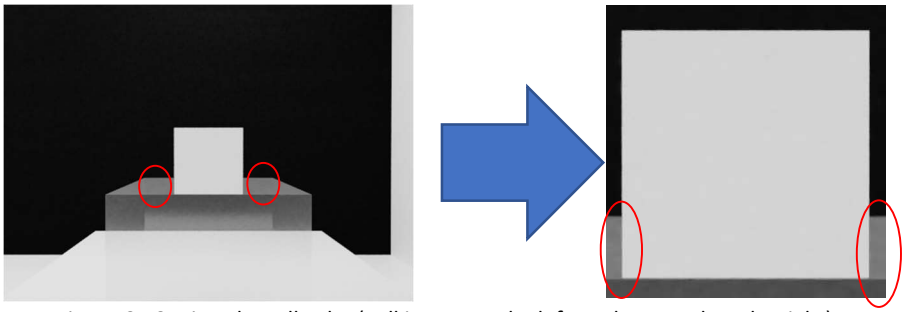


Figure 6.16: Virtual small cube (Full image on the left, and cropped on the right)

According to Figure 6.15, Tables 6.2 and 6.4, the real and virtual answers are different. This difference comes from the real and virtual set up. In the virtual set up, all the background objects presented in the real set up were not simulated removing the error linked to the contrast problem. In addition, the excess paint and detritus on the bottom side of the cube was not generated in Blender.

However, both setups do not give the real answer of the cube due to the noise generated by the background. In Blender, the same experiment was completed a second time, but by designing a perfect white diffuse 0.1 m cube, on black background, with all other objects removed. The results are shown in Figure 6.17.

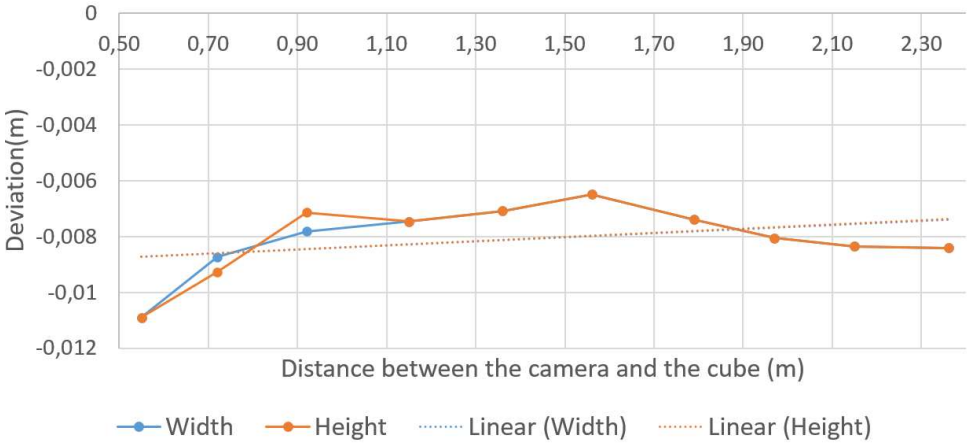


Figure 6.17: Deviation on the perfect 0.4 m cube simulated in Blender

The variation of the width and height of the cube have similar same trends with an overall variation of 0.004 m, except between 0.5 m and 1.10 m. The variations on the two dimensions of the cube are due to the pixel resolution affecting the grey scale, and the detection as shown in Figure 6.18 on the distance of the object between 0.5 m and 1.73 m. Figure 6.18 shows the variation of the grey scale over the width and height of the cube. These variations are accentuated (blurred) across the width of the cube as the distance to the object increases, thus generating an error.

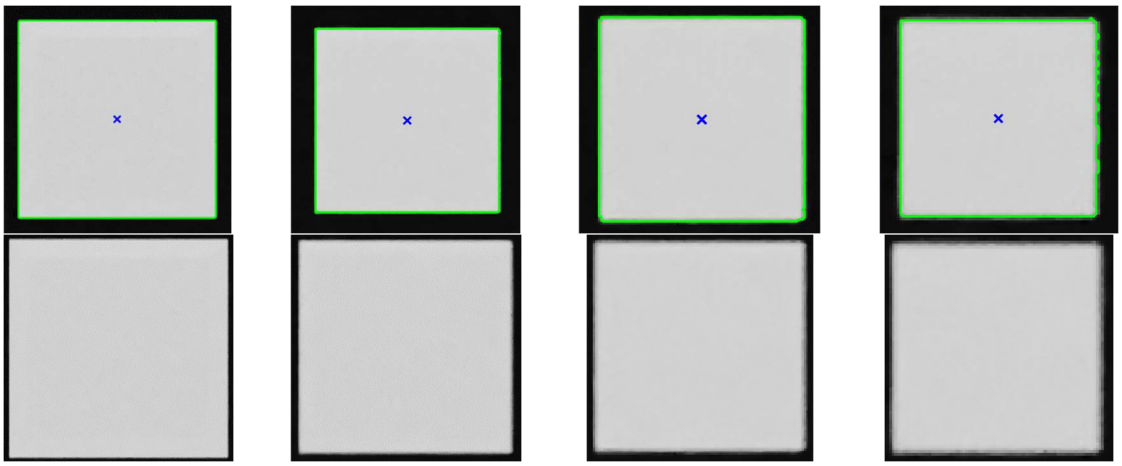


Figure 6.18: Detection of the 0.1 m perfect cube over 0.5 m, 0.7 m, 1.13 m and 170 m
First line: Detected cube
Second line: The virtual image of the cube

Virtual cube of 0.2 m edges (Medium cube)

The results for the virtual 0.2 m cube are presented in Figure 6.19, with the real mean answers.

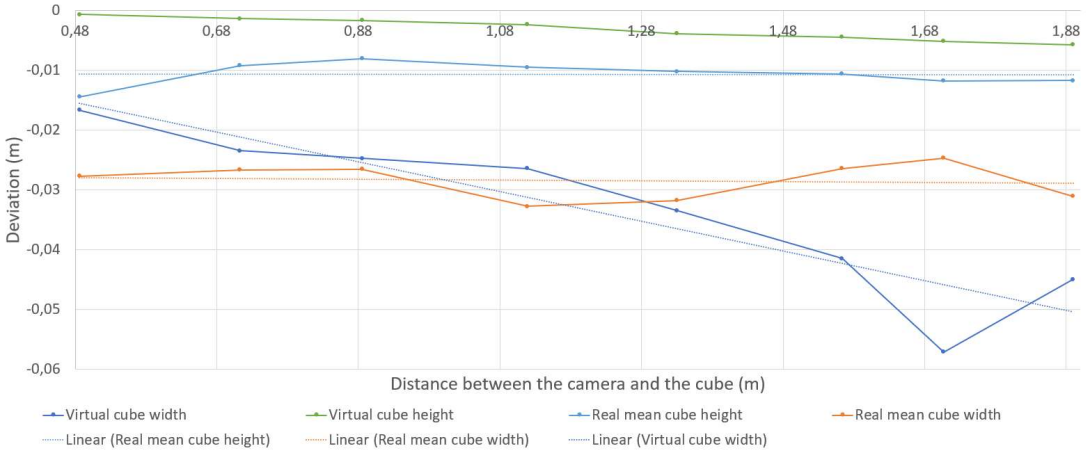


Figure 6.19: Deviation on the height and width of the 0.2 m edges cube

Table 6.5: Deviation on the dimensions of a cube with 0.2 m edges for a distance camera-cube between 0.2 m and 2 m

Object distance	**0.50 m**	**0.90 m**	**1.33 m**	**1.71 m**
Height (mm)	-0.7	-1.7	-3.9	-5.2
Width (mm)	-17	-25	-33	-57

According to Figure 6.19 and Table 6.5, the virtual height and width of the cube are not detected similarly as noticed with the real results on the same cube, and the results for the 0.1 m cube in both environments. In addition, the deviation on both virtual cube dimensions regularly increases with increasing of the object distance. It is also noticeable that the same behaviour at 1.68 m can be observed in the response to the width of the real and virtual cube. As a reminder, this behaviour at 1.68 m is due to a cube width detection error caused by background noise (illustrated in Figure 6.12).

As explained in the virtual and real experiment of the 0.1 m cube, the difference between the virtual 0.2 m cube dimensions come from the detection of background elements of the scene (Figure 6.16). In addition, the real and virtual answers seem similar. It seems that the partial simulation of the scene elements in Blender, was composed in the real environment by the size of the cube hiding some of them. However, there is a discrepancy between the results of the two environments, which could be due to the total lack of similarity between the two set ups. These limitations come from the software, which cannot simulate everything, and from the identification of sources of error in the real environment.

6.2.2.Camera location relative to a calibration artefact: Checkerboard

6.2.2.1. Real-world reprojection error analysis

The results from the real-world multiple camera measurement of the checkerboard are shown in Figure 6.20 and Figure 6.21, and the experimentation was repeated three times. The x-axis represents the object distance, and the y-axis is the reprojection error of the board.

Two sets of data are presented for each experiment due to the utilisation of two separate checkerboards. The smaller one was used for an object distance varying between 0.4 m and 0.8 m (Figure 6.20), and the larger one was used for object distances varying between 1 m and 1.8 m (Figure 6.21).

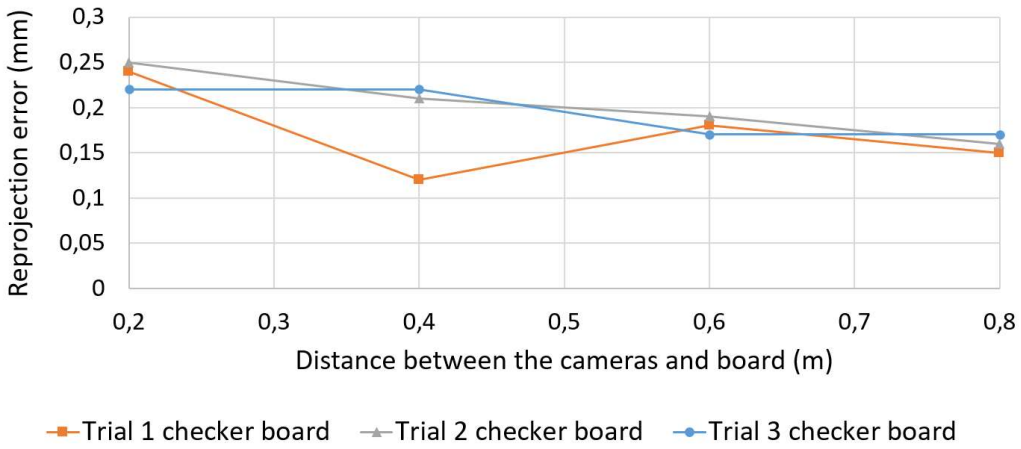


Figure 6.20: Real environment reprojection error (0.4 m to 0.8 m) - small

Table 6.6 Reprojection error on object distance of 0.4 m, 0.6 m and 0.8 m

Trials	0.4 m (mm)	0.6 m (mm)	0.8 m (mm)
1	0.12	0.18	0.15
2	0.21	0.19	0.16
3	0.22	0.17	0.17

According to Figure 6.20 and Table 6.6, all three trials have a similar answer, except at 0.4 m, with the presence of an aberration point in the trial 1 data. In addition, the variations between the three trials are approximately 0.01 mm – 0.02 mm. As explained in Section 4.5.2 of Chapter 4, the measurement

of the reprojection error is subject to different sources of error such as the incidence angle, or the variation of the illumination on the board.

The peak observable in Trial 1 at 0.4 m, will be caused by the predominant error sources, e.g., camera movement, misalignment with the board, angle of incidence and light making the board surface too bright to be detected. Otherwise, according to the small variations between the three trials it seems that the measurement process is robust.

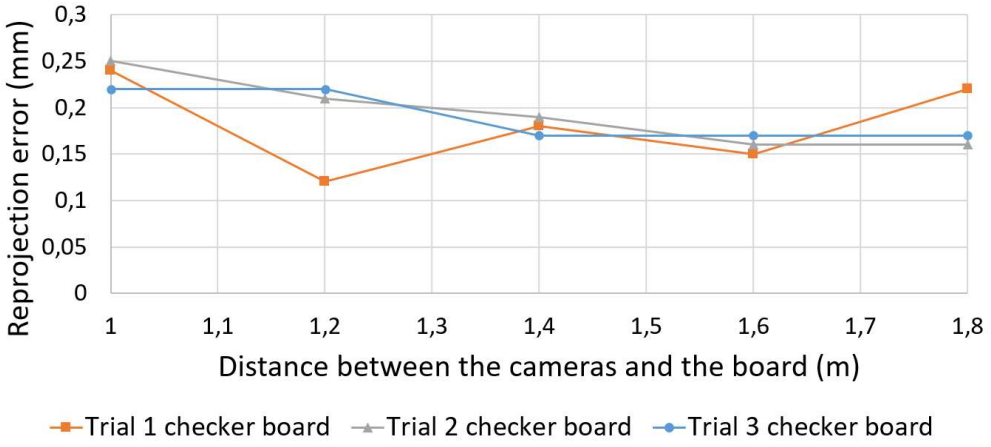


Figure 6.21: Real environment reprojection error (1 m to 1.8 m) - large

Table 6.7: Reprojection error on object distances of 1 m, 1.2 m, 1.4 m and 1.8 m

Trial	1 m (mm)	1.2 m (mm)	1.4 m (mm)	1.8 m (mm)
1	0.24	0.12	0.18	0.22
2	0.25	0.21	0.19	0.16
3	0.22	0.22	0.17	0.17

Regarding Figure 6.21 and Table 6.7, similar conclusions as regarding Figure 6.20 and Table 6.6 are drawn. Two aberrations points at 1.2 m and 1.8 m are also observable in Trial 1 data, caused by the sources of error listed in Figure 6.22.

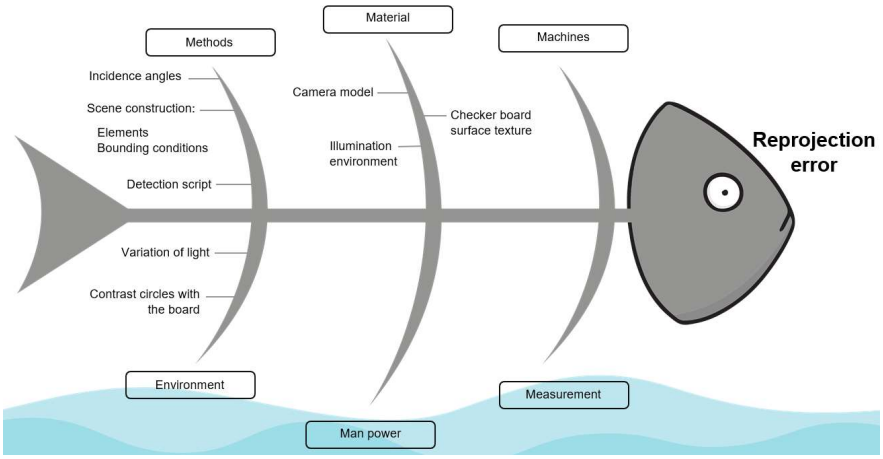


Figure 6.22: 5M diagram of the virtual set up

By comparing the small and large checkerboard results, it is noticeable that their answers are also identical. This identicality shows that the measurement process is robust, and the checkerboard is slightly sensitive to the sources of error listed in Figure 6.22, and less than the real zone circles board (results in Section 4.5.2 of Chapter 4) which has a reprojection error variations between 0 mm and 20 mm at an object distance ranging between 0.2 m and 2 m, with a step of 0.2 m. It seems that the checkerboard might be a better artefact than the zone circle board to determine the optimum camera positions.

6.2.2.2. Blender reprojection error analysis

The results obtained in Blender are shown in Figure 6.23, in a similar way to the real-world analysis in Figure 6.20 (short distance) and Figure 6.21 (long distance). However, the real and virtual trials are plotted together for comparison purposes and, due to the similarity of the real data, the mean was taken for the responses of both board sizes. The standard deviation of each trial for each board size are shown in Table 6.8.

Table 6.8: Standard deviation for each trial the trials for the small and large board answers

Standard deviation	Trial 1	Trial 2	Trial 3
Small board (mm)	0.04	0.03	0.03
Large board (mm)	0.007	0.14	0.39

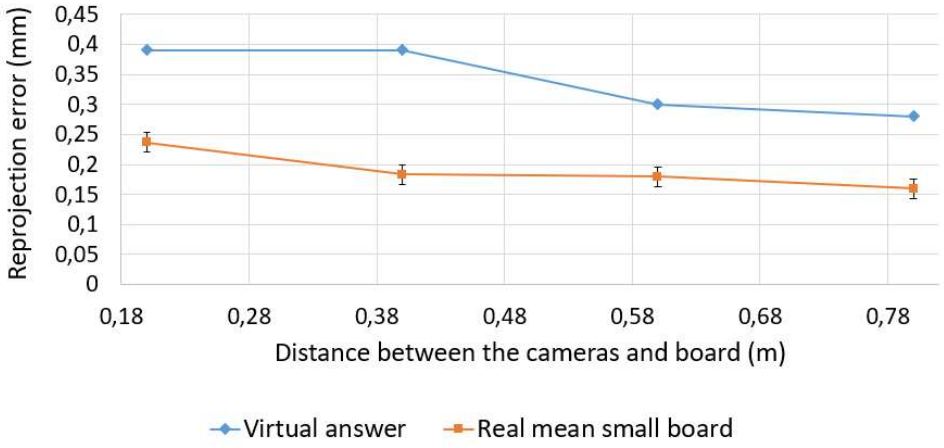


Figure 6.23: Virtual environment reprojection error (0.4 m to 0.8 m)

Table 6.9: Reprojection error on object distances of 0.4 m, 0.6 m and 0.8 m

Configuration	0.4 m (mm)	0.6 m (mm)	0.8 m (mm)
Mean real	0.18	0.18	0.16
Virtual	0.39	0.30	0.28

Regarding Figure 6.23, and Table 6.9, the mean of the real answer and the virtual answer have similar trend, e.g., the reprojection error decreases with increasing object distance. However, the virtual answer is worse than the mean of the real answer.

As explained in Chapter 5, in Sections 5.2.2.6 and 5.4, the virtual and real set up are different due to the object surface textures definition, and the definition of the camera models. These two major differences cause the “gap” observable in Figure 6.23. In addition, the virtual surface texture of the board is a pdf image. As explained in Section 6.1.2, the location of this surface texture on the top surface of the map may cause errors. In this experiment, a checkerboard pattern is used. However, due to the adjustment of the image texture on the map surface, the squares may be distorted and become rectangular, resulting in a detection error in MATLAB.

Regarding to Figure 6.24, and Table 6.10, similar conclusions as those regarding Figure 6.23 and Table 6.9 are drawn. In addition, the comparison of both Figures (Figure 6.23 and Figure 6.24) and Tables (6.9 and 6.10) shows that the reprojection error in the virtual environment is higher for the smaller board than for the larger board, while the reprojection error is similar for both board sizes in the real environment.

In addition, an almost constant trend can be observed for both responses from 0.98 m to 1.18 m, with a peak at 1.78 for both responses. This peak is due to a degradation in the detection of the checkerboard linked to the resolution of the pixels (due to the distance between the camera and the checkerboard), and perhaps the distortion of the board squares because the checkerboard was at the upper limit of the camera's field of view.

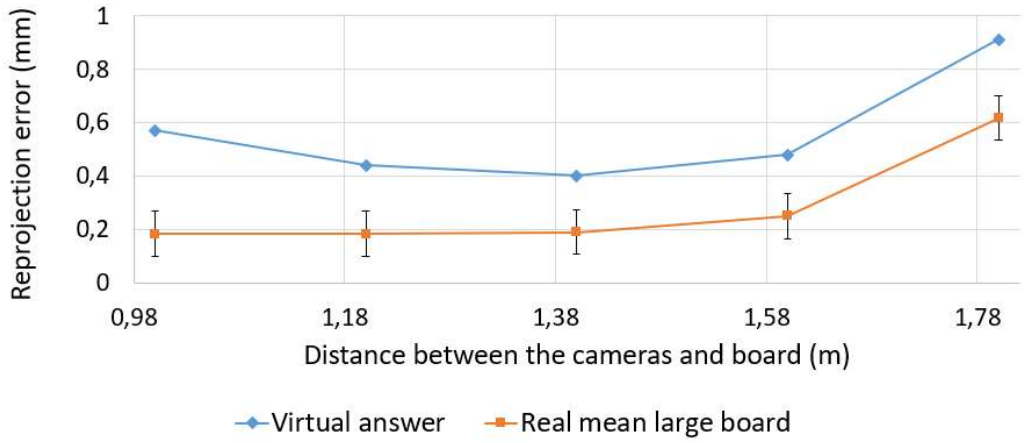


Figure 6.24: Virtual environment reprojection error (1 m to 1.8 m)

Table 6.10: Reprojection error on object distances of 1 m, 1.2 m and 1.8 m

Configuration	1 m (mm)	1.2 m (mm)	1.4 m (mm)	1.8 (mm)
Mean real	0.18	0.18	0.19	0.61
Virtual	0.57	0.40	0.40	0.91

Both virtual boards are subjected to the same sources of error, described in 5M chart, in Figure 6.22. According to the error analysis, the only elements modified in the virtual set up was the board (moving from the smaller to the larger board). As previously explained (Figure 6.23 analysis, and Section 6.1.2), it seems that the location of the surface pattern on the checkerboard top offers a larger contribution to the global error for the larger board than the smaller board. This source of error seems to impact the shape of the square on the checkerboard pattern, making them more difficult to be detected properly.

6.2.3. Camera location relative to a calibration artefact: Zone circle board

6.2.3.1. Real-world reprojection error analysis

Experiments were completed in Section 4.4 of Chapter 4, in both environments to investigate if a 3D virtual reality modelling environment such as Blender can be used to fully model camera-based metrology systems and help in the camera placement related to a calibration artefact. However, due to the lack of "realism", and the not fully defining the modelling conditions, the results from both set ups were quite different.

At this stage, in this Chapter, the aim of this experiment was to:

1. Characterise the answer of the photorealistic virtual environment in the simulations of simple (circles, spheres, and cubes) and more complex objects (checkboard, and zone circle boards).
2. Validate the digital model generated.
3. Quantify the difference between the real world and its virtual replications in simple and complex object simulations.

In addition, a new virtual setup was used, developed in Section 5.4.1, of Chapter 5, and defined as photorealistic. Nevertheless, due to the use of the same real set-up as in Section 4.4.1 of Chapter 4, the real results are not shown again, but the main conclusions of this experiment are recalled for a better understanding of the comparison with the new virtual set-up:

- The reprojection error for the small board was between 0 mm and 13 mm; while it was between 0 mm and 20 mm for the larger board.
- The reprojection error was larger for the larger board than the small board.
- The reprojection error was larger for zone circles 1, 2, 4 and 5, whilst zone circle 3 had the smallest value. This difference was attributed to the fact that the zone circles were paired (1 with 4, 2 with 5) but 3 is an individual zone circle.
- The circles were "ovalized", changing slightly the centre coordinates and the height of the board due to the angle of incidence used and the increase of the object distance.
- The sources of error in Figure 4.28, Chapter 4 were identified as the major sources of error of this experiment.

6.2.3.2. Blender reprojection error analysis – zone circles

The analysis of the reprojection errors determined within the Blender virtual environment is presented in a similar fashion to the checkerboard experiment in Section 6.2.3.1.

Data presented in Figure 6.25 and Table 6.11 show that the reprojection error decreases between 0.4 m and 0.6 m, before increasing between 0.6 m and 0.8 m, and the reprojection error is larger for circles 1, 2, 4 and 5, than for circle 3.

Regarding to the conclusions of the real experiment, similar conclusions are drawn from Figure 6.25, and Table 6.11. Circle 3 has the best reprojection error due to its localisation on the board. However, the reprojection is between 0 mm and 6.34 mm, while it was between 0 mm and 8 mm for the real answer, and between 0 mm and 5 mm for the virtual answer presented in Section 4.5.2 of Chapter 4, with the basic virtual set up.

The new virtual set up has a similar answer than the old virtual set up with this artefact. Regarding the conclusions of Chapters 4 and 5, it seems that the photorealistic approach has not impacted on the result of this experiment, and the real and virtual environments are intrinsically different in their definition.

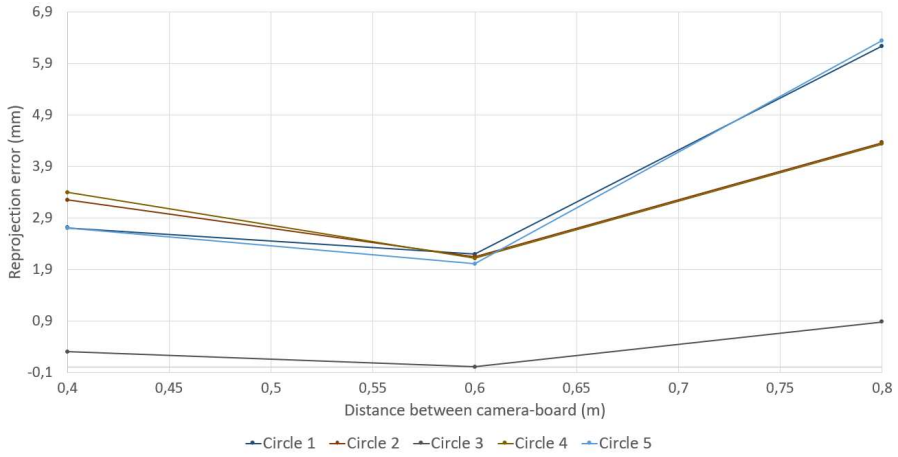


Figure 6.25: Virtual environment reprojection error (0.4 m to 0.8 m)

Table 6.11: Real-world reprojection error on object distances between 0.4 m and 1.8 m

Small board	**0.4 m (mm)**	**0.6 m (mm)**	**0.8 m (mm)**
1	2.71	2.19	6.23
2	3.25	2.15	4.36
3	0.30	0.006	0.88
4	3.40	2.12	4.33
5	2.70	2.00	6.34
Large board	**1.2 m (mm)**	**1.4 m (mm)**	**1.6 m (mm)**
1	0.94	0.2	0.68
2	0.93	0.21	0.68
3	0.043	0.09	0.012
4	0.89	0.12	0.6
5	0.76	0.17	0.75

Data presented in Figure 6.26 and Table 6.11 highlight the reprojection error decreases between 1 m and 1.4 m, before increasing between 1.4 m and 1.6 m, and decreasing again between 1.6 m and 1.8 m, and the reprojection error was larger for circle 1, 2, 4 and 5, than for circle 3.

Regarding to the conclusions of the real experiment similar conclusions are drawn from Figure 6.26, and Table 6.11. However, here the reprojection is between 0 mm and 1 mm, while it was between 0 mm and 20 mm for the real answer, and between 0 mm and 2 mm for the virtual answer presented in Section 4.5.2 of Chapter 4, with the basic virtual set up.

As concluded previously with Figure 6.25, it seems that the photorealism approach has not changed the virtual answer.

However, during that experiment with the zone circles board, the real image and image texture did not cover the object surface exactly, resulting in a change in the coordinates of the area circle centres, which led to incorrect detection and inflated reprojection error terms. It is therefore difficult to establish a direct correlation between the texture of the modelled zone circle board and the real-world numerical values (as for the sphere in the diameter measurement scenario).

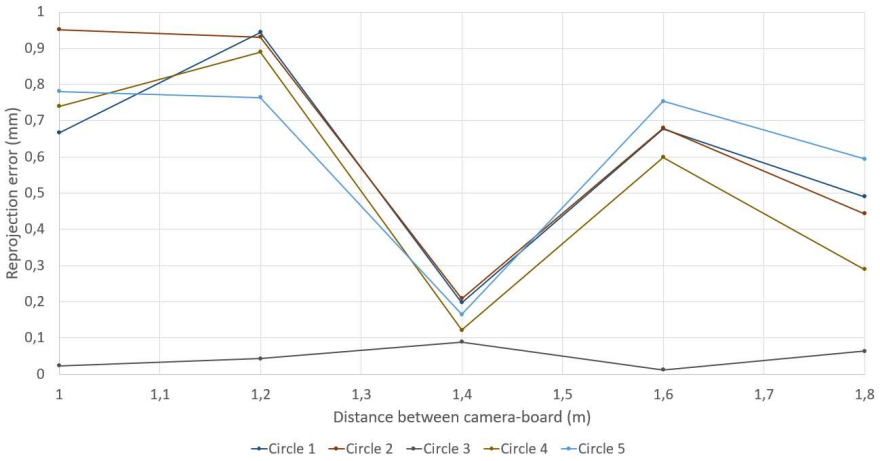


Figure 6.26: Virtual environment reprojection error (1 m to 1.8 m)

6.2.3.3. Texture and floor colour analysis in Blender

Methodology

To understand the impact of this new source of error linked to the texture position on the board, the reprojection error of the zone circles board was measured and compared (only in Blender), in three virtual scenarios where the texture of the zone circle board was:

- An image texture (pdf image of the real board texture generated on the manufacturer website [286]).
- A fully rebuilt virtual texture in Blender (Table 6.12).

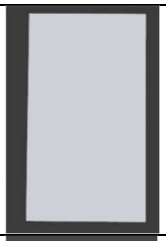
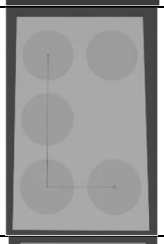
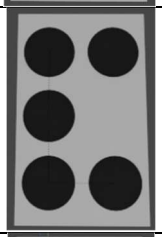
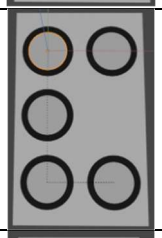
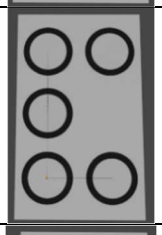
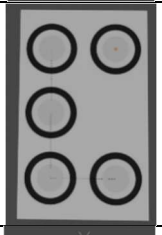
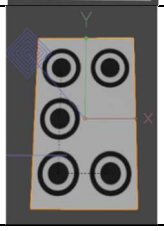
However, to have a deeper understanding and analysis of the impact of the virtual texture on the measurement, the dimensions of the circles on the image texture, and the real texture were used. Two fully rebuilt texture images were used, one with the dimensions of the image texture circles, measured in Blender with the tool ruler; and one with the dimensions of the real image texture circles, measured in reality with a vernier calliper.

Due to the adjustment of the image texture to the rectangular shape of the board, the circles have a diamond shape (called lozenge here), whilst the circles on the real pattern, have a circular shape.

This test was run to determine the impact of the surface texture, and the shape of the circles (circular or lozenge) on the measurement. The board dimensions were those of the large area circle board, and the distance from the object was set to 1.8 m, in order to have a larger field of view and a better image of the board, and the reprojection error was calculated with the same MATLAB set of algorithms as used in Section 3.3.2 of Chapter 3, for all the boards used in the real and virtual environment.

The recreation of the pattern of the zone circles board was achieved by using various Blender modelling tools, and each step of the process are illustrated in Table 6.12.

Table 6.12: Steps of creation of the new board in Blender 2.92

Step	Visualisation
Creation of the base board with a white texture on it.	
Creation of the large circles, positioned with the coordinates in Table 3.1 of Chapter 3, Section 3.1.	
Apply a black texture to the new circles.	
Creation of the medium circles, positioned with the coordinates in Table 3.1 of Chapter 3, Section 3.1, on top of the black circles.	
Apply a white texture to the new circles.	
Creation of the small circles, positioned with the coordinates in Table 3.1 of Chapter 3, Section 3.1.	
Apply a black texture to the new circles.	

The dimensions of the circles corresponded to the actual dimensions. However, due to the creation process, three circles (two black and one white) were created. The circles were separated by a distance of 0.001 m along the z-axis, to avoid image noise generation and to keep a high contrast between the white and black colours, as shown in Figure 6.27. The separation distance of 0.001 m was chosen by a process of trial and error and is the smallest value found to allow a flat board to be seen by the cameras at any angle of incidence. Figure 6.28 shows an example where the separation distance was 0.001 m, and 0.01 m.

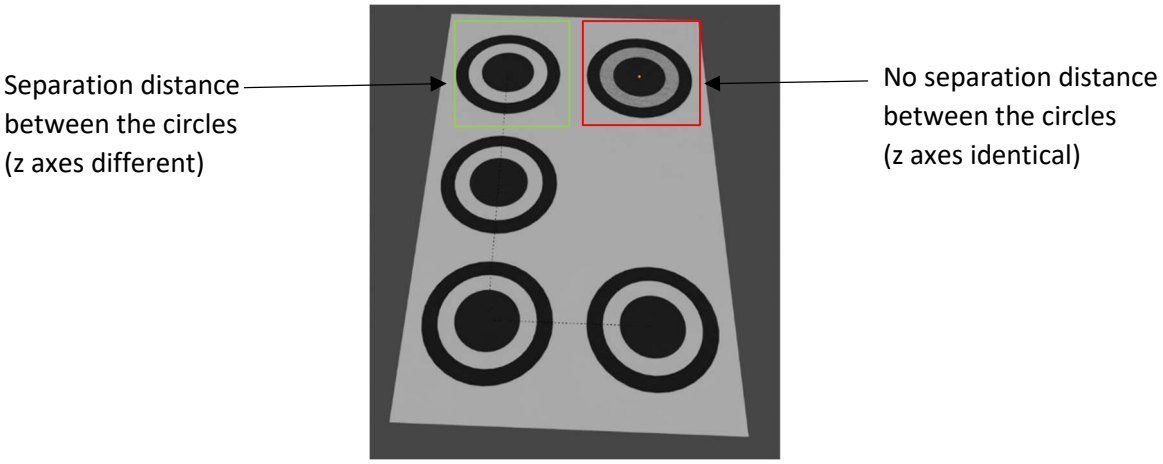


Figure 6.27: Contrast problem linked to the separation distances between the circles

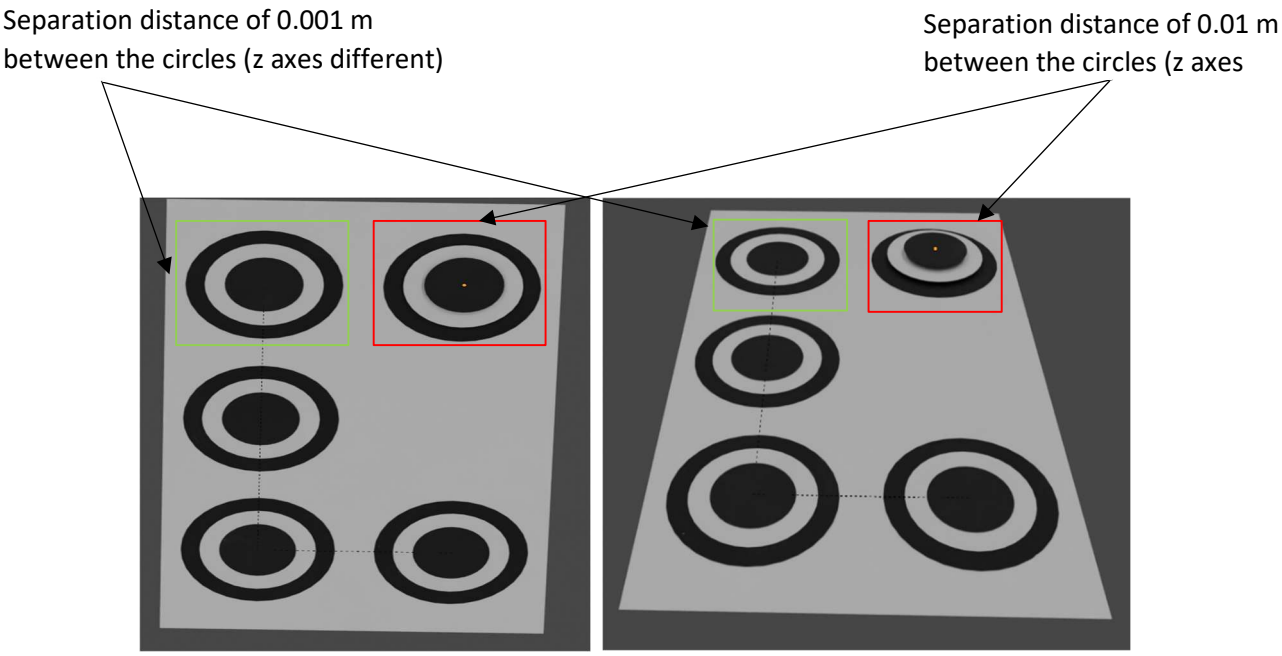


Figure 6.28: Detection problem linked to the separation distances between the circles

In addition, with the identification of the error generated by a wrong alignment of the image texture position on the board surface, questions about the impact of the virtual room floor surface texture on the circle detection (contrast between the floor and the board) were considered. In this experiment, in Blender, the floor surface texture was generated by using colour map, roughness map, and normal map found in online Blender texture libraries [287] such as Poly Haven [288] and Poliigon [289].

To quantify the impact of this possible source of error, the experiment was run only in Blender, with the following floor surface textures:

- A grey colour.
- A grey colour with surface defaults (scratches).
- An RGB (Hex = 0B0B0C) texture colour with surface defaults.
- The texture floor used with surface defaults.

The zone circle board with an image used as surface texture and the fully reconstructed board were used for each floor surface texture.

Results

The results of the zone circle board pattern test are shown in Figure 6.29, while the result for the floor test shown in Figure 119. Noted that the results of the two tests are not overlapping, and a grey scale colour was used for the floor in the zone circle board pattern test.

In Figure 6.29, the board with an image texture is identified as "Texture", the board reconstructed with the size of the circles of the image texture as "Pattern rebuilt with texture size", and the board reconstructed with the actual size of the circles as "Pattern rebuilt with true size".

Regarding to Figure 6.29, similar observations are made about the zone circles detection as for the previous reprojection error experiments, e.g., circles 1, 2, 4 and 5 have a larger error than circle 3. In addition, the three answers show similar behaviours, but the reprojection error for the texture pattern (the pdf image) is smaller than for the two rebuilt boards, and the reprojection error for the rebuilt board with the real circles size has the large reprojection error. For instance, for reprojection error on the detection of circle 2 is at 0.44 m for the texture answer, 0.93 mm for the rebuilt board with the image texture circles' sizes (lozenge shape), and 1.54 mm for the rebuilt board with the real circles' sizes (circular shape).

The pdf image used as a surface texture on the virtual board gives the best results because it is a uniform texture, which can be rotated and scaled uniformly, resulting in a consistent scale and location of the circles constituted the pattern, which minimises the location error.

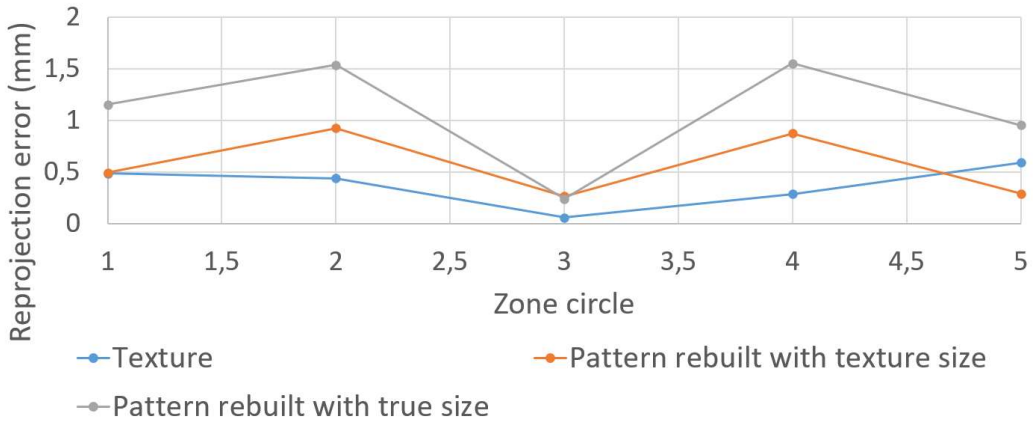


Figure 6.29: Zone circles pattern variation

The results of the reconstructed pattern with the size of the pdf image circles can be caused by the non-uniform propagation of location error, as each circle is created and placed manually on the board.

The results of the reconstructed pattern with the size of the real circles seem to give more realistic results, but still better than the real model (between 1 mm and 9 mm). The difference between this pattern and the other two could be also caused by the non-uniform distribution of the location error. However, this location error impacts both rebuilt patterns. The difference between them comes from the shape of the circles. As explained previously, the shape of the circles is slightly different between the two patterns, one is lozenge, the other is circular, and it seems that this difference impacts the circle detection, and consequently the reprojection error.

Regarding Figure 6.30, the same trend in the circle detection is observable as previously described for the other experiments (circle 3 has the best reprojection error). In addition, the answer given from the grey colour floor with scratches and the rebuilt pattern configuration gives the most realistic answer, because its reprojection error is between 0 mm and 6 mm as the real answer. This result (brown line in Figure 6.30) seems to be due to the texture of the board. This observation is made by comparing the grey colour with the response to scratches (dark blue and brown lines in Figure 6.30) for the two board textures. In comparison, the other answers with this board have a reprojection error between 0 mm and 2 mm.

When an image texture is used instead of the rebuilt pattern for the floor surface texture variations, two trends are observable. For the floor surface texture configurations defined as a grey colour floor with and without scratches, and with the RGB colour floor with scratches, the results have a reprojection error between 0 mm and 2 mm, while the others have a deviation between 0 mm and 0.8 mm.

In this work, the colour of the floor has an impact on the results, but not the defaults (here scratches) of the surface. The different results obtained for the grey colour and the RGB colour could be due to their pigment richness, generating two different greyscale images in MATLAB, during the detection process. In addition, the type of texture applied (image texture or reconstruction texture) on the board surface has an impact on the results, due to the non-uniform distribution of the localisation error, as illustrated in Figure 6.29.

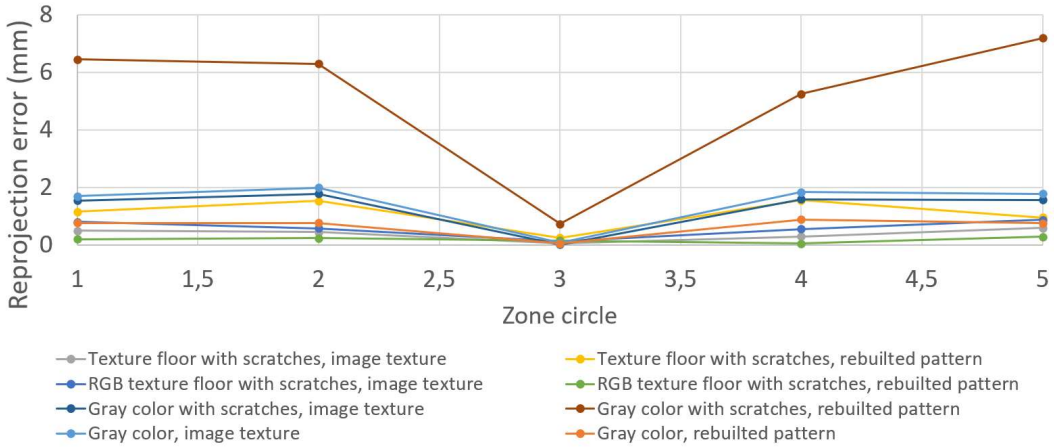


Figure 6.30: Floor texture variations

6.2.4.Sphere benchmark

As explained in Section 6.1.3, this experiment consists of measuring the diameter of 0.09 m, 0.15 m, 0.2 m, 0.25 m and 0.3 m diameters sphere with a single camera in Blender to design a sphere benchmark, to perform the virtual prediction of the evolution of the deviation on the diameter of the spheres related to the increase of the camera-sphere distance. Because the 0.15 m and 0.25 m spheres were not available on the manufacturer website, only the 0.09 m, 0.2 m and 0.3 m diameters spheres were measured in the real world.

The results are shown in Figure 6.31, where the x-axis scale represents the distance between the camera and object in metres, and the y-axis scale is the deviation of the sphere diameter in metres.

According to Figure 6.31, the deviations of the virtual sphere diameters describe similar patterns, and converge to an error of approximately -5 mm, with increasing of the object distance, even if their starting point is different. In addition, the geometrical error observable in the sphere experiment of Section 4.5.1 of Chapter 4 is also noticeable for the 0.2 m, 0.25 m, and 0.3 m sphere diameters, as observed in the first few data points.

The real answers describe all the same data behaviours, similar to the virtual answers, but the 0.2 m and 0.3 m diameter spheres are parallel to the 0.09 m data with diameter deviations of approximately 5 mm. In addition, the 0.09 m answer is the same magnitude of error than the virtual answers.

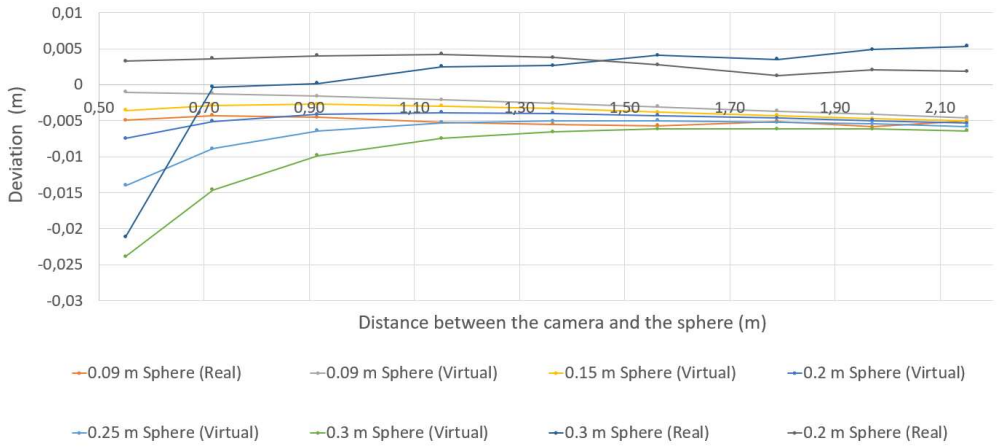


Figure 6.31: Sphere benchmark

According to Table 6.13, the differences in the diameters of the real and virtual spheres are different. The virtual responses are better than the real ones. However, the two responses suggest that an offset exists between the real and virtual scenarios. This offset most probably comes from the difference between the two set-up defined in Section 5.2.2 of Chapter 5. In addition, this offset can potentially be used to correct the predictions of the virtual answers in order to obtain the real answers.

Similar behaviours were noticed in the measurement of object dimensions in Section 6.2.1, suggesting that the diameter deviation differences between each sphere are related to the dimensions of the object. For example, a virtual sphere of 0.09 m has a diameter deviation of -2.1 mm at 1.15 m, while a diameter sphere of 0.2 m has a deviation of -3.9 mm.

Table 6.13: Comparison real and virtual answers on object distances of 0.5 m, 0.7 m, 1.15 m and 1.79 m

Object distance	0.5 m (mm)	0.7 m (mm)	1.15 m (mm)	1.79 (mm)
0.09 m sphere real	-5.0	-4.3	-5.2	-5.0
0.09 m sphere virtual	-1.0	-1.2	-2.1	-3.7
0.2 m sphere real	3.2	3.6	5.2	1.3
0.2 m sphere virtual	-7.5	-5.1	-3.9	-4.7
0.3 m sphere real	-21	-0.3	2.5	3.5
0.3 m sphere virtual	-24	-15	-7.5	-6.1

6.3. Discussion

In this Chapter, the robustness of the virtual model designed in Chapter 5 has been investigated. Experiments were considered to reach this aim, as well as to evaluate the potential of this model to simulate the real-world behaviours, to investigate new potential sources of errors, and characterise the possible differences between the virtual and real scenarios.

The first experiment was the measurement of the diameters of simple objects (circles, and cubes) with a single camera in the real world and virtual environment; the second experiment was the measurement of camera calibration artefacts used for multiple camera positions (in this case eight cameras); and the last experiment, the creating of a sphere benchmark by measuring the diameters of five different spheres in Blender, and to reproduce this measurement for three of them in the real world.

With respect to the object experiments, the virtual and real-world responses are different but have similar trends. In the circle experiments, the trend of the virtual answer was attributed to the virtual camera depth of focus, the pixel density and the camera resolution. In the cubes experiment, noise coming from the cube preparation, the background, and the object contrast were identified as the primary sources of error in the real-world measurement of the cube height. This cube preparation error was not simulated in Blender, partially explaining why the two answers are different.

Through the measurement of the diameters of simple objects, it was noticed that the camera parameters, as well as the environment set up, were important parameters, modifying the virtual environment, and its results. These two parameters can be improved by a better characterisation, and a possible update of the software capacities.

In the camera location and calibration experiments, the results from the virtual and real experimentations are likewise different. The real reprojection error results are between 0.12 mm and 0.24 mm for the checkerboards, and between 0.2 mm and 20 mm for the zone circle board; whilst the virtual results are between 0.28 mm and 0.91 mm for the checkerboard, and between 0.004 mm and 7 mm for the zone circles. It seems that the virtual model performs better than the real one, even after adding “realistic” parameters. In addition, the comparison between the reprojection error experiment in Chapters 4 and 6 show that the photorealism approach has limited impact the virtual answer.

Through this experiment, it was also concluded that the reprojection error is worse with the zone circle, than with the checkerboard. Circles are easily disturbed by distortion from the camera or the angle of incidence, making them oval, while squares are less sensitive due to their geometry [290]. In addition, the geometric coordinates of the centre of the circle in Blender may be different from the actual coordinates due to the way the texture is applied to the board. The checkerboard therefore seems to be a more reliable calibration artefact than the area circle board.

In the zone circles surface texture experiment, it was observable that the utilisation of an image to simulate the surface texture gave the best results, and the rebuilt of the zone circles patterns with the real circles sizes the worse, but still better than the real results. It was concluded that the difference in the results was coming from the spreading of the circle location error, more uniform across the pdf image than the manually location of the circles on the board surface.

For the floor colour experiment, similar conclusions about the zone circles texture experiment was found about the board surface texture, and it was also concluded that the grey colour gave more realistic results than a RGB colour due to the greyscale.

In Blender, textures are not defined as function of physics but as a response cinematography. This introduces errors, because it is very difficult to use the real measured surface value with the only option being to try to reproduce the texture by “playing” with the software options available.

Finally, in the sphere diameters benchmark, it was highlighted that the difference between the real and virtual answers comes from the size of the sphere, and the differences between the real environment and its virtual replication had generated an offset that might be used to correct the virtual answer predictions.

6.4. Conclusion

In this chapter, the robustness of the virtual model designed in Chapter 5 has been investigated, through three experiments; the measurement of the dimensions of simple objects (circles and cubes) with a single camera; the measurement of camera calibration artefacts used in the case of multiple camera positions (in this case eight cameras); and the creation of a sphere benchmark with a single camera.

In these experiments, the following 5M chart was drawn as a summary of the principal sources of errors impacted the virtual results:

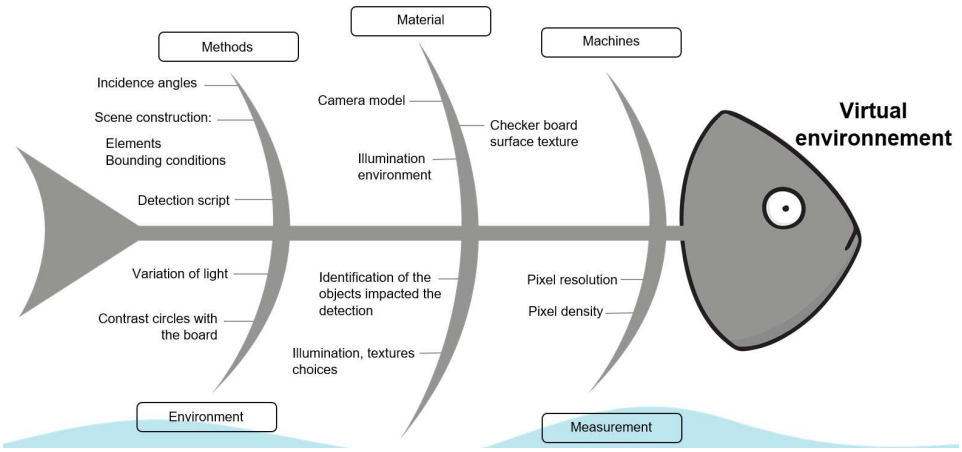


Figure 6.32: 5M chart of the principal sources of error in the virtual environment

The error sources identified in Chapters 4 and 5 were also observable in this Chapter, and it seems that the photorealism approach had limited impact on the new virtual result for the reprojection error.

Thanks to the different experiments presented in this chapter, the same trend seems to be observed between the real and virtual results, as well as a systematic difference between them. This systematic difference has been identified as a function of the dimensions of the object.

To study this systematic difference, a subtraction between the real and virtual answers was completed for the spheres, the circles and the cubes. Noted that most normally, this subtraction is inadvisable

because both environments have different sources of error, but this will be taken into consideration during the analysis.

The results of this subtraction for spheres and circles are shown in Figure 6.33, while the results for cubes are shown in Figure 6.34. In both Figures, the x-axis represents the distance to the object and the y-axis the difference between the real and virtual responses. The actual responses selected are the mean responses.

According to Figure 6.33 and Table 6.14, all curves describe a similar, almost linear trend, but their starting point are different, implying that the diameter of the object could be responsible for this discrepancy. The conclusions of Chapter 4 regarding the light effect, and the pixel resolution on the sphere detection are also observable. For instance, for the 0.3 m sphere diameter, the deviation is negatively increasing between 0.48 m and 0.88 m, before being constant. In the 0.3 m sphere diameter answer, a deviation of -0.015 m is observable at object distance of 0.68 m. The errors affected this measurement might come from the noise in both images (real and virtual) and from the difference in the camera models.

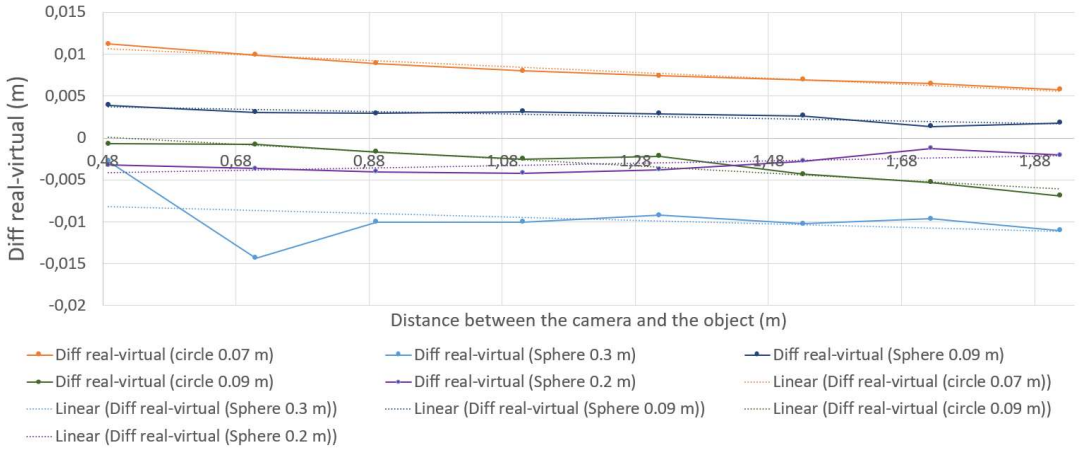


Figure 6.33: "Difference" between the real and virtual answers for the circle and sphere result

Table 6.14: Difference between the real and virtual answers for object distances of 0.49 m, 0.71 m, 1.32 m and 1.92 m

Object distance	0.49 m (mm)	0.71 m (mm)	1.32 m (mm)	1.92 m (mm)
Circle 0.07 m	11.2	9.90	7.34	5.76
Circle 0.09 m	-0.69	-0.79	-2.17	-6.91
Sphere 0.09 m	3.92	3.06	2.88	1.78
Sphere 0.2 m	-3.24	-3.64	-3.77	-2.05
Sphere 0.3 m	-2.74	-14.3	-9.21	-11.0

Regarding Figure 6.34 and Table 6.15 for the analysis of the cubes, the curves show a similar negatively increasing data trend with increasing object distance. Furthermore, the same conclusions than in Section 6.2.2.1 in the cube experiments are observable such as the background noise effected the cubes width measurements in the virtual and real environments.

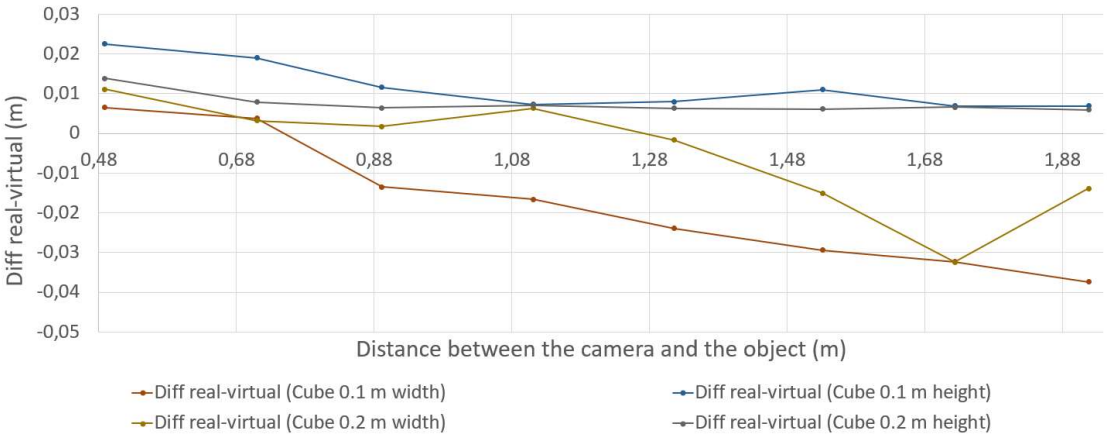


Figure 6.34: "Difference" between the real and virtual answers for the rectangles and cubes results

Table 6.15: Difference between the real and virtual answers for object distances of 0.49 m, 0.71 m, 1.32 m and 1.92 m

Object distance	**0.49 m (mm)**		**0.71 m (mm)**		**1.32 m (mm)**		**1.92 m (mm)**	
	Height	*Width*	*Height*	*Width*	*Height*	*Width*	*Height*	*Width*
Cube 0.1 m	22.5	6.54	19.0	3.74	7.95	-23.9	6.93	-37.4
Cube 0.2 m	13.8	11.1	7.84	3.17	6.34	-1.68	5.97	-13.9

According to these results, the differences between the real and virtual responses show an almost constant trend, except for the width of the cubes and the diameter of the 0.3 m sphere. Variations in these responses are due to noise and light effects combined with pixel resolutions. These sources of error have been previously identified as the main contribution to the deviations observed.

A compensation in the sources of error in both environments is then possible, given a constant data trend observable in the cubes, circles, and spheres experiments. In addition, the mean differences in errors in both environments are highlighted by aberrant points.

The novelty of this Chapter is identified as being:

- The virtual system developed in Chapter 5; Section 5.4.1 has been confirmed as being robust.
- The identification of a constant difference between the real and virtual answers that can be used as an offset to predict the real system behaviours.
- The identification of the new sources of errors in Blender (texture position and floor colour).

In Chapter 7, the performance of the virtual system developed in Chapter 5, Section 5.4.1 is tested in three experiments concerning the measurement of complex object in a controlled room, performing camera position localisation, and measurement of the perimeter of five duck eggs in a new illumination environment.

Chapter 7

Applications

In the previous chapters, a photorealistic digital twin for multi-camera measurement systems was developed in Blender, and its performance was evaluated through a robustness process. The final element of this research was to test the limits of the system.

Three experiment scenarios were designed for this purpose:

1. The ability of the system to reproduce results and behaviours similar to those in the real environment for a more complex object, than those used in previous chapters.
2. Reproduction of the complex environment in Blender, and the characterisation of the impact of light.
3. Characterisation of the impact of the light-object interactions.

The first experiment took place in the controlled environment described in Chapters 3 and 4, and the artefact used was a calibration ball bar. The aim of this experiment was to measure the diameter of the calibration spheres and the distance between their geometric centres.

The second experiment took place in an office style meeting room. The objective of this experiment was to measure the reprojection error of the previously used checkerboard at three different locations in the room, with artificial and natural light.

The last experiment also took place in the office style meeting room, but the object measured was more challenging – in this instance a duck egg.

The methodology of the experiments highlighted above will be introduced in Section 7.1. The results will be presented in Section 7.2, and a discussion of the experiments and their consequences will be undertaken in Section 7.3.

In addition, the following challenges are expected to be met:

- Are the same results observable between the real and virtual environments?
- How to simulate a more complex illumination environment in Blender (light and texture)?
- What is the impact of the illumination environment on the real and virtual measurement?
- How to simulate a more complex environment in Blender?
- How to simulate an egg in Blender?

These questions lead to the following potential innovations:

- Identifying the weaknesses of the virtual system.
- Determination of the further work that should be followed.

7.1. Methodology

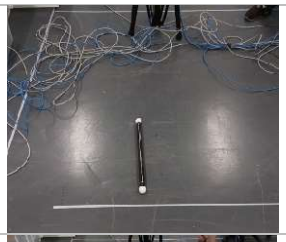
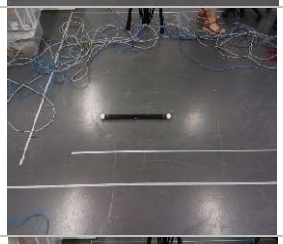
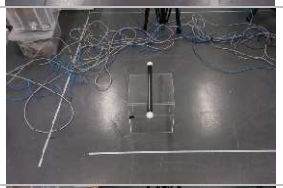
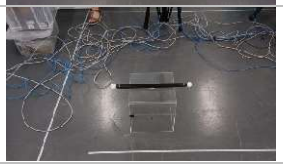
7.1.1. Experimentation 1: Ball bar

The methodology of the first measurement scenario here in Chapter 7 was based on the methodology of the experiment introduced in Chapter 3, Section 3.3.1 concerning the sphere diameter measurement. However, the aim was to measure the diameter of the two spheres constituting a calibration ball bar (presented in Section 3.1.3), and the distance between their geometric centres using a set of eight Raspberry Pi V2 cameras. The cameras were located at 1.0 m and 1.2 m, in a rectangular configuration (shown in Figure 125), around the artefact. The choice of the distances came from the results of the checkerboard experiment in Chapter 6. The reprojection error previously calculated was approximately 0.25 mm for both distances, indicating that these camera locations were optimum. This experiment was conducted in the controlled environment used in the previous experiments described in Chapters 4, 5 and 6.

Experiment

For each ball bar the experiment was run three times (repeatability) and consisted of taking a picture of the ball bar chosen at the different positions illustrated in Table 7.1.

Table 7.1: Descripting of the ball bar position with the 500 mm bar

Description	Illustration
The ball bar was placed on the floor, perpendicular to the length of the cube formed by the camera.	
The ball bar was placed on the floor, parallel to the length of the cube formed by the camera.	
The ball bar was placed on a 300 mm length transparent cube, perpendicular to the length of the cube formed by the camera.	
The ball bar was placed on a 300 mm length transparent cube, parallel to the length of the cube formed by the camera.	

The ball bar was put on a transparent cube to identify the impact of the modification of the z-axis in the real and virtual measurements. In addition, a transparent cube was used to avoid any change in the contrast between the balls and the floor. The eight cameras took images of the artefact in each configuration, and the MATLAB based set of algorithms used in Section 4.3.1 were used to calculate the diameter of both spheres and the distance between their centres.

Real set up

The real-world experiment comprised of the large zone circle board presented in Section 3.1.2, the ball bar introduced previously, and a set of eight Raspberry Pi v2 pinhole cameras located at 1.0 m and 1.2 m, in a rectangular set up, from the artefact.

The experiment took place in a controlled environment (parameters such as light intensity control, noise coming only from the cameras had known values), with two rows of three 38.65 W LED (light intensity measured with a light meter (Sekonic C-800-U)), located at 1.30 m from each other on the x and y axes. The large zone circle board was used to calculate the extrinsic parameters (rotation matrix and translator vector) of the set of cameras, to reproduce the real camera set up in Blender. The vertical distances (z-axis) between the cameras on the same tripod were 0.16 m, 0.22 m, 0.25 m, and 0.27 m.

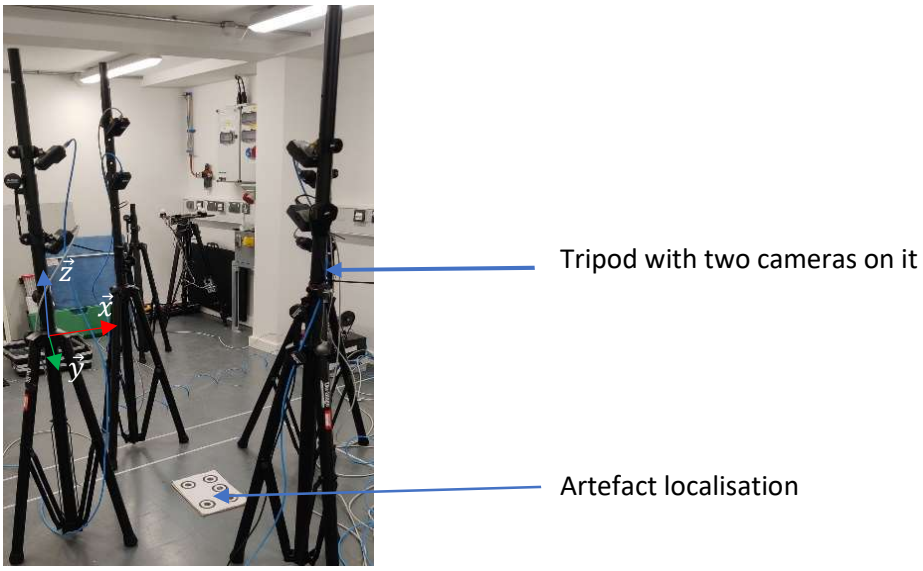


Figure 7.1: Real set-up

The textures of the different elements featured in the real set up were:

- Zone circle board: Diffuse.
- Balls of the ball bar artefact: Diffuse.
- Bar of the ball bar artefact: Specular for the body, and diffuse for the extremities.
- Floor: Diffuse.

And in the virtual set up:

- Zone circle board: Diffuse.
- Balls of the ball bar artefact: Diffuse.

- Bar of the ball bar artefact: Specular for the body, and diffuse for the extremities.
- Floor: Diffuse.

Virtual set up

The real-world experimentation was then recreated in the virtual environment. Figure 7.2 is similar to Figure 5.2, in Chapter 5, due to the re-use of the virtual model developed in that chapter. The difference between these two scenes came from a new definition of the floor, and the design of the ball bar.

The texture of the floor was set up according to the results obtained in Chapter 6, in Section 6.2.3.3. A grey colour was used for the basic colour, and a normal map found on the website ambientCG (Public Domain Resources for Physically Based Rendering [291]) to recreate the scratches (imperfections) on the floor surface.

The diameter of the spheres was measured with the MATLAB toolbox presented in Section 3.3.1, and the image treatment was performed as in Appendix 3.

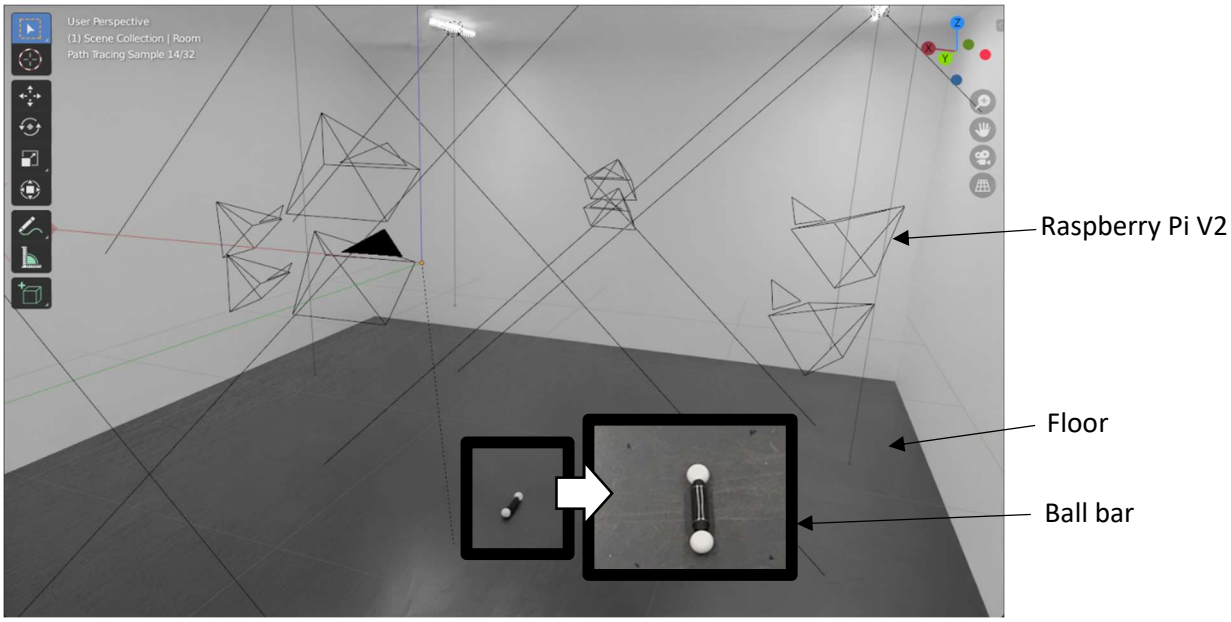


Figure 7.2: Virtual set-up

7.1.2.Experimentation 2: Meeting room

The second measurement scenario consisted of measuring the reprojection error of the previously used checkerboard with a set of Raspberry Pi v2 cameras, in an office style meeting room, shown in Figure 7.3. The methodology of this experiment was based on the methodology of the experiment presented in Chapter 3, Section 3.4.2.

Figure 7.3: Meeting room

This room was chosen for:

- The windows facing southwest (allowing sunlight in).
- The white colour of the walls.
- The three windows in the back of the room (this allowing changing ambient light).
- The small number of objects in the room.
- The table in the middle, causing the object to be higher. In addition, the table imposes a simulation and experimental constraint, because it is too heavy to move. It was an opportunity to work in a more constrained environment, where cameras cannot be placed everywhere, in order to test the robustness and flexibility of the system.

To assess the impact of the lighting environment on the checkerboard, it was placed in three locations every hour. The experiment was carried out on 27 October 2022, from 12 noon to 7pm. The weather on both days was sunny with a few clouds in the sky, providing good lighting conditions. This period was chosen to follow the sun's trajectory and identify the impact of the sun's different positions on the measurements. It should be noted that by 7pm, the sun had set and only artificial lights were illuminating the room.

The checkerboard was placed in the centre of the table, at three different positions, called opposite, middle, and under the window in the rest of the document, and noted with the number 1 (opposite), 2 (middle), and 3 (window) in Figure 7.3. The repeatability was ensured by rotating the calibration board clockwise at each position on the table, three times. Four calibration board variants were then generated for each position on the table (parallel to the width of the table, diagonal, perpendicular, and diagonal).

Due to the room limitations, only four cameras were used to perform the measurement for positions 2 and 3; and six were used for position 1. The MATLAB scripts used to process the image taken by the cameras were the same as used in Section 5.1.2.4.

According to Appendix 3, the checkerboard reprojection error before using the Bundle adjustment was between 22 mm to 23 mm for four cameras, and 12 mm to 13 mm for six cameras. The results given by the four and six cameras are then different and less accurate than for eight cameras. However, due to the limitations of the room, it was difficult to use eight cameras, and this error will be taken into account in the analysis.

The distance between each camera and the checkerboard was 0.71 m and 0.91 m. These distances were chosen according to the test run in Blender to find the optimum camera localisation in this room. The second virtual experiment, which was the replicate of the real one, was completed to determine whether the virtual light environment would have similar impact on the measurement as the real environment.

Blender test: find the optimum camera position in the meeting room

This experiment was developed to determine the optimum camera localisation in the meeting room by simulating a set of virtual cameras in its virtual replication. The zero of the coordinate frames corresponded to the middle of the table in the room. The chairs and other furniture were not simulated due to their possibility to be moved out of the room.

The two checkerboards used in Chapter 6 were re-used in this experiment due to the field of view of the cameras. The smaller one was used between 0.71 m and 0.91 m, and the largest one between 1.1 m and 1.7 m. The object distances chosen were 0.71 m, 0.91 m, 1.1 m, 1.3 m, 1.5 m, and 1.7 m.

The camera configuration was determined according to the camera behaviours, and the results obtained in the previous experiments. Through these experiments it was noticed that:

- For incidence angles less than $\pm 45°$ with respect to the x-axis, the reprojection error is large. The first and last cameras needed to be placed at $+45$ ° and $-45°$ from the x-axis.
- In the real-world, due to equipment limitation, it was difficult to simulate $\mathbf{90}$ °, however it was physically possible to place cameras at ± 45 °from the z-axis.
- In the virtual world, a camera placed perpendicular to the board gave the smallest reprojection error.
  A camera had then to be placed perpendicular to the board, to check if the result obtained were correct.

These results helped to design a draft of the camera configuration for this experiment. It was chosen that cameras 1 and 6 were to be located at $\pm 45°$ from the x-axis, and camera 4, perpendicular to the board. The first draft is displayed in Figure 7.4.

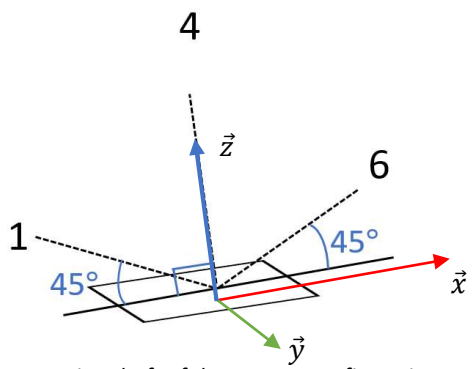


Figure 7.4: First draft of the camera configuration

Regarding the meeting room geometry, two camera configurations are possible. The first configuration, shown in Figure 7.5, placed the cameras in a circular pattern over the table. This pattern was generated with the mesh tool in Blender. This tool can generate automatically different objects such as meshes (cubes, spheres), lights, cameras, and path. The circle pattern, shown in Figure 7.5, was then generated with this tool, and was perpendicular to the length of the table, and its centre was the same as the centre of the table, e.g., the centre of the coordinate frame.

To cover the whole space around table, corresponding to the measurement volume, three other circles were proposed, diagonal to the table (+ 45° and - 45° from the x-axis) and parallel to the length of the table.

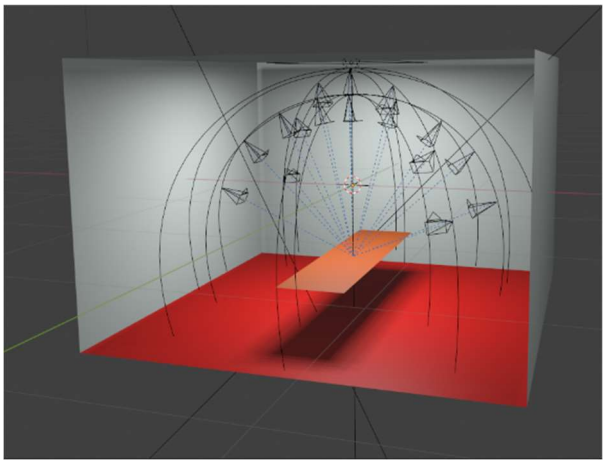


Figure 7.5: Circle configuration

However, the room was a rectangular in shape. Using a circular pattern for the camera installations excluded some camera positions due to the circular shape. A rectangular shape for camera placement was chosen. However, in Blender, the option to generate rectangular path does not exist, only circular or linear path can be generated automatically using the mesh tool. This path, illustrated in Figure 7.6, was then generated through a python script, where the position of each camera was manually entered in the script, to create a rectangular pattern. Noted that the camera positions were setup according to the results of Chapter 4.

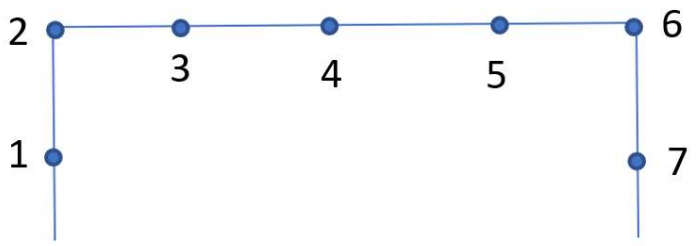


Figure 7.6: Rectangle configuration

Figure 7.6 shows the final camera configuration, used in this experiment. As in Figure 7.4, cameras 1 and 7 were located at $\pm 45°$ from the x-axis, and camera 4, perpendicular to the board, and in the middle of the configuration. Cameras 2 and 6 were located 1/3 of the rectangle length from cameras 1 and 7 on the z axis; and cameras 3 and 4 were at ± 45 °from the z-axis. The value of 1/3 was chosen as a first estimate to observe the results and allow the refining of the location of the camera at all the points.

The number of cameras was then seven in this experiment, different than number previously given. This difference is due to the fact that this experiment was carried out to determine the optimal camera location without taking into account the restriction of the equipment in the meeting room. In addition, this experiment was used as a baseline and was not replicated in the real world.

Real set up

The real-world experiment comprised of the small zone circle board, the small checkerboard presented in Section 5.1.2.4, and a set of six Raspberry Pi V2 pinhole cameras located at 0.71 m and 0.91 m, in a rectangular set up, from the artefact. The distances between the cameras on each tripod were similar, 0.24 m for a distance of 0.71 m and 0.3 m for a distance of 0.91 m.

The small zone circle board was used to calculate the extrinsic parameters (rotation matrix and translator vector) of the set of cameras, to replicate the virtual camera set up in Blender.

The experiment, illustrated in Figure 7.7, took place in a none controlled environment (4.42 m x 8.45 m x 2.81 m), with two rows of three 10.22 W LED (light intensity measured with a light meter (Sekonic C-800-U)), located at 2.02 m from each other on the x axis, at 2.9 m on the y axes, and 1.4 m on the z-axis. Three windows (1.16 m x 0.86 m) were located at the back of the room, letting the ambient light into the room. The calibration board was placed on the middle of a wooden table (1.42 m x 5.48 m). The walls of the room were white, allowing the reflection of the light, and the floor covered by a red carpet.

The textures of the different elements in the experiment were considered as:

- Large checkerboard: Diffuse.
- Carpet: Diffuse.
- Table: Diffuse.
- Walls: Diffuse.
- Windows: Specular.

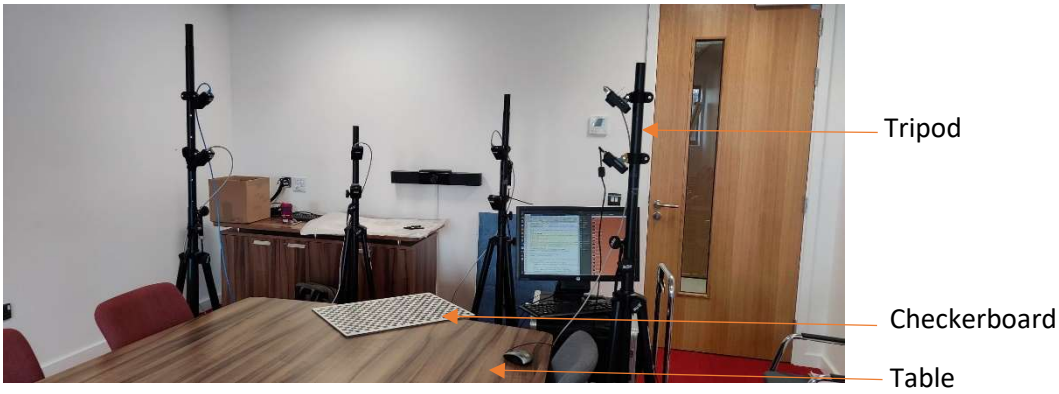


Figure 7.7: Real environment set up

Virtual set up

The real-world experimentation was recreated in the virtual environment. The whole scene was generated in Blender using a Python script as shown in Figure 7.8.

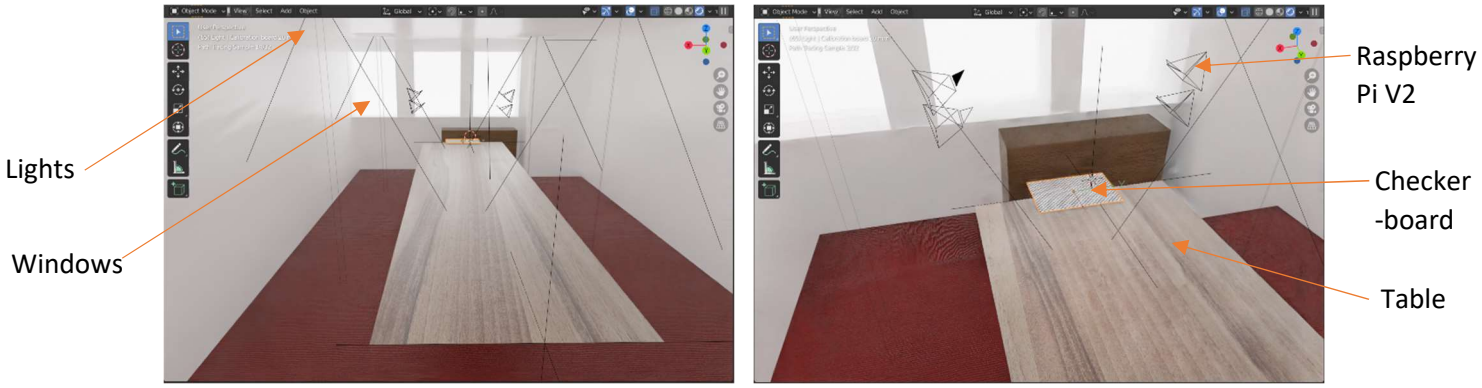


Figure 7.8: Virtual set up: Whole view of the scene (left picture), and a zoom on the Checkerboard (right picture)

The Blender model was composed of the same elements as the real environment. The lights were rectangular lights placed at 1.4 m above the table, with a power of 10.22 W. Two lights were created for each “real” light, and oriented at + 45 and – 45 degrees to recreate the light spreading in the real environment, similar to Chapter 4.

The sun light position was managed by the sun position in Blender. This option is available in the properties panel, in the world section. As shown in Figure 7.9, it places the sun at the right position according to the date, the place (GPS coordinates) and the time wanted. The distance between the sun and the object illuminated can also be defined with this panel, and the sun was oriented southwest as in the real model.

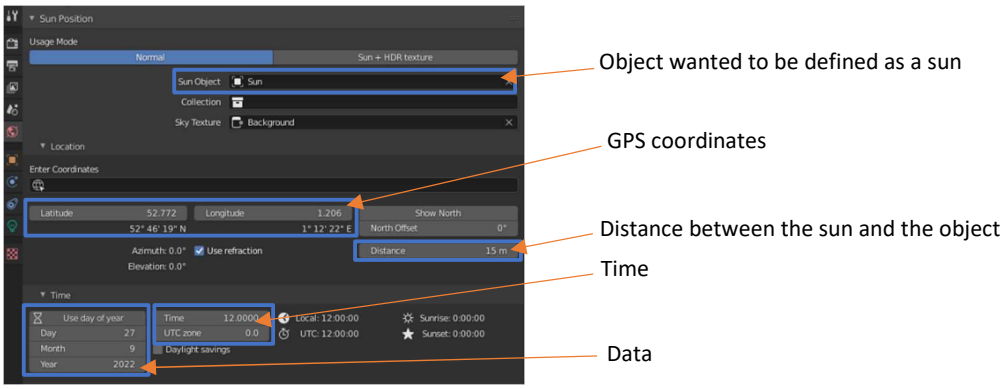


Figure 7.9: Sun position options

The MATLAB functions and data processing developed for the real-world scenario were used for the virtual data. However, due to the processing time required to generate the sets of images (3 days for the rotation of a board at each calibration position on the table each day), and the unreliability of the virtual light environment (presented in 7.2.2), only the first position was performed in the virtual setup (board parallel to the width of the table). This decision also provided an initial insight into the difference between real and virtual light environments and the behaviour of the virtual model.

In addition, in the real environment, due to the light on the zone circles board, it was sometimes not possible to perform the calculation of the extrinsic parameters and then the perfect replication of the real camera configured in Blender. Cameras that did not have their extrinsic parameters were not replaced to avoid making errors.

7.1.3.Experimentation 3: Duck egg

The third measurement scenario consisted of measuring the perimeter of five duck eggs (presented in Section 3.1.3 of Chapter 3) with a set of Raspberry Pi V2 cameras, based on the same methodology then the ball bat experiment. This experiment also took place in the meeting room used in experiment 2 and shown in Figure 7.10.

The objects were placed at the positions 2 and 3 illustrated in Figure 7.10, using the same experimental process, with the same number of cameras as used in experiment 2. A black background was used under the eggs to ensure the contrast between the eggs and the table, and the object distance was 0.91 m. The zone circles board used in experiments 1 and 2 was also used in this experiment to ensure the replication of the real cameras set up in the virtual environment.

The perimeter of each egg was calculated with the MATLAB set of functions presented in Section 3.3.7 of Chapter 3, for both environments.

Real set up

Due to the use of the same environment, the real set up for this experiment was exactly the same as for experiment 2. A black background was added to the table to increase the contrast of the eggs.

Virtual set up

As explained in the real set-up, the virtual set up was the same as the one used in experiment 2. The differences were the artefact used, the egg, and the addition of the black background. The egg was modelled in Blender by modifying the top of an icosphere (polyhedral sphere made up of triangles). A diffuse texture was applied on the object. The background was simulated by a black plane, with diffuse texture as well. Figure 7.10 shows the virtual set up.

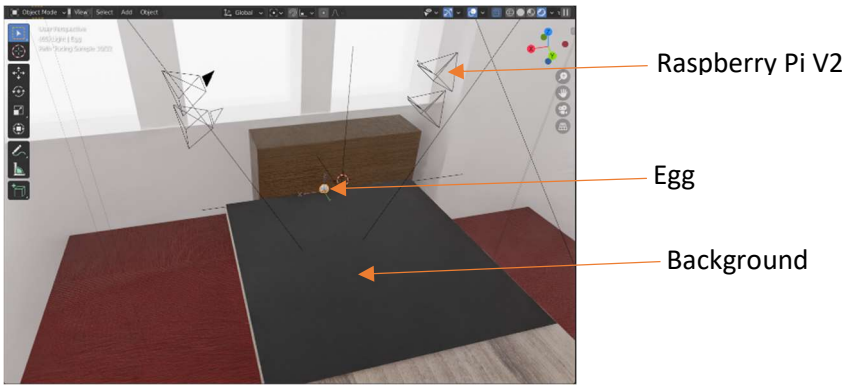


Figure 7.10 Virtual set up

7.2. Results

7.2.1.Experimentation 1: Ball bar

In the ball bar experiment, it was noted that the detection of the centre of the spheres was not consistent because the reprojection vector of each camera did not cross at the same point, and the average value of the centres of the spheres did not correspond to the point of crossing of the vectors.

The reprojection vector of a camera corresponds to the reprojection of the 2D point in the camera frame into 3D space. In this case, if the centre of the spheres is seen identically by each camera, their reprojection vector will intersect at the same point in the 3D space, given the 3D location of the centre of the spheres.

In addition, the 3D coordinates of the detected centres in each image will lie around this crossover point, and their mean value will be confused with it, which leads to an aberrant numerical value of the reprojection error (for instance 300 mm). The discrepancy between the mean value and the crossover point is usually due to a setting and detection error.

However, as shown in Figure 7.11 and Figure 7.12 (3D reconstruction of the average value of the two sphere centres detected in the real images), and Figure 7.13 (3D reconstruction of the detection of each sphere centre in the real images), the vectors do not cross at the same point, and the average values do not correspond to this crossing point, and the points are not located around it.

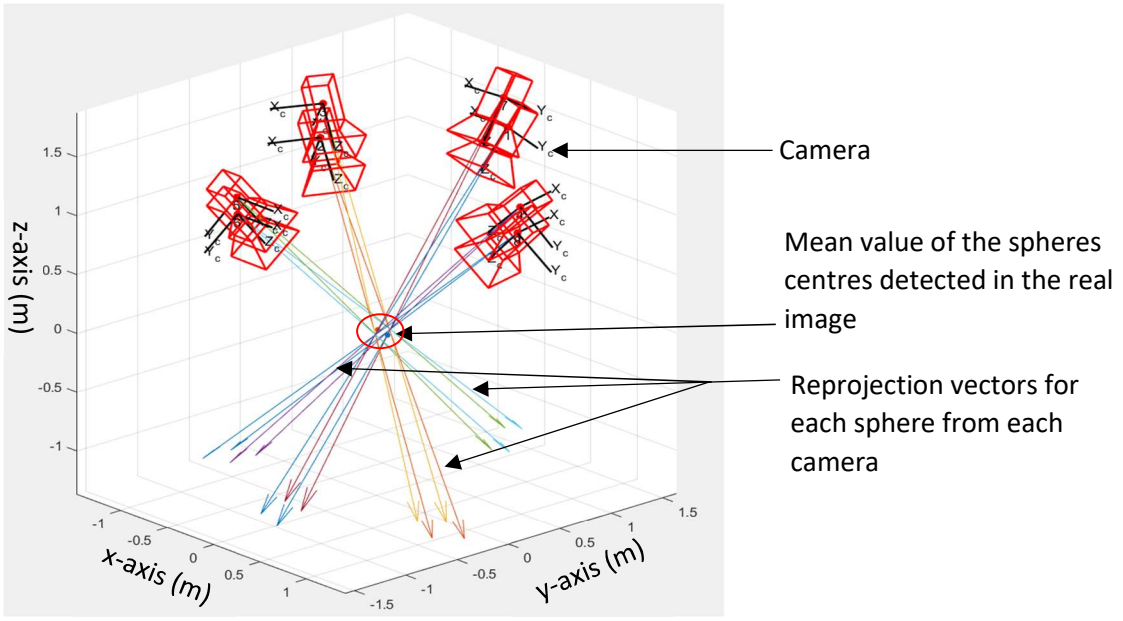


Figure 7.11: 3D reconstruction of the mean value of two sphere centres detected in the real images with the reprojection vector for each camera.

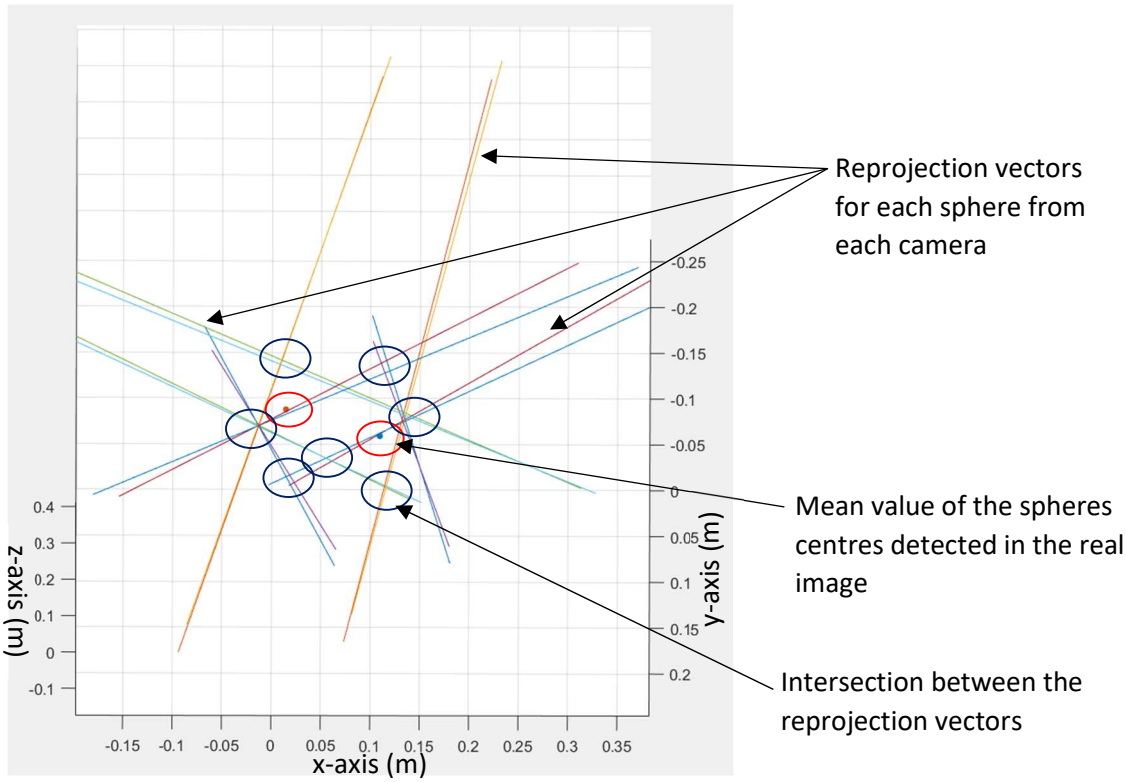


Figure 7.12: (Top view) Zoom in on the points of intersection between the reprojection vectors and the mean value of the centre of the spheres (real data)

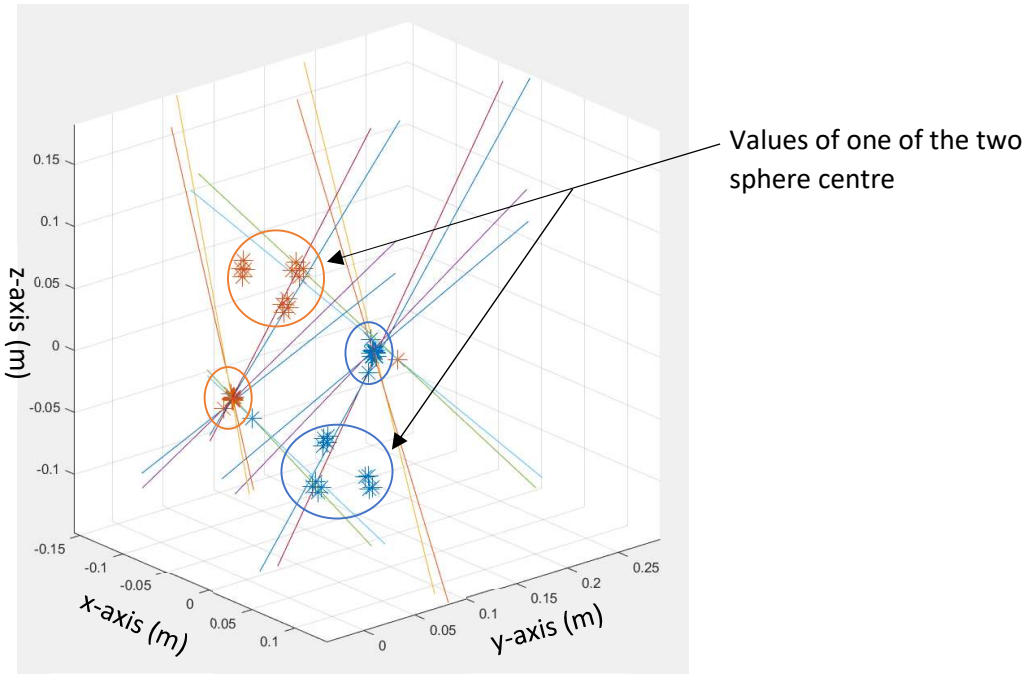


Figure 7.13: Zoom on the 3D reconstruction of each value of two sphere centres detected in the real images with the reprojection vector for each camera

The mean values shown in Figure 7.11 and Figure 7.12, were completed with all 3D coordinates of the centres of the sphere detected in the set of images by triangulation, and the reprojection vectors are represented by the coloured lines coming from the cameras. Regarding these Figures, the reprojection vectors intersect in seven different points (in the blue circles in Figure 7.12), which means that the coordinates of each 3D point in each image of the set do not correspond to the same values. The triangulation has then failed.

This idea is reinforced by Figure 7.13, where the centre of each sphere detected in each image of the set is plotted. It can be seen that the points are not located in the same place, and are separated into clusters, consisting of different numbers of data, around four of the different intersection points identified in Figure 7.12. These four intersections are then the true vectors intersections, and the others were generated by the plotting, and the rotation of the image in the 3D space. Furthermore, the reconstruction for the virtual environment gave exactly the same reconstruction result, which means that the same errors occur.

To understand how this detection error occurred, a 5M diagram was drawn, displayed in Figure 7.14.

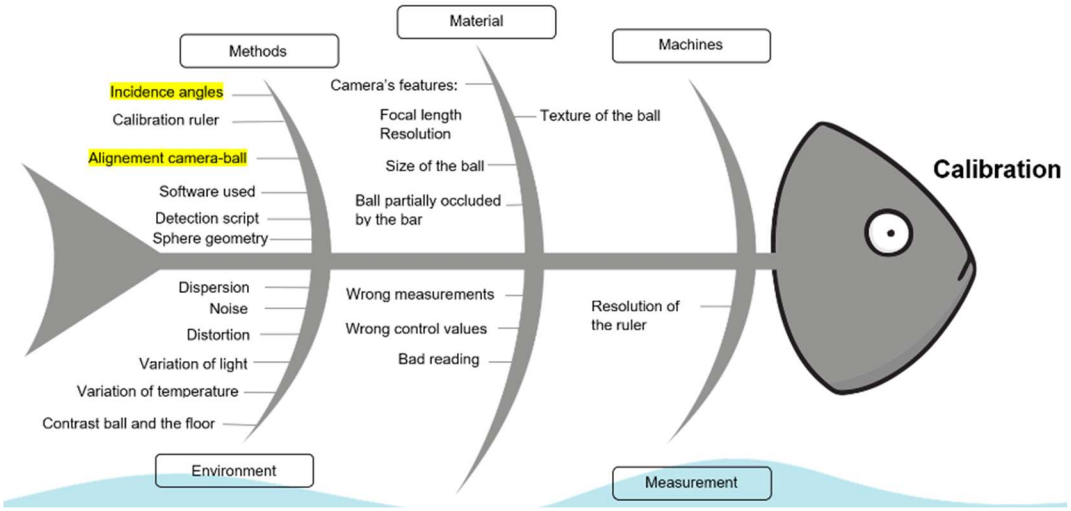


Figure 7.14: 5M diagram know as fishbone diagram

In Figure 7.14, two sources of error are highlighted in yellow, and are identified as the main cause of the detection error in this experiment. The evidence from this conclusion came from the reprojection errors experiments in Chapters 4 and 5 (incident angles), and Figure 7. 15 (3D reprojection of the 200 mm sphere detected in Chapter 4).

In Section 4.2.4, Blender was used to model the incidence angle around the large checkerboard for a distance varying between 2 m and 10 m. The conclusion from this experiment was that for some angles (between $45°$ and $90°$), the reprojection error, and the camera position were incorrect. Moreover, the distance was an important parameter playing on the angle and the reprojection error. The reprojection was largest for distances larger than 4 m.

In Figure 7. 15, the (Chapter 4) real-world result of the 3D reconstruction of the 200 mm sphere centres detected by the Raspberry Pi V2 moving from 0.2 m to 2 m from the sphere is shown. The experimental method was the same as described in Chapters 4 and 5. The camera was placed at the same height as the sphere, and was moved on the x-axis of the sphere. The optical centre of the camera was aligned with the geometrical centre of the sphere.

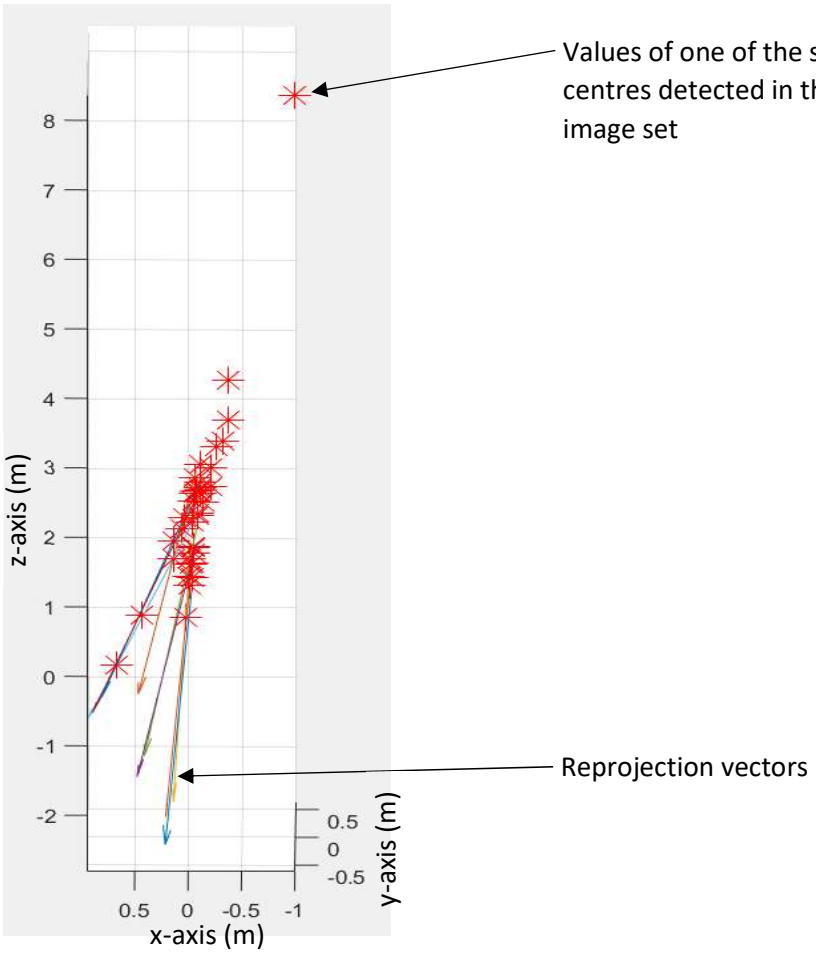


Figure 7. 15: 3D reprojection of the 200 mm sphere detected in Chapter 4

In Figure 7. 15, the cameras are not shown in order to better visualise the distribution of the red crosses, which correspond to the centre of the detected sphere in each image of the set for each possible camera pair. In the case of perfect detection, all the crosses would be aligned along the same line, spaced by the distance of the object. However, the red crosses are scattered in the Figure 7. 15, showing that the centre of the sphere is not at the same place in each image.

This misalignment problem comes from human factors of the real-world experiment setup that was propagated to the virtual setup via the camera parameters, and the measurements of the object positions in the setup. When the camera is moved in reality, it is difficult to position it exactly in front of the sphere. Moreover, this exercise becomes more difficult as the distance between the camera and the object increases. In the ball bar experiment, similar results were observable in Figure 7.11, and showed that the camera optical centre was not perfectly aligned with the sphere centre, causing an error.

As a reminder, the plot of the 3D reconstruction of the Blender results are not presented because they are identical to the real world one. However, it is interesting to note that the virtual equivalence of this experiment behaves identically as its real peer, even if some differences exist between them (camera model, surface texture).

Despite the detection problem, the diameter of one of the two spheres, of the ball bar artefact, was measured in both the real and virtual environments at an object distance of 1 m, and plotted against the camera that took the image. The results are shown in Figure 7.16, and Table 7.2.

In Figure 7.16, the x-axis represents the cameras used in the setup, and the y-axis the deviation on the sphere diameter.

Multiple repeats (n = 3) of the experiments were completed to determine any variance in the output. The standard deviation on each repetition was 0.76 mm, 1.7 mm and 1.8 mm, with a mean standard deviation of 1 mm.

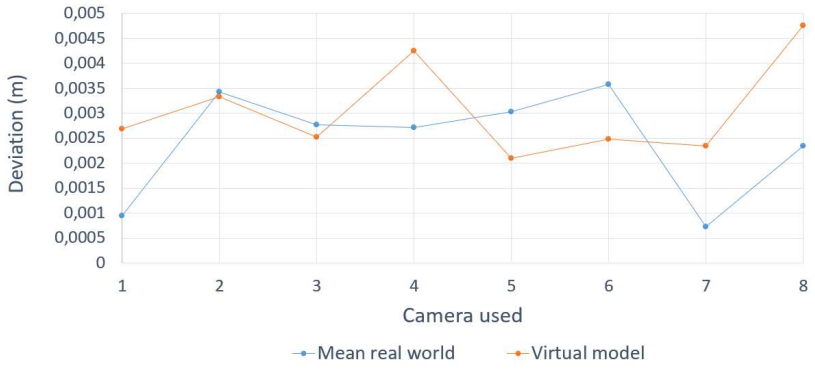


Figure 7.16: Sphere diameter deviation for each camera constituted the setup for both environments

Table 7.2: Deviation of the sphere diameter for camera 2, 3, 5 and 7 in both environments

Set up	Cam 2 (mm)	Cam 3 (mm)	Cam 5 (mm)	Cam 7 (mm)
Real	3.4	2.8	3.0	0.7
Virtual	3.3	2.5	2.1	2.3

Figure 7.16 and Table7.2 demonstrate that the sphere diameter deviation is not constant, variations are observable in both data sets, which appear to be a function of the camera position with respect to the sphere. Due to the rectangular set up of the camera around the artefact, some of the cameras are closer to the sphere chosen for the detection than others, thus having better viewing angles than others.

It is also observable that the difference between the real and virtual answers is approximately 1 mm from almost all the cameras, except for cameras 2 and 3. This difference might come from the same sources of errors identified in Chapters 5 and 6, e.g., the camera model, the illumination environment, and the surface texture.

7.2.2.Experimentation 2: Meeting room

Blender test: find the optimum camera position in the meeting room

Figure 7.17 shows the results for the smaller checkerboard, and Figure 7.18 for the larger checkerboard. For both figures, the x-axis represents the camera number, and the y-axis the reprojection error.

In Figure 7.17, the distance object was 0.71 m and 0.91 m, while it was 1.1 m, 1.3 m, 1.5 m and 1.7 m in Figure 7.18.

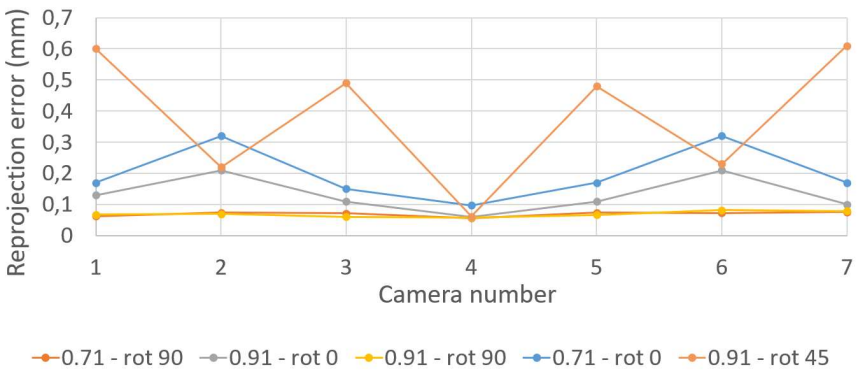


Figure 7.17: Reprojection error with the small Checkerboard

The data for a rotation of 90 ° of the checkerboard from the x-axis (parallel to the checkerboard) for both distances give the best results, and the data for a rotation of 45° of the checkerboard from the x-axis (diagonal to the checkerboard clockwise), gives the worst results. It is also noticeable that the best results for every configuration is given by camera 4. The result of camera 4 is normal because this camera is perpendicular on the z-axis to the board, which is the best position, according to the results in Section 4.3, about the incident angle. In addition, a symmetrical pattern is observable in Figure 7.17. It seems that cameras 1-7; 2-6; and 3-5 gave the same answer.

Comparing these results with the placement of the camera in Figure 130, it is noticeable that these cameras are symmetrical. So, it is logical to have the pattern described in Figure 7.17, and Figure 7.18.

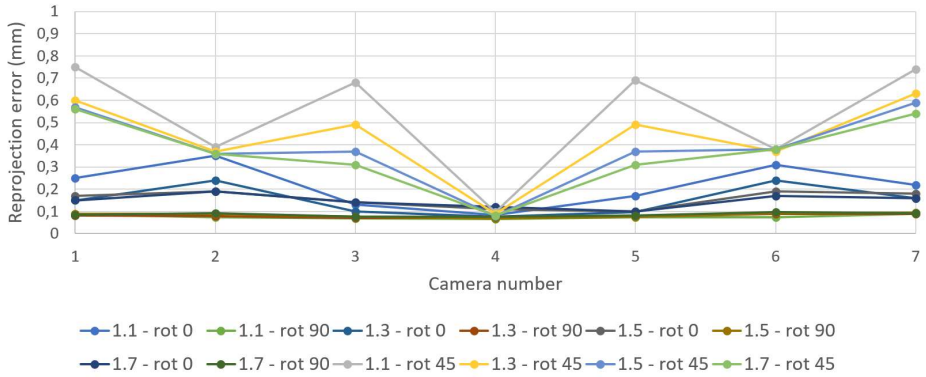


Figure 7.18: Reprojection error with the large Checkerboard

The same remark as previous is appliable on Figure 7.18 as well. The data for a rotation of 90 ° of the checkerboard from the x-axis gives the best results, while the data for a rotation of $45°$ of the checkerboard from the x-axis gives the worst results, with the camera 4 having for every configuration the best results.

To check the symmetry in the results, the reprojection error was calculated, with the same methodology as the results shown in Figure 7.17 and Figure 7.18, for a rectangular configuration of the camera, rotated at $-45°$ (diagonal to the board anti clockwise) from the x-axis, at a distance of 1.7 m from the board. These results were compared in Figure 7.19, with the results obtained for a rotation of $+45°$ at the same distance.

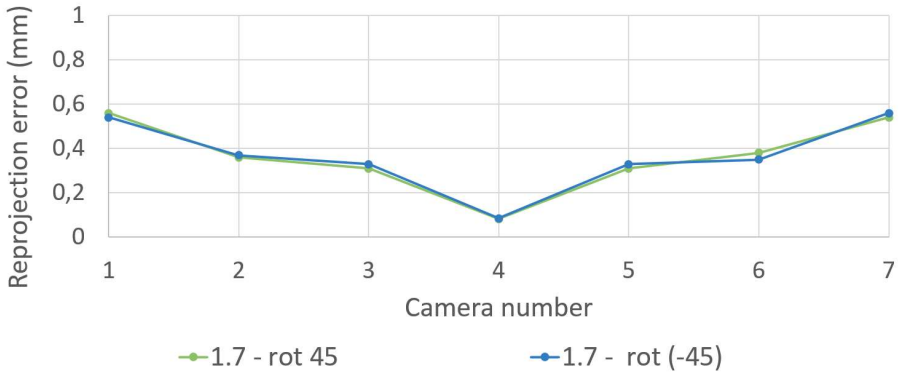


Figure 7.19: Reprojection error with the large Checkerboard to check the symmetry of the answer

In Figure 7.19, the two results give almost the same results. For instance, for camera 1, the rotation of 45 ° of the checkerboard gives a result of 0.56 mm, while the -45 ° rotation of the checkerboard gives a result of 0.54 mm. Furthermore, the symmetry of the data according to camera 4, which can be seen in Figure 7.17 and Figure 7.18, can also be seen in Figure 7.19, as well as the excellence of the results of camera 4.

The small difference between the two answers may be a function of the light, the camera angle (incidence angle), and the background of the pictures taken, shown in Figure 7.20.

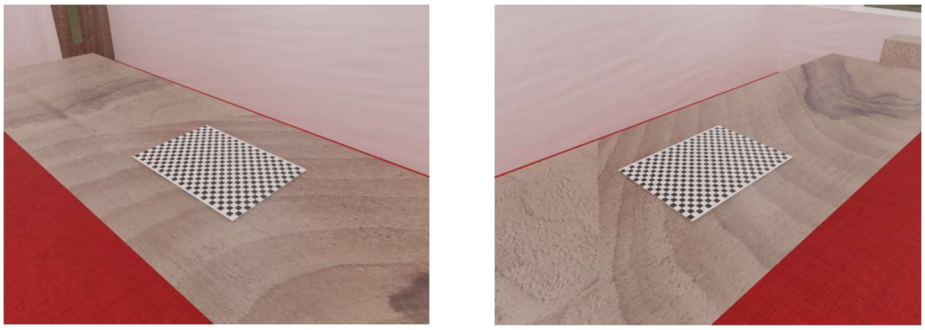


Figure 7.20: Right-rotation of 45°; Left-rotation of -45° taken by virtual camera 1

Real results in the meeting room

The real and virtual results in the meeting room are presented using XY scatter plots and not as previously with lines for a better readability and interpretation.

The results of this experiment are presented using three charts showing the results for each day time selected according to the position of the checkerboard on the tables. The x-axis of each figure, is the time of the day; and the y-axis, the reprojection error.

In Figure 7.21, it is observable that at 4pm the reprojection error is higher than for the other time of the day. Due to this peak and the extended y-axis, the data for 4 pm were removed, to provide a better reproduction of the other results, shown in Figure 7.22.

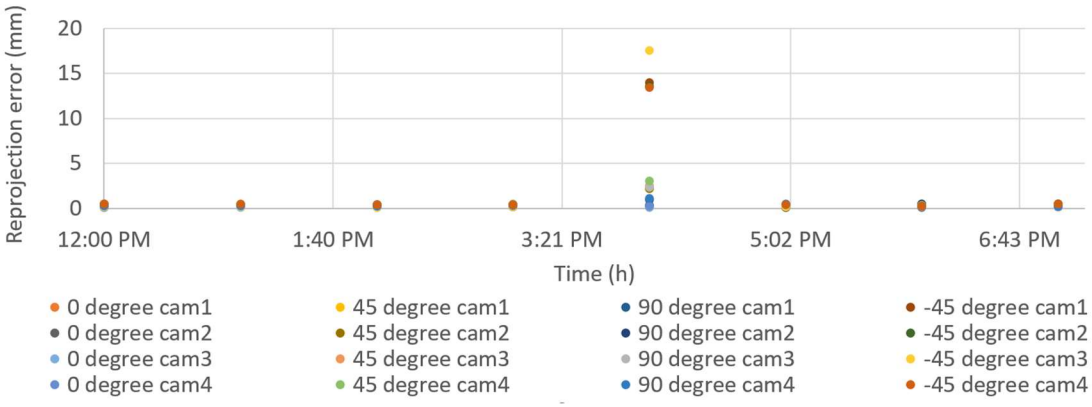


Figure 7.21: Checkerboard located opposite to the window

The data in Figure 7.22 shows the same trend. The reprojection error increases slightly with the time of day. Furthermore, as also shown in Figure 7.17 and Figure 7.18, the best results are obtained for a 90° rotation of the checkerboard with respect to the x-axis, and the worst for a 45° rotation of the checkboard. These results show that at 4 pm the sunlight covered the whole table and had an impact on the measurement due to the excessive amount of light on the measured object.

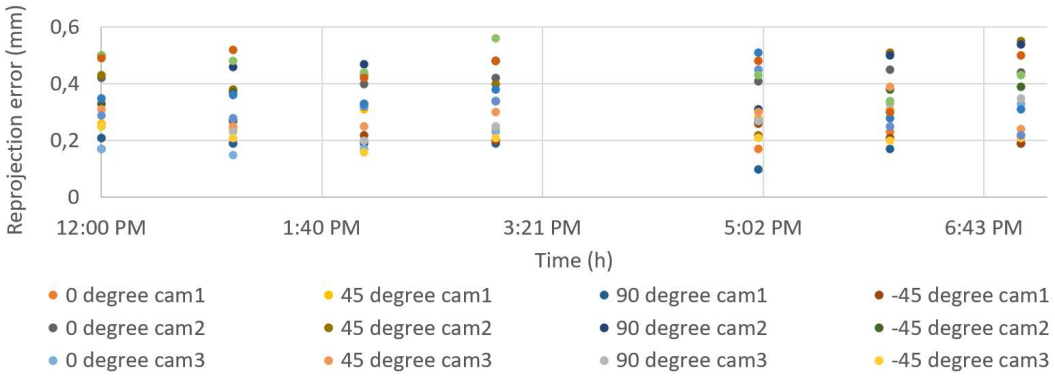


Figure 7.22 Checkerboard located opposite to the window without the data at 4 pm

The data in Figure 7.23 describes the same trend as Figure 7.22. The reprojection error increases with the time of day before decreasing. Furthermore, as also shown in Figure 7.17 and Figure 7.18, the best results are obtained for a 90° rotation of the of the checkerboard with respect to the x-axis, and the worst for a 45° rotation of the checkboard.

The calibration artefact placed in the middle of the table was more easily illuminated by sunlight than the previous position, and more often than the position under the window. The results in Figure 7.23 are more sensitive to the total path of the sun, and show when the room was more sunlit.

At 12 pm, the sun is generally at its highest point in the day, and the angle of illumination between the window and the incoming sunlight is the smallest. However, as the sun sets, this angle becomes larger, bringing more light into the room. At 4 pm is the peak of sunlight in the room, and therefore the largest angle of light between the room and the sun, before the sun sets at 6 pm.

In addition, 7 pm corresponds to night time, when only artificial lights illuminate the room. It is noticeable that the reprojection error is smaller with the artificial light than with the combination of artificial light and sunlight. This difference could be due to the orientation of the illumination. The artificial lights are exactly perpendicular to the checkerboard, while the sunlight comes in diagonally.

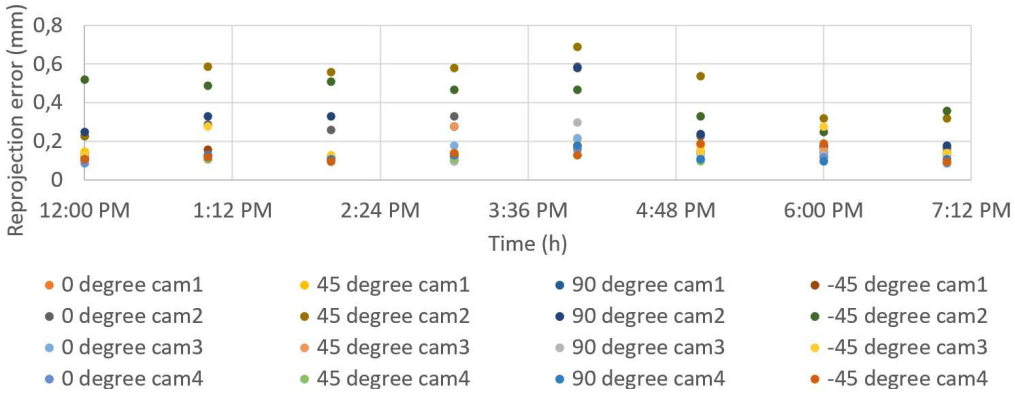


Figure 7.23: Checkerboard located in the middle of the table

In Figure 7.24, the board was located near the three windows, and the data describes a similar trend as far Figure 7.22 and Figure 7.23, where the board was located in the middle of the table, and

opposite to the window. The reprojection error increases from 12 pm to 1 pm before decreasing with increasing time of day. Furthermore, as shown in Figure 7.17 and Figure 7.18, the best results are obtained for a 90° rotation of the checkerboard with respect to the x-axis, and the worst for a 45° rotation of the checkboard.

Due to the position of the checkerboard near the window, the sunlight only fully illuminated the board when the angle between the window and the sun was the smallest. So, after 1 pm, the sun illumination passed over the checkerboard placed near the window.

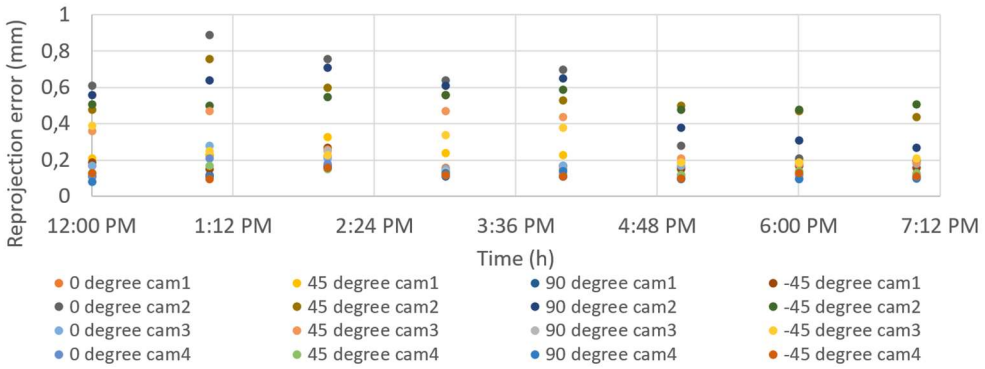


Figure 7.24: Checkerboard located under the windows

Finally, the reprojection errors in the Figure 7.22 to Figure 7.24 are between 0.05 mm and 17 mm, depending on the illumination. Otherwise, the data trend is similar in the four Figures, meaning that the light impact depends only on the board position and the time of the day.

Virtual results of the real application

The results of this experiment are presented in the same manner as the real results. The x-axis of each Figure is the time of the day and the y-axis, the reprojection error. Note that due to an excess of light on the zone circle board, it was difficult to detect the checkerboard accurately in all the images. This led to an incorrect calculation of the position of the cameras in the scene at each camera position used in this experiment, and to inaccurate or missing positions of the virtual cameras due to the use of real camera positions to place the virtual cameras in the virtual reproduction. The virtual replications of these cameras were therefore not created, causing some “data holes” in the charts.

In Figure 7.25, the reprojection error decreases from 12 pm to 1 pm before increasing from 1 pm to 4 pm, and decreasing again with increasing time of day. As for the real experiment, a peak is observable at 4 pm, for reasons mentioned previously (sunlight illumination).

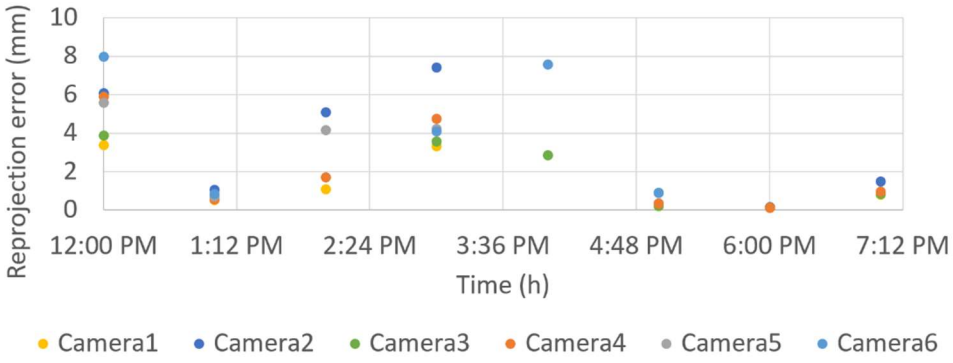


Figure 7.25: Checkerboard in located opposite to the window

In Figure 7.26 for the checkerboard in the middle of the table, the variation of the reprojection error increases and decreases over time, with peaks observable at 1 pm for camera 2 (reprojection error at 0.2 mm), sightly after 2.24 pm for camera 1 (reprojection error at 0.17 mm), and slightly after 4.48 pm for cameras 2 and 3 (reprojection errors at 0.17 mm and 0.16 mm). These peaks are difficult to explain due to the difficult to analysis the virtual data.

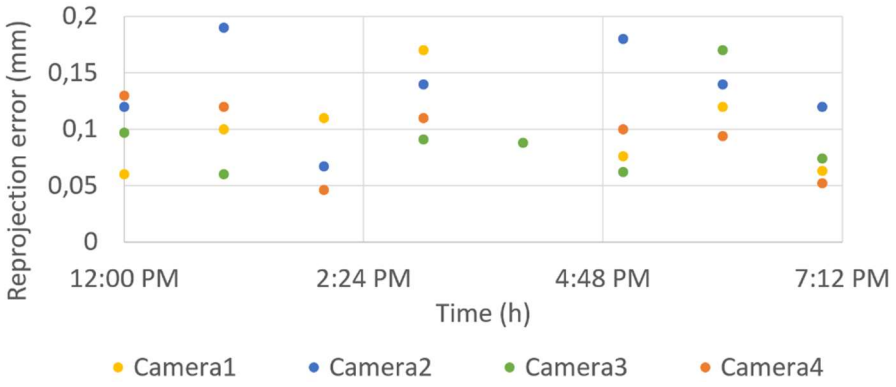


Figure 7.26: Checkerboard located in the middle of the table

In Figure 7.27, for the checkerboard located under the window, the variation of the reprojection error increases and decreases over time, with a peak observable slightly after 4.48 pm for camera 2, with a reprojection error at 0.2 mm. As explained previously, the peak at 4.48 pm is difficult to explain due to the difficult to analysis the virtual data.

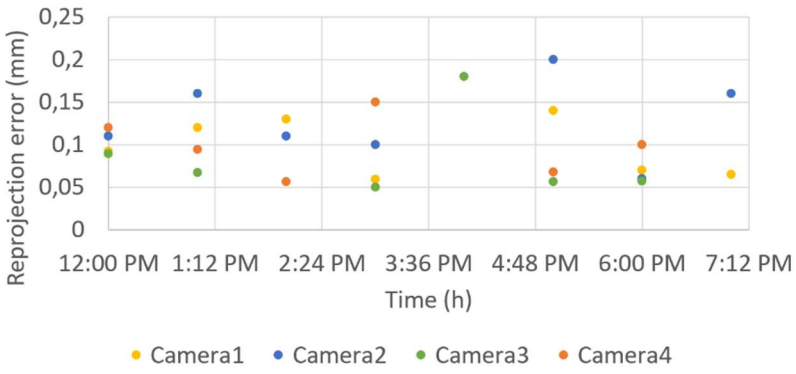


Figure 7.27: Checkerboard located under the windows

Regarding all the virtual data, it is difficult to draw conclusions. However, all the real and virtual answers have similar y axis, the reprojection error is then in the same magnitude, even if the virtual results were found not to fully describe the real system behaviours. This difference of behaviour comes from the light set up. In Figure 7.28, a real and virtual image of the Checkerboard near the window taken by camera 1, at 1 pm is displayed.

Figure 7.28: Real image on the right, virtual image on the left

The virtual image is darker than the real image. This impression comes from the position of the sunlight and the texture of the surface used in the virtual configuration, which are difficult to model correctly. This can come from a wrong definition of the texture surface properties (such as diffuse, reflected) of the window, the calibration board, the table and the floor. It also seems that the light coming from the sun created in Blender is not fully taken into consideration in the rendering process. This might be caused by incorrect definition of the rendering parameters.

7.2.3.Experimentation 3: Duck egg

Figure 7.29 shows the results for the eggs placed in the middle of the table, and Figure 7.30 for the location near the window. For both figures, the x-axis represents the egg number, and the y-axis the perimeter. The real and virtual answers were plotted on the same chart for comparison reasons.

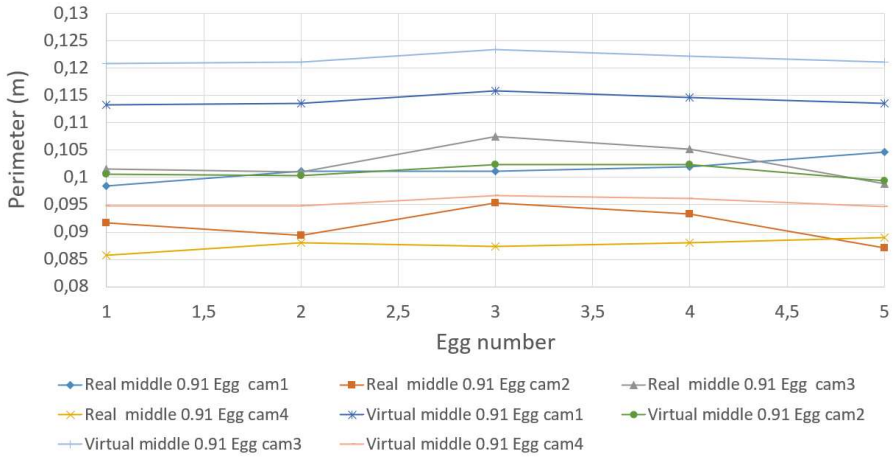


Figure 7.29: Eggs in the middle of the table, real and virtual answers

Table 7.3: Perimeters of the eggs 1, 3 and 5 detected by cameras 1 and 2 in both environments

	Perimeter egg 1 (m)	**Perimeter egg 3 (m)**	**Perimeter egg 5 (m)**
Cam 1 real	0.098	0.101	0.105
Cam 1 virtual	0.113	0.116	0.114
Cam 2 real	0.092	0.095	0.087
Cam 2 virtual	0.101	0.102	0.1

For the case of when the eggs are in the middle of the table, it is observable, in Figure 7.29, and Table 47, that the real and virtual answers follow the same data trends. In addition, some variations are observable between the detections of the set of five eggs, due to the shape of the egg (not uniform from one to another). It is also noticeable that the real and virtual answers are different. The virtual answers are larger than the real ones. However, the difference between both answers is approximately 0.01 m.

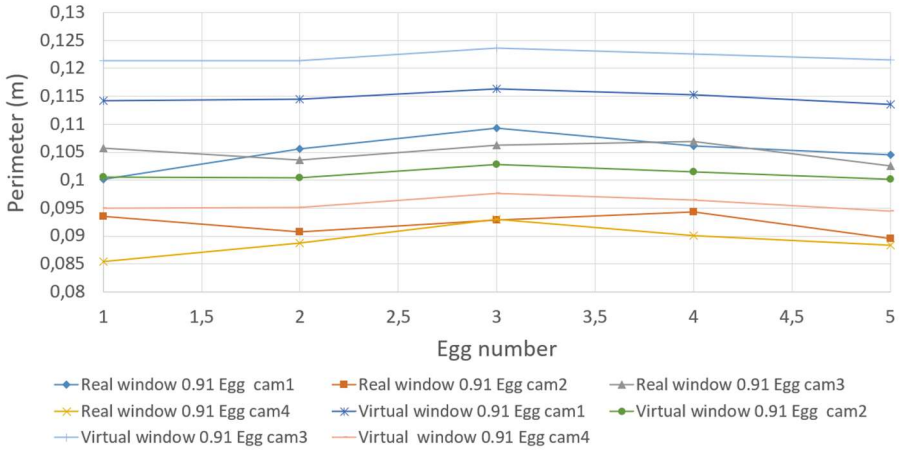


Figure 7.30: Eggs near the window, real and virtual answers

Table 7.4: Perimeters of the eggs 1, 3 and 5 detected by cameras 1 and 2 in both environments

	Perimeter egg 1 (m)	Perimeter egg 3 (m)	Perimeter egg 5 (m)
Cam 1 real	0.1	0.109	0.105
Cam 1 virtual	0.114	0.116	0.114
Cam 2 real	0.094	0.093	0.09
Cam 2 virtual	0.101	0.103	0.1

For the case of the when the eggs are near the windows, and according to Figure 7.30, and Table 7.4, similar results as those in Figure 7.29 and Table 7.34 are observable. In addition, the virtual answers are still larger than the real ones, with a difference of approximately 0.01 m.

The difference between the real and virtual results does not come from the light, but from the shape of the egg, as shown in Figure 7.31. In Blender, the dimensions of the real egg were established by manually defining the height, length, and width of the egg. However, in Blender, when the dimensions of an object are modified, only the height, width and length of the object are changed, not its shape.

In the case of the egg, the shape defined manually in the egg creation process was never changed, even though the dimensions of the egg were, resulting in five eggs with different dimensions but the same shape, whereas the real eggs had variations in shape and dimensions. However, it seems that the light environment and the shape of the object had no significant impact on the measurement.

Figure 7.31: Real egg (left) VS Virtual egg (right)

7.3. Discussion

The virtual model designed in Chapter 5 was tested in three different applications. The first application was taking place in the controlled environment and aimed to test the ability of the system to reproduce results and behaviours similar to those observables in the real environment for a ball bar. Through this experiment, the reprojection error could not be used to compare the real and virtual models due to erroneous values. The investigation of these values led to discovering two predominant sources of errors: the incidence angle and the alignment between the ball and the camera. These sources have not been identified in the previous chapters, due to the utilisation of large artefacts (checkerboard and zone circle board), and the location of the camera optical centre on the same x-axis than the small artefacts (cubes, spheres, circles) measured. However, even if two new sources of

errors were identified, the Blender model was affected by them, given the same 3D reconstruction than the real model.

The second application involved testing the new virtual model outside the controlled room, reproducing a more complex environment, and characterising the impact of light on the measurement. The new environment consisted of three windows at the back of the room, with a rectangular table in the middle. The test in this room consisted of placing a set of four-six cameras around the table, looking at the large checkerboard, and the reprojection error was used to determine the impact of light on the measurement. Designing this test in Blender was challenging because:

- Blender does not provide a 'ready-made rectangular trajectory', only circular and linear trajectories are provided.
- The location of the table and the dimensions of the room limited the number of cameras that could be used. Four cameras could be used near the window and in the middle of the table, and six at the other end of the table, near the door.
- The virtual cameras therefore had to be coded manually in the Python script used to generate the test in Blender.
- Due to a lack of knowledge about the design/creation of textures and lights in Blender, the real and virtual images were very different.

The last two parameters (camera location + texture and light creation) gave poor virtual results that were difficult to interpret. In contrast, the real results were good, demonstrating a correlation between the illumination angle and the degradation of checkerboard detection in the real images.

The last application was carried out in the same environment as the second. Its aim was to characterise the impact of light-object interaction. A set of five duck eggs was used. Because of their small size, the images had to be cropped, which degraded the image resolution, which was already poor due to the object-camera distance. Although the virtual light and texture were not identical to those in the real environment, similar results were obtained between the real and virtual environments. The conclusion is interesting because the shape of the virtual eggs and the real eggs was not the same. It therefore seems that, for a small 3D object, the shape of the object has no impact on detection.

7.4. Discussion and conclusion

In this chapter, three experiments were performed to explore the limits of the virtual system for measuring more complex objects in a more complex lighting environment.

Through the experiment of measuring the diameter of two 38 mm diameter spheres and the distance between their geometric centres, two sources of error were identified as in Chapters 4 and 5: the angle of incidence and the misalignment between the centre of the sphere and the optical centre of the camera. In addition, it was also observed that the virtual replication of this experiment had the same behaviour as its real counterpart. However, the diameter measurement was not affected by these sources of error, and the comparison between the real and virtual results showed that the virtual model performed worse than the real one, but the difference between them is persistently about 3 mm.

The experiment to measure the reprojection error of the checkerboard artefact in a complex light environment gave different virtual and real results. Analysis of the real results showed that the detection of the board was influenced by the light, and that the position of the artefact in the room was important, depending on the position of the sun. The virtual results were difficult to interpret due to a significant difference between the real and virtual images, resulting from the challenge of modelling the lighting environment and the surface texture of the objects in the scene. Further research is needed on these parameters to better understand them and generate a better model.

Finally, in the egg experiment, the difference between the measurements is approximately 0.01 m, even though the lighting environment and the shape of the egg were different between the real and virtual configuration.

The novelty of this chapter has been the identification of the weaknesses of the virtual model with respect to the light reproduction and surface texture management, as well as the determination of the further work that should be followed.

Chapter 8

Conclusion

Over the past few centuries, the concept of industry has evolved considerably, transforming from the agricultural and craft economy into one dominated by industry and machine manufacturing through the invention of new technologies. Nowadays, the world is faced again with a new type of industrial revolution, called Industry 4.0, which defines a new model and philosophy of industry by increasing mechanisation and automation, digitalisation, networking and miniaturisation. However, the digitalisation of current factories, as well as increasingly incorporating new technologies such as robots or artificial intelligence have also generated new needs and requirements.

As explained in Chapter 2, in the factory of the future, the development of virtual technologies such as simulation tools, and wireless networks, has introduced the notion of virtual space, and new possibilities to virtualise the physical assets of the production line improving decision-making, production time and competitiveness. In addition, with the rise of new technologies driven by this revolution, new solutions, based on these technologies, have emerged to better ensure worker safety, monitor different production technologies, optimise production and avoid collisions.

Optical systems based on multi-camera networks are one solution developed by industries due to their ease of use, implementation, low cost and non-intrusiveness. However, current solutions are not necessarily optimised, easy to use or can take a long time to implement. Regarding the nature of the optimisation problem and the digitisation policy, digital models and digital twins are being investigated to solve these problems.

Blender is an open-source 3D computer graphics software toolset used for creating animated films, visual effects, interactive 3D and virtual reality applications. It combines the functionality to model the interactions of scenes with light sources, and generate simulated images based on modelled cameras using a number of rendering engines options based on ray-tracing. However, Blender has received very limited attention as a viable metrology compliant environment for multi-camera measurement system modelling. Therefore, the objective of this research was to determine whether Blender was suitable for use as a modelling environment for multi-camera digital metrology systems; and to design one if so.

However, designing a digital model in Blender is challenging, as shown in Chapters 3 and 4. The virtual scene has to be realistic enough to match the real-world scenario, and the modelling boundaries must be defined correctly, otherwise the virtual model will provide different results compared to the real-world model (error magnitude: tens of millimetres for the real-world versus millimetres for the virtual world).

In Chapter 5, the main photorealistic parameters (environment, camera, light, surface texture) were identified and characterised to better design the final virtual model with appropriate detail. Through this photorealistic process, it was identified that the surface texture management in Blender was visual / artist oriented (not directly linked to common engineering definitions); the virtual and real camera models were inherently different; and that reproducing real-world scenarios in the virtual environment is challenging due to human and software limitations.

The experiments in Chapter 5 also showed that camera measurement systems can be simulated in Blender and that the result of the camera measurement system can be very similar to the real-world equivalent. The robustness study, presented in Chapter 6, showed that the virtual model is robust, albeit that a constant deviation may exist between the real and virtual model depending on the errors identified in Chapter 5, but also on the dimensions of the measured object.

Finally, the limitations of the virtual model were tested by three experiments in Chapter 7. From the results obtained, the Blender model was able to reproduce the same major sources of error as the real environment in a controlled environment, but gave different answers when a more complex illumination environment was modelled. Further studies on the construction of the virtual environment, as well as a better understanding of the rendering process, and a better definition of the texture surface will be needed to generate more realistic models, closer to real examples.

However, in Chapter 1, the objective of this project was to develop a model/digital twin of a multi-camera optical system for measuring shapes and (eventually) tracking humans and objects; and the result of this research is a virtual model of a multi-camera optical system.

According to Chapter 2, the digital twin is subject to various definitions. However, commonly the digital twin is defined as consisting of three elements, which describes a cycle between physical and virtual states:

- A physical product.
- A virtual representation of that product.
- The two loops of data connections. One from the Real Space to the Virtual Space, for the data transmission; and another one from the Virtual Space to the Real Space, for information/processes communication.

In view of this definition, the virtual model defined in this work can be considered as a digital twin. However, with regard to the digitisation process generated by Industry 4.0, the systems must be agile, resilient and fast. Virtual models and systems must therefore be autonomous to meet these needs, and then develop an architecture to generate two autonomous data loops (input and output) between the real and virtual layers of the system.

8.1. Research answers

In Chapter 1, three Research Questions were defined. Regarding the work completed in Chapters 4, 5, 6 and 7. Answers to the Research Questions are defined as follows.

Research question 1: Can a camera-based metrology system be modelled in a 3D animation software?

In Chapter 4, two experiments were designed to validate whether a 3D virtual reality modelling environment such as Blender can be used to fully model camera-based metrology systems, and potentially support the placement of camera-based measurement systems in industrial environments.

The first experiment was about the measurement of a 0.09 m sphere using three cameras, and the variation of the real-world results was between -46 mm and -2.02 mm, whilst the variation of the virtual results was between -2.15 mm and -0.74 mm. The second experiment involved the measurement of a camera calibration artefact used to optimise multiple cameras positions (in this case eight cameras), based on the reprojection error. The virtual results were between 0.03 mm and 4.8 mm, whilst in the real scenario, they were between 0.43 mm and 14 mm. Regarding the differences between the real and virtual results in these two experiments, the virtual model is better than the real one, and cannot be considering as a digital representation of the real world. In addition, the pixel density, and the colour were not the same in the sphere images generated by Blender, and coming from the virtual cameras, as well as the light definitions in both illumination environment (10 kW for the Blender model, and 39 W for the real model).

Due to the design of the virtual model based on the basic and default tools available in Blender, a lack of realism was observable, and the modelling boundaries were deemed to require better definition. However, through the modelling of these two experiments, and the validation of the laws of optics with the presence of the sphere geometrical error in the real and virtual results, Blender has shown significant promise as a tool to support investigative multi-camera metrology systems.

Research question 2: To what extent should a virtual measurement model conform to the real-world equivalent?

Due to the Chapter 4 conclusions, that Blender cannot be used as a digital twin in its raw or native environment (default parameters used to design a perfect environment), Chapter 5 was about designing a realistic model based on a photorealistic approach.

In Chapter 5, it was shown that photorealistic concepts can be used to proactively optimise (in this case, degrade) the default Blender environment to better model the equivalent real-world measurement scenario (camera-based measurement system measuring the diameter of spheres). The level of tuning depends on key parameters, in this case, the characteristics of the light, the texture of the object and the enhanced camera definitions. For example, in the camera characterisation experiment reported in Section 5.1.2.6, changing the camera parameters impacted the deviation of the diameter of the sphere, as well as the profile of the image frequency spectrum.

Similarly, the ability to refine the correlation between the Blender modelled measurement system and the real measurement system was limited by the scope of Blender's software tools and the subjective interpretation of the operator. For example, the surface texture definition in Blender is very different compared to the real-world engineering definitions, and is subjective.

The results demonstrate that, on an individual basis, it is possible to optimise the parameters that lead to equivalent measurement performance of the camera system in the Blender virtual environment. Furthermore, thanks to the data presented in Chapter 5 (showing the system response for each level of complexity), it is possible to identify, and select, the parameters necessary to achieve a similar level of reality compared to the real-world scenario without having a virtual model identical to the real one.

Research question 3: How can a metrology system based on a virtual camera and 3D animation software be made as traceable as a real-world system?

The question of the traceability was addressed in Chapters 5 and 6. In Chapter 5, two methods were used to characterise the variation of the virtual camera frequency spectrum regarding the complexity of its definition. The Discreet Fourier Transform (DFT) was applied to the Blender output and the real images to plot their spectrum and measure the position and the amplitude of their fundamentals. In these experiments, the fundamental of the basic virtual camera model was, for instance, at 133 Hz on the x-axis, and 129 Hz on the y-axis with a magnitude of 8,755,191, whilst the fundamental of the real camera model was at 156 Hz on the x-axis, and 162 Hz on the y-axis with a magnitude of 12,494,169: both for an object distance of 0.5 m.

However, the DFT shows only the fundamental of the camera spectrum, its logarithm was then used to visualise the whole spectrum. The logarithm of the DFT of the real camera, and of each virtual model generated with different complexity of camera parameters, showed that the complexity added to the camera model had a different impact on the camera spectrum. For example, the depth of field had an impact on the low frequencies; the level noise and lens distortion on the fundamental, high and low frequencies; and the shape of the aperture (blades) on the fundamental and high frequencies. The same observation as with the DFT was also observable with its logarithm; the spectrum of the real camera is different with all the generated virtual models regarding the position of the fundamental, the camera spectrum, and the camera answer on the sphere deviation as a function of the object distance.

In the same chapter, a no-reference image quality metrics was used to measure the percentage of blur of the virtual and real images as a function of increasing object distance, and the complexity of the virtual camera definition. The reference image was generated in MATLAB, with the same colour, and pixel resolution as the real one. Regarding the results obtained, the same conclusions as for the DFT analysis were found. In addition, it was identified that the best virtual model gave a higher blurred result.

In Chapter 6, the performance of the final virtual system was compared with the real model through the measurement of simple objects (circles, spheres, cubes). Through these experiments, the virtual and real-world absolute responses were shown to be different but with similar trends. With respect to the cube experiments (measurement of the cube width) the difference between the real and virtual results was between 13 mm and 20 mm; whilst for the circle experiments, the diameter difference was between 5 mm and 6 mm.

In addition, the reprojection error of the checkerboards and the zone circle boards were used to measure the performance of the virtual system with respect to the real system. Regarding the checkerboards, real and virtual reprojection errors were constant, with a difference between the real and virtual world between 0.12 mm and 0.18 mm for the small board; and between 0.30 mm and 0.49 mm for the large one. With the zone circle boards the real reprojection errors varied between -46 mm and 2.02 mm, whilst the virtual reprojection errors were between 0.006 mm and 6.34 mm. However,

the variations in the results were explained by a limited number of error sources such as the problem of virtual texture on the zone circle boards, objects illumination being different between the two environments, and pixel density of the images being different.

8.2. Thesis novelty and contributions to knowledge

In Chapter 1, a first definition of the research novelty was proposed. Regarding the research completed and described in the previous chapters, this anticipated novel can be extended, and reformulated through the four points below.

- **The investigation of a 3D animation software as a potential metrology tool.**

As explained in Chapter 4, very limited evidence exists to show that Blender itself (or any other similar software) has been considered as a viable compliant environment for multi-camera metrology system modelling. Blender was chosen because it combines the functionality to model interactions of scenes with light sources, and generate simulated images based on modelled cameras using internal rendering engine based on ray-tracing and Python scripting options.

Due to the lack of evidence of the use of 3D animation software, a need to investigate the Blender suitability for this task existed. This problem was solved by modelling two real scenarios; one about measuring the diameter of a sphere with a single camera; the other about measuring the reprojection error of a small and large checkerboard with a set of eight cameras; and comparing their results with the real world equivalents. The comparison between the two results shows the potential of Blender as a metrology simulation tool, but also the difference between the perfect environment generated in the software, and the real world.

- **The development of a digital model of multi-camera measurement system in Blender.**

Due to industrial requirements (safety, work and production management), a digital model of a multi-camera measurement system was needed. However, this concept has been previously explored, with solutions already available, but with drawbacks such as a lack of accuracy, or computationally very expensive.

In this research, a virtual model of a multi-camera measurement system was developed in Blender. Due to the lack of evidence of its use for this task, the first step was to study its ability to simulate a metrology tool, and to check whether or not the laws of optics were respected. A virtual scene was then built with the default Blender tools, and its performance was compared to that of the real model. This comparison identified a lack of realism in the virtual model, as well as problems in the definition of the model boundaries. The second step was then to improve the previous model using photorealistic concepts. However, this process revealed the limitations of the software and human understanding in generating a virtual model that is identical to the real one.

Finally, the last two parts of the development of this virtual model consisted of evaluating whether the model was robust or not, and identifying its weaknesses. Through the experiments, the numerical model proved to be robust, with similar behaviour and results to the real scenario in the simplest cases, e.g., using a controlled environment. However, when the lighting environment starts to become complex, problems in its replications in the 3D world are observable. These problems seem to stem

from the definitions of rendering parameters, virtual surface texture, the lighting environment, and the virtual image generation.

- **The use of photorealism concepts in metrology applications.**

Photorealism is an art movement created in the late 1960s and early 1970s in the United States, which consists of reproducing a photograph in other media such as paintings or drawings in as realistic a manner as possible. This art movement is already known in the scientific community as a tool for generating more realistic scenes and is used in the video game and film industries, but also in Industry V4.0, in digital twins to meet different requirements.

These requirements can be about continuous monitoring procedures to detect the level of deterioration of architectural structures, or modelling realistic scenes and pictures for geolocation problems, connected and automated vehicle, and networks training. However multi-camera network digital twins/models do not use photorealism technologies for their designs. Generally based on creating a whole virtual camera from its sensor to its radiometric characteristics, they are time-power consuming and is not necessarily accurate.

In this work, due to the lack of realism in the development of the digital model, and the use of a 3D animation software, photorealism concepts were used to develop a more photorealistic digital model of the real target scene. So, the novelty of this work, is the utilisation of these concepts to design a more realistic digital metrology tool.

- **The development of a set of tools to ensure the traceability of the digital model in Blender.**

Regarding the technology used (cameras), and the output of the system (images), the DFT seems to be a suitable tool to use. Applying DFT to images is not novel but applying it to the Blender images is. Its utilisation allows the understanding of the impact of the modification of the virtual camera parameters, and to quantify the differences/similarities with the real model.

In addition, using a no-reference image quality assessment algorithm is also not novel, but its application to the Blender images, and the image used as reference (MATLAB image) is. This method was used to quantify the blur percentage between a reference image and the virtual/real images. Through the results, an understanding of the impact of the modification of the virtual camera parameters, as well as a quantification of the differences/similarities with the real model was achieved.

Finally, the subtraction between real and virtual responses is novel, showing the residuals generated by differences in configuration, camera model, and modelling parameters. However, this "tool" should be used with caution, due to the potential non-equivalence between the two environments, which is a source of error, and should be analysed after a prior study of the systems and potential sources of errors.

8.3. Further work

In this research, digital models of camera measurement system were generated. However, considering the needs of Industry 4.0, these models need to evolve into digital twins by automating the two communication loops between the real and virtual models, building a network and using deep learning architectures.

Nevertheless, building a digital twin is not enough to meet the challenges of tomorrow. As explained in Chapter 1, the aim of the digitisation process is not simply to automate factories, but to create agile, resilient, fast and knowledgeable manufacturing enterprises that can draw on resources as needed to get things done, while reducing risk and maximising profits and business opportunities - from the shop floor to the management structures [15-17]. The system developed must therefore be flexible, agile and adaptable to different situations.

Currently, this is not the case, and research is still needed to improve these models. Furthermore, very limited work has been completed on this subject before, and this research can therefore be considered as a pioneer in the use of 3D animation software in the design of digital twins, for simulating metrology tools. In the light of all the chapters and their conclusions, the following lines of further research can be identified:

- **The creation of a realistic virtual illumination environment.**

In Blender, the illumination environment is composed of different types of light (point, rectangular, spot, and sun), with different parameters for each (such as colour, power, size). In this research, the modelled lights were designed with the real light source dimension and power, but the colour spectrum of the lights were not taken into consideration.

A trial to simulate sunlight was attempted in Chapter 7, and has initially demonstrated that, real sunlight and the virtual equivalent light may be different. These differences might come from the colour of the light, and location in the scene. The colour of the sunlight can be changed in the node system in the *Shading* window. However, the location of the sunlight in the scene may well be challenging. No substantive literature was found on this problem, and a rapid process of trial and error was attempted but only with limited success. A starting point of further investigation would be a deeper literature review on how sunlight is simulated in Blender, before trying to implement simulation methods.

The characterisation of the scene illumination spectrum might also be a good starting point to understand the differences between the real and virtual illumination environments. As for the object texture surface definition, it seems that the light is not defined in the same way between the real and the 3D world. Real world sunlight is a spread of wavelengths, whilst Blender defines it as an object with an illumination power in W/s^2.

- **Finding a correlation between real and virtual surface textures.**

As explained in Chapter 3, in Blender, object surfaces are generally defined by the BSDF (bidirectional scattering distribution) principle, which is based on light behaviours calculated for each type of texture existing in the real world, for a more realistic rendering. However, it is an approximation of the physics driven by artistic and visual needs. It is therefore difficult to correlate the real values of the surface

texture with their equivalent in the 3D world. A starting point of research on this topic could be to check whether other 3D animation software generate surface texture in the same way as Blender, before trying to develop a Python script simulating the BSDF principle from a physical point of view potentially, removing the problem. In addition, the Python script could be defined afterward as a Blender add-on.

- **The calibration of the digital camera.**

Due to the direction chosen in this research, the calibration of the virtual camera was initiated but identified key issues. A calibration process was completed in the real world, to determine the camera parameters (extrinsic and intrinsic) by using the checkerboard, and zone circles board. These parameters were then used to set up manually the positions and orientations of the virtual cameras in Blender.

The assumptions at the beginning of the thesis were that Blender was a perfect environment, and that it was possible to generate a digital twin or model identical to the real target. However, through this research, it was shown that Blender is not perfect, that a budget of error exists, and that it is difficult to generate a digital model similar to the real target. In addition, it was also proved that the real and virtual worlds have two different budgets of error, with similar sources of error, but also some intrinsic ones.

Regarding all these elements, the full calibration of the virtual cameras can potentially be achieved by defining all of the sources of errors linked to the camera, examining the script generating the camera in the Blender library (if accessible), and characterising the virtual sensor to understand how Blender generates picture, and determine all the sources of errors linked to it.

- **The automatization of the system.**

The automatization of the system can be explored through the generation of Python scripts to create the virtual environment, taking pictures from the virtual cameras, and so on. However, the output of this research is a digital model that needs to meet the requirement of the digitalisation trend of Industry 4.0. To do this, the connection between the virtual model and its real representation needs to be achieved automatically, and not manually as is currently the case. In order to connect and have an exchange of data, automation processes such as designing a network based on the internet network, or microcontroller can be used. Nevertheless, a time gap would exist between the real and virtual block of the digital twin. In order to have a connection and exchange of data in real time, potentially artificial intelligence or machine learning architectures may be explored.

- **The zone circle detection script.**

In MATLAB, a script has been written to find the centre of the zone circles board. However, the detection is sensitive to light and image contrast. In order to have a more robust script, techniques from diverse fields such as machine learning with classification, or detection methods can be used to improve the detection, and make the script less sensitive.

- **Utilisation of other 3D animation software**

Other 3D animation software exists such as Autodesk 3ds Max, Autodesk Maya, Cinema 4D, Houdini and Unity. The problem with these programs is that they master one field such as modelling, 3D rendering or animations, and they are not open source and free products. Other, free and open-source software exists but they are typically used for 2D modelling, except K- 3D, which could be an alternative to Blender.

However, the use of 3D animation to model metrology applications is still an opportunity gap, and perhaps the use of 3D animation specialising in 3D rendering will provide a better digital foundation than Blender. In addition, the fact that the majority of software is commercial may also ensure better and constant development of current and new tools than open-source software, with better support and assistance in case of problems.

The development of a digital model of a multi-camera measurement system in another 3D animation software would be based on the same methodology used in this research, but adapted to the new software used.

- **Scale up Model a factory environment and multi camera localisation**

In this work, the last step was about modelling a more complex environment (meeting room), which is clearly different and still less complicated, compared to a factory environment. Usually, the factory environment is composed of a light environment (which can be simple or complex), people walking around (working on machines), multiple machines, and other workplace objects.

To scale up and to allow us to model a factory environment, significant steps need to be considered, and the sources of error have to be identified for each step:

- Modelling different light environment correctly, going from the simple, to the more complex.
- Modelling the light environment of the factory.
- Modelling the factory environment.
- Modelling multiple small objects in the environment of the factory to test the object detection script.
- Modelling multiple medium objects in the environment of the factory.
- Modelling multiple large objects in the environment of the factory.
- Modelling multiple size of objects in the environment of the factory.
- Coding human tracking scripts.
- Modelling people in different positions (such as walking, sitting down, learned) in the environment of the factory without any objects to test the human tracking script.
- Modelling the factory environment, with humans and objects.

For each step, pictures from the virtual cameras have to be taken to detect any issues related to different sources of error such as the illumination and object light interactions. In addition, if possible, the virtual model has to be compared to the real one, using the toolbox developed in this research.

The results of the real and virtual factory environments should be compared. The results of this comparison will help to improve the virtual replication of the factory until the customer's requirements, or the success parameters defined at the beginning of the project, are met.

In the industrial environment, it is necessary to identify the placement constraints related to the angle of incidence of the camera, possible occlusions due to machines and people working in the factory, the target and the defined measurement volume. Once the constraints have been identified, the digital model can be generated (regarding the bullet points above) and different camera configurations simulated using algorithms such as Loopless Algorithms [292], and their reprojection errors measured. The final result would then be used to place the real cameras, and perform the actual measurement of the target in the measurement volume.

- **Use other types of cameras**

An attempt was made to use an alternative camera model with a NIKON D3100. This camera was chosen because Blender has a preset for this model in its default library. The experiment consisted of measuring the deviation of the diameter of the 0.09 m sphere with a single camera, following the same methodology as Experiment 1 in Chapter 4. Two lenses, NIKON 3.5 mm/ F 1.8, and the AF-S G NIKKOR DX-35 mm, were used to change the focal length of the camera in both environments (real and virtual).

Through these experiments, it was observed that:

- The auto focus of the camera needed to be removed, to manually adjust the camera focal length and F-stop.
- Even in manual mode, the camera still tended to automatically adjust the focal length and F-stop for object distance range used.
- The image taken in the controlled room was too bright, and the object could not be detected in MATLAB.

The auto adjustment of these parameters does not allow continuous values to be maintained for each distance, as well as a full digitisation of this camera model in Blender. In addition, the light problem in the image did not allow any measurement of the sphere diameter.

Possible solutions to these problems would be to change the camera's white balance, and ISO (the camera's sensitivity to light), as well as using a lens and a polarising filter to reduce the light entering the shutter. However, these options may not be available in the Blender, or not under the same name.

This experience also shows that cameras used for photography may not be adaptable to industrial applications due to their complexity and coding to find the optimal settings for the majority of use cases. Industrial machine vision camera should be explored to replace the Raspberry Pi.

- **Multi objects detection in the same image**

In this research, only the detection of a single object was considered. However, in a factory environment, the detection of one or more objects is necessary to perform measurements or monitoring. This problem is not new, and has already been studied by other researchers, providing a large body of literature, as well as solutions already developed and tested.

For the evolution of this research, a variety of techniques from different fields can be considered, such as machine learning with classification methods, or CNN (Convolutional Neural Network), or detection methods based on Python or MATLAB scripts. Furthermore, these techniques can be merged with the detection script already written in MATLAB, in order to improve it and to have a starting base.

- **Better understanding of pixel resolution in Blender and how camera models are defined**

As shown in Chapter 4, the pixel resolution between the real and the Blender images are different. The pixel resolution was measured between the real and virtual image taken by the final camera model. It was noticed that a difference still existed but was smaller than with the basic camera model. The camera parameters can have an impact on the pixel resolution. In addition, the definition of the rendering parameters can also have an impact, as they define the number of samples used for light management, colour management, denoising, for instance.

The question about pixel resolution can be extended to the image construction. In Chapter 7, it was noticed, in Experiment 2, that the virtual image was different compared to the real one. However, this virtual rendering was also different to the image view by the camera. It seems that the virtual image is defined as the image to be seen by the viewer, not the image seen by the camera.

Camera models in Blender are defined by different camera parameters such as focal length, or focus distance. However, even if the laws of physics are respected, very little research has been published on how the camera model is originally defined in Blender, and how the physics is actually simulated with the Blender code.

- **Better understanding the error budget in the real and virtual environments, and coding the real error budget of in the virtual environment**

The error budget is a list of every possible source of random and systematic errors with identification as to their classification, absolute values, and probability distributions [293].

Identifying all the sources of error in the real environment is difficult, but identifying the principal ones is possible through experiments as shown in this work, literature, and analysis of the elements composing the real world and their possible impact.

Once the real-world error budget is drafted (and its principal components understood and quantified), it can then be implemented in the virtual replication. This implementation will have the impact of degrading the virtual environment, which is considered to be initially perfect, and can be achieved by using the concepts of photorealism by defining new key elements, such as the characterisation of noise in the real environment, or the characterisation of the spectrum of the environment. However, this still depends on Blender having software options to be able to correct model the real-world error components.

8.4. Final remarks

This research has shown that developing a digital twin for multi-camera optical system is not an easy task, but is possible in a virtual environment such as Blender (although with varying levels of correlation between the virtual and real worlds). The reproduction of reality requires thinking about various elements that are not obvious, such as light-object interactions. Furthermore, it still seems difficult to accurately reproduce the real environment in the digital world due to the limitations of technology and humans.

The use of 3D animation software offers opportunity for, saving time on the creation of the camera model, thanks to the availability of pre-built camera models. However, it also poses problems related to the definition of its programming. Blender is a software originally designed to meet the needs of artists, and film markets. By market choice, some options such as surface texture, do not adequately simulate real physics but a compromise between physics and artistic expectations is achieved.

In this work, the marriage between an artistic movement and technologies has shown that it is possible to push further the limits of our understanding, and of what we can do. In addition, this research is a pioneer in the development of a multi-camera measurement system in a virtual environment, and offers the potential to model and simulate systems in more complex environment. When measurement systems can be simulated accurately in virtual environment (such as Blender) with respect to their real peers, then more rapid position optimisation and camera deployment may be possible.

References

[1] Reinganum, J.F., 1985. Innovation and industry evolution. The Quarterly Journal of Economics, 100(1), pp.81-99.

[2] Shane, S. ed., 2009. The handbook of technology and innovation management. John Wiley & Sons.

[3] Harrison, J., 2012. Occupational safety and health in the United Kingdom: securing future workplace health and wellbeing, Industrial Health, 50(4), pp. 261–266.

[4] Rigby, D., 2009. Management tools. Bain and Company.

[5] Hofmann, D.A., Burke, M.J. and Zohar, D., 2017. 100 years of occupational safety research: From basic protections and work analysis to a multilevel view of workplace safety and risk. Journal of Applied Psychology, 102(3), p.375.

[6] Ghobakhloo, M., 2020. Industry 4.0, digitization, and opportunities for sustainability. Journal of Cleaner Production, 252, p.119869.

[7] Lasi, H., Fettke, P., Kemper, H.G., Feld, T. and Hoffmann, M., 2014. Industry 4.0. Business & Information Systems Engineering, 6(4), pp.239-242.

[8] Fischer, C., Goller, M., Brinkmann, L. and Harteis, C., 2018. Digitalisation of work: Between affordances and constraints for learning at work. In Digital Workplace Learning (pp. 227-249). Springer, Cham.

[9] Jones, D., Snider, C., Nassehi, A., Yon, Y., Hicks, B., 2020. Characterising the Digital Twin: A systematic literature review, CIRP, Journal of Manufacturing Science and Technology, 29, pp. 36–52.

[10] Semeraro, C., Lezoche, M., Panetto, H., Dassisti, M., 2021. Digital twin paradigm: A systematic literature review, Computers in Industry, 130, p. 103469.

[11] Haratian, R. (2022) Motion capture sensing technologies and techniques: A sensor agnostic approach to address wearability challenges, Sensing and Imaging, 23(1), pp. 1–21.

[12] Van der Kruk, E. and Reijne, M.M., 2018. Accuracy of human motion capture systems for sport applications; state-of-the-art review. European Journal of Sport Science, 18(6), pp.806-819.

[13] Das, A., Chakraborty, A. and Roy-Chowdhury, A.K., 2014, September. Consistent re-identification in a camera network. In European conference on computer vision (pp. 330-345). Springer, Cham.

[14] Salierno, G., Leonardi, L. and Cabri, G., 2021. The Future of Factories: Different Trends, Applied Sciences, vol. 11, pp 9980, 11(21), pp. 9980.

[15] (2022) Rethinking Digital Transformation: More Than a Smart Factory, Rethinking Digital Transformation: More Than a Smart Factory, https://www.ibaset.com/, access data: 12th of January 2023

[16] Ghobakhloo, M. and Fathi, M., 2019. Corporate survival in Industry 4.0 era: the enabling role of lean-digitized manufacturing. Journal of Manufacturing Technology Management, vol. 31 no. 1, pp. 1-30.

[17] Sjödin, D., Parida, V., Kohtamäki, M. and Wincent, J., 2020. An agile co-creation process for digital servitization: A micro-service innovation approach. Journal of Business Research, 112, pp.478-491.

[18] Du, J.C. and Duffy, V.G., 2007. A methodology for assessing industrial workstations using optical motion capture integrated with digital human models. Occupational Ergonomics, 7(1), pp.11-25.

[19] Nikolakis, N., Alexopoulos, K., Xanthakis, E. and Chryssolouris, G., 2019. The digital twin implementation for linking the virtual representation of human-based production tasks to their physical counterpart in the factory-floor. International Journal of Computer Integrated Manufacturing, 32(1), pp.1-12.

[20] Fanton, J.P., 2019. A brief history of metrology: past, present, and future. International Journal of Metrology and Quality Engineering, 10, p.5.

[21] Alonso, V., Dacal-Nieto, A., Barreto, L., Amaral, A. and Rivero, E., 2019. Industry 4.0 implications in machine vision metrology: an overview. Procedia Manufacturing, 41, pp.359-366.

[22] Seneviratne, D., Ciani, L., Catelani, M. and Galar, D., 2018. Smart maintenance and inspection of linear assets: An Industry 4.0 approach. Acta Imeko.

[23] Alvarez-Meaza, I., Borregan-Alvarado, J., Cilleruelo-Carrasco, E. and Rio-Belver, R.M., 2021, February. Advanced Manufacturing or Industry 4.0 scholarly works: Are they relevant to technology development?. In International Joint Conference on Industrial Engineering and Operations Management (pp. 349-358). Springer, Cham.

[24] (2017) 'Metrologie für die Digitalisierung von Wirtschaft und Gesellschaft', https://www.ptb.de/cms/fileadmin/internet/publikationen/ptb_mitteilungen/mitt2017/Heft4/PTB-Mitteilungen_2017_Heft_4.pdf#page=36, access data: 5th of January 2023

[25] Li, H., Liu, G., Wang, H., Wen, X., Xie, G., Luo, G., Zhang, S. and Yang, M., 2021. Mechanical movement data acquisition method based on the multilayer neural networks and machine vision in a digital twin environment. Digital Twin, 1(6), p.6.

[26] Nikolakis, N., Alexopoulos, K., Xanthakis, E. and Chryssolouris, G., 2019. The digital twin implementation for linking the virtual representation of human-based production tasks to their physical counterpart in the factory-floor. International Journal of Computer Integrated Manufacturing, 32(1), pp.1-12.

[27] Harris, L. E., 2004, New cloud animation software on the horizon, Computing in Science and Engineering, 6(3), pp. 6–7.

[28] Papathomas, T. V., Schiavone, J. A. and Julesz, B., 1987, Stereo Animation, IEEE Computer Graphics and Applications, 7(9), pp. 18–27.

[29] Thalmann, D. and Magnenat-Thalmann, N., 1986, Artificial intelligence in three-dimensional computer animation, Computer Graphics Forum, 5(4), pp. 341–348.

[30] Grieves, M., 2002, October. Completing the Cycle: Using PLM Information in the Sales and Service Functions [Slides]. In SME Management Forum.

[31] Difference Between Digital Twin, Digital Model, and Digital Shadow, https://www.wizata.com/knowledge-base/difference-between-digital-twin-digital-model-and-digital-shadow, access data: 18th of January 2023.

[32] Wei, J., Thomas, A., Li, G. and Pattabiraman, K., 2014, June. Quantifying the accuracy of high-level fault injection techniques for hardware faults. In 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (pp. 375-382). IEEE.

[33] Mehrabi, M.G., Ulsoy, A.G. and Koren, Y., 2000, Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11(4): 403-419.

[34] Setchi, R.M. and Lagos, N., 2004, Reconfigurability and reconfigurable manufacturing systems: state-of-the-art review. In 2nd IEEE International Conference on Industrial Informatics. INDIN'04 (pp. 529-535). IEEE.

[35] Michalos, G., Makris, S., Chryssolouris, G., 2015, The new assembly system paradigm, International Journal of Computer Integrated Manufacturing, 28(12), 1252-1261.

[36] Jovane, F., Koren, Y. and Boer, C.R., 2003. Present and future of flexible automation: towards new paradigms. CIRP Annals, 52(2), pp.543-560.

[37] Meredith, J.R., 1987. The strategic advantages of the factory of the future. California Management Review, 29(3), pp.27-41.

[38] Frank, A.G., Dalenogare, L.S. and Ayala, N.F., 2019. Industry 4.0 technologies: Implementation patterns in manufacturing companies. International journal of production economics, 210, pp.15-26.

[39] Liu, Z., Xie, K., Li, L. and Chen, Y., 2020. A paradigm of safety management in industry 4.0. Systems Research and Behavioral Science, 37(4), pp.632-645.

[40] Bragança, S., Costa, E., Castellucci, I. and Arezes, P.M., 2019. A brief overview of the use of collaborative robots in industry 4.0: human role and safety. Occupational and environmental safety and health, Studies in Systems, Decision and Control, vol 202. Springer, Cham.

[41] Muehlmann, U., Ribo, M., Lang, P. and Pinz, A., 2004, April. A new high speed CMOS camera for real-time tracking applications. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 (Vol. 5, pp. 5195-5200). IEEE.

[42] Gawande, U., Hajari, K. and Golhar, Y., 2020. Pedestrian detection and tracking in video surveillance system: issues, comprehensive review, and challenges. In Recent Trends in Computational Intelligence; Intech Open Publisher: London, UK.

[43] A. Shen, T.T., 2014. Marker-less motion capture for biomechanical analysis using the kinect sensor," B.S. thesis, Universitat Politecnica de Catalunya.

[44] Yabukami, S., Kikuchi, H., Yamaguchi, M., Arai, K.I., Takahashi, K., Itagaki, A. and Wako, N., 2000. Motion capture system of magnetic markers using three-axial magnetic field sensor. IEEE Transactions on Magnetics, 36(5), pp.3646-3648.

[45] Vicon.com, What is motion capture, https://www.vicon.com/about-us/what-is-motion-capture/, access data: 7th of September 2022.

[46] Nikon.com, https://www.nikonmetrology.com/images/brochures/kscan-mmdx-kcmm-en.pdf, access data: 13th of September 2022.

[47] Petrosyan, T., Dunoyan, A. and Mkrtchyan, H., 2020. Application of Motion capture systems in ergonomic analysis. Armenian Journal of Special Education, 1(1), pp.107-117.

[48] Drouot, A., Zhao, R., Irving, L. and Ratchev, S., 2019. Towards industry 4.0: the future automated aircraft assembly demonstrator. In Precision Assembly in the Digital Age: 8th IFIP WG 5.5 International Precision Assembly Seminar, IPAS 2018, Chamonix, France, January 14—16, Revised Selected Papers 8 (pp. 169-182). Springer International Publishing.

[49] Mündermann, L., Corazza, S. and Andriacchi, T.P., 2006. The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. Journal of Neuroengineering and Rehabilitation, 3(1), pp.1-11.

[50] Muralikrishnan, B., Phillips, S. and Sawyer, D., 2016. Laser trackers for large-scale dimensional metrology: A review. Precision Engineering, 44, pp.13-28.

[51] Lau, K., Hocken, R. and Haynes, L., 1985. Robot performance measurements using automatic laser tracking techniques. Robotics and Computer-Integrated Manufacturing, 2(3-4), pp.227-236.

[52] Wang, L., Muralikrishnan, B., Hernandez, O.I., Shakarji, C. and Sawyer, D., 2020. Performance evaluation of laser trackers using the network method. Measurement, 165, p.108165.

[53] Conte, J., Majarena, A.C., Aguado, S., Acero, R. and Santolaria, J., 2016. Calibration strategies of laser trackers based on network measurements. The International Journal of Advanced Manufacturing Technology, 83, pp.1161-1170.

[54] Donati, S., 2004. Electro-optical instrumentation: sensing and measuring with lasers. Prentice Hall, Upper Saddle River, NJ, Sect. 4.5.2.

[55] Villa, F., Severini, F., Madonini, F. and Zappa, F., 2021. SPADs and SiPMs arrays for long-range high-speed light detection and ranging (LiDAR). Sensors, 21(11), p.3839.

[56] Fersch, T., Weigel, R. and Koelpin, A., 2017. A CDMA modulation technique for automotive time-of-flight LiDAR systems. IEEE Sensors Journal, 17(11), pp.3507-3516.

[57] Wei, X., Phung, S.L. and Bouzerdoum, A., 2011, June. Pedestrian sensing using time-of-flight range camera. In CVPR 2011 WORKSHOPS (pp. 43-48). IEEE.

[58] Cui, Y., Schuon, S., Chan, D., Thrun, S. and Theobalt, C., 2010, June. 3D shape scanning with a time-of-flight camera. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 1173-1180). IEEE.

[59] Buttgen, B. and Seitz, P., 2008. Robust optical time-of-flight range imaging based on smart pixel structures. IEEE Transactions on Circuits and Systems I: Regular Papers, 55(6), pp.1512-1525.

[60] Liu, J., Sun, Q., Fan, Z. and Jia, Y., 2018, September. TOF lidar development in autonomous vehicle. In 2018 IEEE 3rd Optoelectronics Global Conference (OGC) (pp. 185-190). IEEE.

[61] Whyte, R., Streeter, L., Cree, M.J. and Dorrington, A.A., 2015. Application of lidar techniques to time-of-flight range imaging. Applied Optics, 54(33), pp.9654-9664.

[62] Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D., 2012, October. A benchmark for the evaluation of RGB-D SLAM systems. In 2012 IEEE/RSJ International Conference on Intelligent robots and Systems (pp. 573-580). IEEE.

[63] Giancola, S., Corti, A., Molteni, F. and Sala, R., 2017. Motion capture: an evaluation of kinect V2 body tracking for upper limb motion analysis. In Wireless Mobile Communication and Healthcare: 6th International Conference, MobiHealth 2016, Milan, Italy, November 14-16, 2016, Proceedings 6 (pp. 302-309). Springer International Publishing.

[64] Moen, T.S., 2014. Evaluation of a Markerless motion capture system as a tool for sports movement analysis-implications for ACL injury risk assessment (Master's thesis), Universitetsbiblioteket i Oslo.

[65] Han, J., Shao, L., Xu, D. and Shotton, J., 2013. Enhanced computer vision with Microsoft Kinect sensor: A review. IEEE transactions on Cybernetics, 43(5), pp.1318-1334.

[66] Wu, H.H. and Bainbridge-Smith, A., 2011. Advantages of using a Kinect Camera in various applications. University of Canterbury, New Zeeland, pp.1-4.

[67] Wang, Q., Kurillo, G., Ofli, F. and Bajcsy, R., 2015, October. Evaluation of pose tracking accuracy in the first and second generations of Microsoft Kinect. In 2015 International Conference on Healthcare Informatics (pp. 380-389). IEEE.

[68] Drouot, A., Zhao, R., Irving, L. and Ratchev, S., 2019. Towards industry 4.0: the future automated aircraft assembly demonstrator. In Precision Assembly in the Digital Age: 8th IFIP WG 5.5 International Precision Assembly Seminar, IPAS 2018, Chamonix, France, January 14—16, 2018, Revised Selected Papers 8 (pp. 169-182). Springer International Publishing.

[69] Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P. and Savatier, X., 2017.

A study of Vicon system positioning performance. Sensors, 17(7), p.1591.

[70] Burge, J.H., Su, P., Zhao, C. and Zobrist, T., 2007, September. Use of a commercial laser tracker for optical alignment. In Optical System Alignment and Tolerancing (Vol. 6676, pp. 132-143). SPIE.

[71] Liu, J., Sun, Q., Fan, Z. and Jia, Y., 2018, September. TOF lidar development in autonomous vehicle. In 2018 IEEE 3rd Optoelectronics Global Conference (OGC) (pp. 185-190). IEEE.

[72] Li, N., Ho, C.P., Xue, J., Lim, L.W., Chen, G., Fu, Y.H. and Lee, L.Y.T., 2022. A progress review on solid‐state LiDAR and nanophotonics‐based LiDAR sensors. Laser & Photonics Reviews, 16(11), p.2100511.

[73] Sahu, R.K., 2021. A review on application of laser tracker in precision positioning metrology of particle accelerators. Precision Engineering, 71, pp.232-249.

[74] Mojica, G.A.R. and Tabago, S.N., The Camera.

[75] Housel, R. ed., 2009. From camera lens to critical lens: a collection of best essays on film adaptation. Cambridge Scholars Publishing.

[76] Banks, M.S., Cooper, E.A. and Piazza, E.A., 2014. Camera focal length and the perception of pictures. Ecological Psychology, 26(1-2), pp.30-46.

[77] Surya, G. and Subbarao, M., 1993, June. Depth from defocus by changing camera aperture: A spatial domain approach. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (pp. 61-67). IEEE.

[78] Kosloff, T.J. and Barsky, B.A., 2007. An algorithm for rendering generalized depth of field effects based on simulated heat diffusion. In Computational Science and Its Applications–ICCSA 2007: International Conference, Kuala Lumpur, Malaysia, August 26-29, 2007. Proceedings. Part III 7 (pp. 1124-1140). Springer Berlin Heidelberg.

[79] Simon, G., Vakulya, G. and Rátosi, M., 2022. The way to modern shutter speed measurement methods: A historical overview. Sensors, 22(5), p.1871.

[80] Photographylife, What is Noise in Photography, https://photographylife.com/what-is-noise-in-photography#what-is-noise, access date: 17th of January 2022.

[81] Scholer, H., 1975. On photogrammetric distortion. American Society of Photogrammetry, Journal of, 41(6), p. 761-769.

[82] Weng, J., Cohen, P. and Herniou, M., 1992. Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(10), pp.965-980.

[83] Mihas, P. and Andreadis, P., 2005. A historical approach to the teaching of the linear propagation of light, shadows and pinhole cameras. Science & Education, 14(7-8), pp.675-697.

[84] Guo, K., Ye, H., Gu, J. and Chen, H., 2021. A novel method for intrinsic and extrinsic parameters estimation by solving perspective-three-point problem with known camera position. Applied Sciences, 11(13), p.6014.

[85] Bárány, B., Käenmäki, A. and Koivusalo, H., 2018. Dimension of self‐affine sets for fixed translation vectors. Journal of the London Mathematical Society, 98(1), pp.223-252.

[86] Blesgen, T., 2015. On rotation deformation zones for finite-strain Cosserat plasticity. Acta Mechanica, 226(7), pp.2421-2434.

[87] Simsek, H. and Özdemir, M., 2016. Generating hyperbolical rotation matrix for a given hyperboloid. Linear Algebra and Its Applications, 496, pp.221-245.

[88] Luhmann, T., 2010. Close range photogrammetry for industrial applications. ISPRS Journal of Photogrammetry and Remote Sensing, 65(6), pp.558-569.

[89] Lachat, E., Macher, H., Landes, T. and Grussenmeyer, P., 2015. Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling. Remote Sensing, 7(10), pp.13070-13097.

[90] Newhall, B., 1982. The history of photography (p. 167). New York: Museum of Modern Art.

[91] Hartley, R.I. and Sturm, P., 1997. Triangulation. Computer Vision and Image Understanding, 68(2), pp.146-157.

[92] Zhang, Z., 1998. Determining the epipolar geometry and its uncertainty: A review. International Journal of Computer Vision, 27, pp.161-195.

[93] Hata, K. and Savarese, S., 2017. Cs231a course notes 3: Epipolar geometry. http://web.stanford.edu/class/cs231a/course_notes/ 03-epipolar-geometry.pdf, access date: 5th of January 2023.

[94] Lee, D.S., Shan, J. and Bethel, J.S., 2003. Class-guided building extraction from Ikonos imagery. Photogrammetric Engineering and Remote Sensing, 69(2), pp.143-150.

[95] Chen, Y., Chen, Y. and Wang, G., 2019. Bundle adjustment revisited. arXiv preprint arXiv:1912.03858.

[96] Alsadik, B., 2017. Unified approach of Least Squares Adjustment–an application in 3D geomatics.

[97] Atkinson, K.B., 1996. Close Range Photogrammetry and Machine Vision Whittles Publishing. Latheronwheel, Scotland, UK.

[98] Long, L. and Dongri, S., 2019. Review of camera calibration algorithms. In Advances in Computer Communication and Computational Sciences: Proceedings of IC4S 2018 (pp. 723-732). Springer Singapore.

[99] Song, L., Wu, W., Guo, J. and Li, X., 2013, August. Survey on camera calibration technique. In 5th International conference on intelligent human-machine systems and cybernetics (Vol. 2, pp. 389-392). IEEE.

[100] Qi, W., Li, F. and Zhenzhong, L., 2010, May. Review on camera calibration. In Proceedings of the 2010 Chinese Control and Decision Conference, Xuzhou, China, 26–28 May 2010, pp. 3354–3358.

[101] Qin, L., Feipeng, D. and Heming, H., 2010, September. An Improved method of location of the circular target International Conference on Computational Aspects of Social Networks, Taiyuan, China, 2010, pp. 251-254.

[102] MathWorks, Camera Calibration, https://uk.mathworks.com/help/vision/camera-calibration.html, access data: 5th of October 2022.

[103] Ji, Q. and Zhang, Y., 2001. Camera calibration with genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 31(2), pp.120-130.

[104] Cauchick-Miguel, P., Kinga, T. and Davis, J., 1996. CMM verification: a survey. Measurement, 17(1), pp.1-16.

[105] Wang, Z., Wu, W., Xu, X. and Xue, D., 2007. Recognition and location of the internal corners of planar checkerboard calibration pattern image. Applied Mathematics and Computation, 185(2), pp.894-906.

[106] Yang, L., Normand, J.M. and Moreau, G., 2016, September. Practical and precise projector-camera calibration. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (pp. 63-70). IEEE.

[107] Gavin, H.P., 2019. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. Department of Civil and Environmental Engineering, Duke University, p 19.

[108] Lourakis, M.I., 2005. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. Foundation of Research and Technology, 4(1), pp.1-6.

[109] Polyak, B.T., 2007. Newton's method and its use in optimization. European Journal of Operational Research, 181(3), pp.1086-1096.

[110] Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[111] Clarkson, J.A., 1936. Uniformly convex spaces. Transactions of the American Mathematical Society, 40(3), pp.396-414.

[112] Miranian, L. and Gu, M., 2003. Strong rank revealing LU factorizations. Linear algebra and its applications, 367, pp.1-16.

[113] Weisstein, E.W., 2002. Least squares fitting. https://mathworld. wolfram. com/.

[114] Eberly, D., 2000. Least squares fitting of data. Chapel Hill, NC: Magic Software, pp.1-10.

[115] Hartley, R. and Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge University Press.

[116] Tao, F., Zhang, H., Liu, A. and Nee, A.Y., 2018. Digital twin in industry: State-of-the-art. IEEE Transactions on Industrial Informatics, 15(4), pp.2405-2415.

[117] Steil, T. and Roßmann, J., 2014, November. Validating the camera and light simulation of a virtual reality testbed by means of physical mockup data. 2nd International Conference on Artificial Intelligence, Modelling and Simulation, Madrid, Spain, 2014, pp. 143-148.

[118] Farrell, J.E., Xiao, F., Catrysse, P.B. and Wandell, B.A., 2003, December. A simulation tool for evaluating digital camera image quality. In Image Quality and System Performance (Vol. 5294, pp. 124-131). SPIE.

[119] Wright, L. and Davidson, S., 2020. How to tell the difference between a model and a digital twin. Advanced Modeling and Simulation in Engineering Sciences, 7(1), pp.1-13.

[120] Rasheed, A., San, O. and Kvamsdal, T., 2020. Digital twin: Values, challenges and enablers from a modeling perspective. IEEE Access, 8, pp.21980-22012.

[121] FutureLearn, 2022, What is 3D modelling and what is it used for?, https://www.futurelearn.com/info/blog/general/what-is-3d-modelling, access data: 17th October 2022.

[122] SIEMENS, 3D Modeling, https://www.plm.automation.siemens.com/global/en/products/plm-components/3d-modeling.html, access data: 22th October 2022.

[123] Ekaran. S., How to 3D model: the basics of 3D modeling, https://all3dp.com/2/3d-modeling-basics-simply-explained/, access data: 25th October 2022.

[124] Wang, X., Liang, C.J., Menassa, C. and Kamat, V., 2020, October. Real-time process-level digital twin for collaborative human-robot construction work. In Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC), Kitakyushu, Japan.

[125] Zhu, Z., Liu, C. and Xu, X., 2019. Visualisation of the digital twin data in manufacturing by using augmented reality. Procedia CIRP, 81, pp.898-903.

[126] Stączek, P., Pizoń, J., Danilczuk, W. and Gola, A., 2021. A digital twin approach for the improvement of an autonomous mobile robots (AMR's) operating environment—A case study. Sensors, 21(23), p.7830.

[127] Balit, E., Vaufreydaz, D. and Reignier, P., 2016, March. Integrating animation artists into the animation design of social robots an open-source robot animation software. In 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI) (pp. 417-418). IEEE.

[128] CB CREATIVE BLOQ, 2022, The best 3D modelling software in January 2023, https://www.creativebloq.com/features/best-3d-modelling-software, access date: 2nd of October 2022

[129] 3D sourced, The 7 Best 3D Animation Software 2022 (Some are Free!), https://www.3dsourced.com/3d-software/best-3d-animation-software/, access date: 4th of November 2022.

[130] Shanoon, C., 2023, Best 3D Animation Software Free and Paid, https://filmora.wondershare.com/animated-video/best-3d-animation-software-free-paid.html access date: 10th of November 2022.

[131] Wikipedia, Autodesk Maya, https://en.wikipedia.org/wiki/Autodesk_Maya, access date: 14th of November 2022.

[132] Wikipedia, Autodesk 3ds MAX, https://en.wikipedia.org/wiki/Autodesk_3ds_Max, access date: 16th of November 2022.

[133] Fremox, 2017, Quel Logiciel 3D Choisir ? (1/4), https://motion-cafe.com/quel-logiciel-3d-choisir-01, access date: 19th of November 2022.

[134] ROCKET BRUSH, 2022, MAYA VS BLENDER - WHAT TO CHOOSE IN 2022, https://rocketbrush.com/blog/maya-vs-blender-for-3d-game-artist, access date: 31st of November 2022

[135] Wikipedia, Blender, https://en.wikipedia.org/wiki/Blender_(software), access date: 23rd of November 2022

[136] Sculpteo, Blender 3D Modeling Software, https://www.sculpteo.com/en/glossary/blender-definition/, access date: 1st of December 2022.

[137] Zumolabs, 2021, Why Blender Is the Best Software for the 3D Workflow, https://www.zumolabs.ai/post/blender-is-the-best, access date: 2nd of December 2022.

[138] Wikipedia, Cinema 4D, https://en.wikipedia.org/wiki/Cinema_4D, access date: 25th of November 2022.

[139] Wikipedia, Houdini, https://en.wikipedia.org/wiki/Harry_Houdini, access date: 30th of November 2022.

[140] SideFX, Houdini | 3D Procedural Software for Film, TV & Gamedev | SideFX, https://www.sidefx.com/products/houdini/access date: 6th of December 2022.

[141] Reitmann, S., Neumann, L. and Jung, B., 2021. Blainder—a Blender AI add-on for generation of semantically labeled depth-sensing data. Sensors, 21(6), p.2144.

[142] Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A. and Katam, H., 2019. Blenderproc. arXiv preprint arXiv:1911.01911.

[143] Zhang, H., Kwan-yee, K.W. and Zhang, G., 2007. Camera calibration from images of spheres. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(3), pp.499-502.

[144] Zhang, H., Zhang, G. and Wong, K.Y., 2005, September. Camera calibration with spheres: linear approaches. In IEEE International Conference on Image Processing 2005 (Vol. 2, pp. II-1150). IEEE.

[145] Bulaha, N., 2015, June. Analysis of service properties of cylindrically ground surfaces, using standard ISO 25178-2: 2012 surface texture parameters. In ENVIRONMENT. TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference (Vol. 1, pp. 16-21).

[146] Söğütlü, C., Nzokou, P., Koc, I., Tutgun, R. and Döngel, N., 2016. The effects of surface roughness on varnish adhesion strength of wood materials. Journal of Coatings Technology and Research, 13, pp.863-870.

[147] Soudarissanane, S., Lindenbergh, R., Menenti, M., Teunissen, P. and Delft, T., 2009, September. Incidence angle influence on the quality of terrestrial laser scanning points, ISPRS. In Proceedings ISPRS Workshop Laser scanning (pp. 1-2).

[148] MathWorks, Camera Calibration, https://uk.mathworks.com/help/vision/camera-calibration.html, access date: 15th of December 2022.

[149] Adlaj, S., 2012. An eloquent formula for the perimeter of an ellipse. Notices of the AMS, 59(8), pp.1096-1097.

[150] Eltra Trade, 2020, What is Industrial camera? Main types and image processing, https://eltra-trade.com/blog/what-is-industrial-camera, access date: 14th of January 2024.

[151] GETCAMERAS, COMPUTER VISION CAMERA PRODUCTS, https://www.get-cameras.com/Computer-Vision-Camera-Products?utm_term=industrial%20camera&utm_campaign=GeT-Cameras.com+%233&utm_source=adwords&utm_medium=ppc&hsa_acc=8926064794&hsa_cam=20157046309&hsa_grp=149767263112&hsa_ad=658906833801&hsa_src=g&hsa_tgt=kwd-135264064&hsa_kw=industrial%20camera&hsa_mt=p&hsa_net=adwords&hsa_ver=3&gclid=EAIaIQobChMI-afdzdLZgAMVS-3tCh3AIQkuEAAYAyAAEgJFRfD_BwE, access date: 14th of January 2024.

[152] Zhang, Z., 2014. Camera Model. In: Ikeuchi, K. (eds) Computer Vision. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-31439-6_165

[153] Camera Models, http://www.cs.toronto.edu/~kyros/courses/418/Notes/Camera.pdf, access date: 14th of January 2024.

[154] Rahman, S.M., Liao, Z., Jiang, L. and Wang, Y., 2016, August. A regret-based autonomy allocation scheme for human-robot shared vision systems in collaborative assembly in manufacturing. In 2016 IEEE International Conference on Automation Science and Engineering (CASE) (pp. 897-902) IEEE.

[155] Blender stack Exchange, 2022, How Blender sets its reference frame, https://blender.stackexchange.com/questions/221105/how-blender-sets-its-reference-frame, access date: 6th of January 2022.

[156] Young, M., 1971. Pinhole optics. Applied Optics, 10(12), pp.2763-2767.

[157] Gallas, A.H., Gilbert, C.A. and Hitterdal, A.B., 1965. Pinhole optics and simulators. Journal of the SMPTE, 74(4), pp.321-323.

[158] Young, M., 1972. Pinhole imagery. American Journal of Physics, 40(5), pp.715-720.

[159] Elsayed, M., Sammani, F., Hamdi, A., Albaser, A. and Babalghoom, H., 2018. A new method for full reference image blur measure. Int J Simul Syst Sci Technol, 19(4).

[160] EDUCBA, Blender Tools, https://www.educba.com/blender-tools/, access date: 16th of December 2022.

[161] Blender 3.4 Manual, Modeling, https://docs.blender.org/manual/en/latest/modeling/index.html, access date: 19th of December 2022.

[162] Concept Art Empire, What is UV Mapping & Unwrapping?, https://conceptartempire.com/uv-mapping-unwrapping/, access date: 26th of December 2022.

[163] Blender 3.4 Manual, UV Editor, https://docs.blender.org/manual/en/latest/modeling/index.html, access date: 26th of December 2022.

[164] Seven, M., 2020, Overview of free Blender renderers, https://www.blendernation.com/2020/08/03/overview-of-free-blender-renderers/, access date: 28th of December 2022.

[165] Blender 3.4 Manual, Physics, https://docs.blender.org/manual/en/latest/physics/introduction.html, access date: 26th of December 2022.

[166] Blender, Requirements, https://www.blender.org/download/requirements/, access date: 1st of January 2022.

[167] Makeuseof, What Is the Difference Between a UV Sphere and an Icosphere in Blender?, access date: 3rd of January 2022.

[168] Blender 3.4 Manual, Light Objects https://docs.blender.org/manual/en/latest/render/lights/light_object.html, access date: 25th of January 2022.

[169] Blender 3.4 Manual, Cameras, https://docs.blender.org/manual/en/latest/render/cameras.html, access date: 20th of January 2022.

[170] Developer, Camera (sensor) Presets update, https://devtalk.blender.org/t/camera-sensor-presets-update/17974, access date: 20th of January 2022.

[171] reddit, Cinematic preset cameras?, https://www.reddit.com/r/blender/comments/74fyyi/cinematic_preset_cameras/, access date: 20th of January 2022.

[172] Blender 3.4 Manual, Cameras, Panoramic Cameras https://docs.blender.org/manual/en/latest/render/cameras.html, access date: 23th of January 2022.

[173] Dutta, S., 2021. Depth-aware blending of smoothed images for Bokeh effect generation. Journal of Visual Communication and Image Representation, 77, p.103089.

[174] Wu, J., Zheng, C., Hu, X., Wang, Y. and Zhang, L., 2010. Realistic rendering of Bokeh effect based on optical aberrations. The Visual Computer, 26, pp.555-563.

[175] ALL3DP, 2022, Blender: Geometry nodes – Simply explained, https://all3dp.com/2/blender-geometry-nodes-simply-explained, access date: 25th of January 2022.

[176] Blender 3.4 Manual, Math Node https://docs.blender.org/manual/en/latest/compositing/types/converter/math.html, access date: 25th of January 2022.

[177] Bidaj, K., Begueret, J.B. and Deroo, J., 2018. Jitter definition, measurement, generation, analysis, and decomposition. International Journal of Circuit Theory and Applications, 46(12), pp.2171-2188.

[178] Blender 3.4 Manual, Lens Distortion Node https://docs.blender.org/manual/en/latest/compositing/types/distort/lens_distortion.html, access date: 26th of January 2022.

[179] Kuan, D.T., Sawchuk, A.A., Strand, T.C. and Chavel, P., 1985. Adaptive noise smoothing filter for images with signal-dependent noise. IEEE Transactions on Pattern Analysis and Machine Intelligence, (2), pp.165-177.

[180] Boncelet, C., 2009. Image noise models. In The essential guide to image processing (pp. 143-167). Academic Press.

[181] Schulte, S., Nachtegael, M., De Witte, V., Van der Weken, D. and Kerre, E.E., 2006. A fuzzy impulse noise detection and reduction method. IEEE Transactions on Image Processing, 15(5), pp.1153-1162.

[182] Evermotion, Understanding the Noise Node in Blender, Understanding the Noise Node in Blender - Evermotion, access date: 26th of January 2022.

[183] Hecht, E., 2002. Optics, 5e. Pearson Education India.

[184] Blenderartisits, Coping with GI energy loss, especially with PBR, https://blenderartists.org/t/coping-with-gi-energy-loss-especially-with-pbr/680301/4, access date: 30th of January 2022.

[185] Varga, B.E., Gao, W., Laurik, K.L., Tátrai, E., Simó, M., Somfai, G.M. and Cabrera DeBuc, D., 2015. Investigating tissue optical properties and texture descriptors of the retina in patients with multiple sclerosis. Plos one, 10(11), p.e0143711.

[186] R&C Lighting, Candela Vs. LUX Vs. Lumens, https://rclite.com/blog/candela-vs-lux-vs-lumens/, access date : 31st of January 2022.

[187] Martin. J, 7 Ways to achieve realistic lighting in Blender, 7 Ways to Achieve Realistic Lighting in Blender | Blender Render Farm | Blendergrid, access date: 31st of January 2022.

[188] Burley, B. and Studios, W.D.A., 2012, August. Physically-based shading at Disney. In Acm Siggraph (Vol. 2012, pp. 1-7). vol. 2012.

[189] Blender 3.4 Manual, Principled BSDF https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html, access date: 23th of January 2022.

[190] Asmail, C., 1991. Bidirectional scattering distribution function (BSDF): a systematized bibliography. Journal of Research of the National Institute of Standards and Technology, 96(2), p.215.

[191] Grobe, L.O., 2019. Photon mapping in image-based visual comfort assessments with BSDF models of high resolution. Journal of Building Performance Simulation, 12(6), pp.745-758.

[192] Lee, E.S., Geisler-Moroder, D. and Ward, G., 2018. Modeling the direct sun component in buildings using matrix algebraic approaches: Methods and Validation. Solar Energy, 160, pp.380-395.

[193] Schiff, T.F., Stover, J.C., Cheever, D.R. and Bjork, D.R., 1989, April. Maximum and minimum limitations imposed on BSDF measurements. In Stray light and contamination in optical systems (Vol. 967, pp. 50-57). SPIE.

[194] Hanrahan, P. and Krueger, W., 1993, September. Reflection from layered surfaces due to subsurface scattering. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (pp. 165-174).

[195] Blender Guru, 2018, Comment Utiliser le Nouveau Shader "Ultime" de Blender: Le Principled BSDF, https://www.youtube.com/watch?v=4H5W6C_Mbck, access date: 21st of January 2022.

[196] Burley, B., 2015. Physically Based Shading in Theory and Practice-Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering. SIGGRAPH Courses.

[197] Pusztai, Z., Eichhardt, I. and Hajder, L., 2018. Accurate calibration of multi-lidar-multi-camera systems. Sensors, 18(7), p.2139.

[198] Kärrholm, A. and Rickman, L., Multi-Camera Multi-Person Tracking Using Reinforcement Learning.

[199] Fortini, L., Leonori, M., Gandarias, J.M. and Ajoudani, A., 2022. Open-vico: An open-source gazebo toolkit for multi-camera-based skeleton tracking in human-robot collaboration. arXiv preprint arXiv:2203.14733.

[200] Blender Guru, 2017, The Secret Ingredient to Photorealism, https://www.blenderguru.com/tutorials/secret-ingredient-photorealism, access date: 6th of January 2022.

[201] Olagoke, A.S., Ibrahim, H. and Teoh, S.S., 2020. Literature survey on multi-camera system and its application. IEEE Access, 8, pp.172892-172922.

[202] Konda, K.R. and Conci, N., 2013. Global and local coverage maximization in multi-camera networks by stochastic optimization. Infocommunications Journal, 5(1), pp.1-8.

[203] Sakane, S., Ish, M. and Kakikura, M., 1987. Occlusion avoidance of visual sensors based on a hand-eye action simulator system: HEAVEN. Advanced Robotics, 2(2), pp.149-165.

[204] Sakane, S., Niepold, R., Sato, T., Shirai, Y.: Illumination setup planning for a hand-eye system based on an environmental model. Advanced Robotics 6 (1991) 461–482

[205] Zhang, H., Eastwood, J., Isa, M., Sims-Waterhouse, D., Leach, R. and Piano, S., 2021. Optimisation of camera positions for optical coordinate measurement based on visible point analysis. Precision Engineering, 67, pp.178-188.

[206] Erdem, U.M. and Sclaroff, S., 2004, May. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In Omnivis2004, The Fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical cameras. Citeseer.

[207] Pinto, M., Dauvergne, D., Freud, N., Krimmer, J., Létang, J.M., Ray, C., Roellinghoff, F. and Testa, E., 2014. Design optimisation of a TOF-based collimated camera prototype for online hadrontherapy monitoring. Physics in Medicine & Biology, 59(24), p.7653.

[208] Bodor, R., Drenner, A., Schrater, P. and Papanikolopoulos, N., 2007. Optimal camera placement for automated surveillance tasks. Journal of Intelligent and Robotic Systems, 50, pp.257-295.

[209] Urban, S., Wursthorn, S., Leitloff, J. and Hinz, S., 2017. MultiCol bundle adjustment: a generic method for pose estimation, simultaneous self-calibration and reconstruction for arbitrary multi-camera systems. International Journal of Computer Vision, 121, pp.234-252.

[210] Ranganathan, A., 2004. The Levenberg-Marquardt algorithm. Tutoral on LM algorithm, 11(1), pp.101-110.

[211] Grimson, W.E., 1986. Sensing strategies for disambiguating among multiple objects in known poses. IEEE Journal on Robotics and Automation, 2(4), pp.196-213.

[212] Mittal, A. and Davis, L.S., 2008. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. International Journal of Computer Vision, 76(1), p.31.

[213] Erdem, U.M. and Sclaroff, S., 2006. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. Computer Vision and Image Understanding, 103(3), pp.156-169.

[214] Lagaros, N.D., 2010. Multicomponent incremental dynamic analysis considering variable incident angle. Structure and Infrastructure Engineering, 6(1-2), pp.77-94.

[215] Rigaud, E. and Le Bot, A., 2013. Influence of incidence angle on wear induced by sliding impacts. Wear, 307(1-2), pp.68-74.

[216] Agusanto, K., Li, L., Chuangui, Z. and Sing, N.W., 2003, October. Photorealistic rendering for augmented reality using environment illumination. In The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings. (pp. 208-216). IEEE.

[217] Meisel, L.K. and Harris, E.K.M., 2018. Photorealism in the digital age. New York: Harry N. Abrams.

[218] Nakamae, E. and Tadamura, K., 1995. Photorealism in computer graphics—Past and present. Computers & Graphics, 19(1), pp.119-130.

[219] Turnock, J.A., 2022. The Empire of effects: industrial light and magic and the rendering of realism. University of Texas Press.

[220] Abreu. R, 2021, What is CGI? How CGI works in movies and animation, https://www.studiobinder.com/blog/what-is-cgi-meaning-definition/, access date: 9th of December 2022.

[221] Sohaliya, G., & Sharma, K, 2021, August, An evolution of style transfer from artistic to photorealistic: a review. In 2021 Asian Conference on Innovation in Technology (ASIANCON) (pp. 1-7). IEEE.

[222] Suznjevic, M., Slivar, I., & Skorin-Kapov, L, 2016, Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of NVIDIA GeForce NOW. In 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX) (pp. 1-6). IEEE.

[223] Comaniciu, D., Engel, K., Georgescu, B., & Mansi, T, 2016, Shaping the future through innovations: From medical imaging to precision medicine. Medical Image Analysis, 33, 19-26.

[224] Vairo, A., Marro, M., De Ferrari, G. M., Rinaldi, M., & Salizzoni, S, 2019, Use of a photo‐realism 3D rendering technique to enhance echocardiographic visualization of the anatomical details during beating‐heart mitral valve repair. Echocardiography, 36(11), 2090-2093.

[225] Mania, P., & Beetz, M, 2019, A framework for self-training perceptual agents in simulated photorealistic environments. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 4396-4402).pho IEEE.

[226] Mahmood, F., Chen, R., Sudarsky, S., Yu, D., & Durr, N. J. (2018). Deep learning with cinematic rendering: fine-tuning deep neural networks using photorealistic medical images. Physics in Medicine & Biology, 63(18), 185012.

[227] Zaspel, P., & Griebel, M, 2011, Photorealistic visualization and fluid animation: Coupling of Maya with a two-phase Navier-Stokes fluid solver. Computing and Visualization in Science, 14(8), 371-383.

[228] Lehner, H., & Dorffner, L, 2020, Digital geoTwin Vienna: towards a digital twin city as geodata hub. PFG 88, 63–75.

[229] Wang, Z., Han, K., & Tiwari, P, 2021, Digital twin simulation of connected and automated vehicles with the unity game engine. In 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI) (pp. 1-4). IEEE.

[230] Alexopoulos, K., Nikolakis, N., & Chryssolouris, G, 2020, Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. International Journal of Computer Integrated Manufacturing, 33(5), 429-439.

[231] Steil, T., and Roßmann, J, 2014, Validating the camera and light simulation of a virtual reality testbed by means of physical mockup data. In 2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation (pp. 143-148). IEEE.

[232] Farrell, J.E., Xiao, F., Catrysse, P.B. and Wandell, B.A., 2003, December. A simulation tool for evaluating digital camera image quality. In Image Quality and System Performance (Vol. 5294, pp. 124-131). SPIE.

[233] Brito, A., 2008. Blender 3D: Architecture, buildings, and scenery: Create photorealistic 3D architectural visualizations of buildings, interiors, and environmental scenery. Packt Publishing Ltd.

[234] Georgiou, M., David, S., Loudos, G., Tsougos, I. and Georgoulias, P., 2012. Experimental and simulation studies for the optimization of dedicated scintimammography cameras. Journal of Instrumentation, 7(01), p.P01011.

[235] Doss, E., 1995. Benton, Pollock, and the politics of modernism: from regionalism to abstract expressionism. University of Chicago Press.

[236] Odom, N.G., 1981. The photograph and photo-realism in painting, Master thesis, Eastern Illinois University.

[237] Benjamin, W., 1972. A short history of photography. translated by P. Patton, in Classic Essays on Photography, edited by Alan Trachtenberg, 199-216 (New Haven, CT: Yale University Press).

[238] Meisel, L.K., 1980. Photo-realism. New York: Harry N. Abrams

[239] Invaluable, Photorealism and Its Impact on Contemporary Art, access date: 5th of December 2022.

[240] The arts story, Photorealism Movement Overview, https://www.theartstory.org/movement/photorealism/, access date: 7th of December 2022.

[241] Heckmann. C, 2021, What is realism in film? cinematic realism explained, https://www.studiobinder.com/blog/what-is-realism-in-film-definition/, access date: 12th of December 2022.

[242] Shi, L., Li, B., Kim, C., Kellnhofer, P., & Matusik, W, 2021, Towards real-time photorealistic 3D holography with deep neural networks. Nature, 591(7849), 234-239.

[243] Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S.N. and Guenter, B., 2019, September. Photorealistic image synthesis for object instance detection. In 2019 IEEE international conference on image processing (ICIP) (pp. 66-70). IEEE.

[244] Cui, Q., McIntosh, S., & Sun, H, 2018, Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs. Computer, Materials and Continua, 55(2), 229-241.

[245] Steam, What are “samples” ?, What are "samples"? :: Blender Discussions générales (steamcommunity.com), access date: 14th of December 2022.

[246] Blender Guru, 2012, 4 Easy Ways to Speed Up Cycles, 4 Easy Ways to Speed Up Cycles — Blender Guru, access date: 16th of December 2022.

[247] Burningham, N., Pizlo, Z., Allebach, Jan P, 2002, Image quality metrics. In Hornak, Joseph P. (ed.). Encyclopedia of Imaging Science and Technology. New York: Wiley.

[248] Silverstein, D.A. and Farrell, J.E., 1996, September. The relationship between image fidelity and image quality. In Proceedings of 3rd IEEE International Conference on Image Processing (Vol. 1, pp. 881-884). IEEE.

[249] Ferreira, W.D., Ferreira, C.B., da Cruz Júnior, G. and Soares, F., 2020. A review of digital image forensics. Computers & Electrical Engineering, 85, p.106685.

[250] Khanna, N., Mikkilineni, A.K., Chiang, P.J., Ortiz, M.V., Shah, V., Suh, S., Chiu, G.T.C., Allebach, J.P. and Delp, E.J., 2007, April. Printer and sensor forensics. In 2007 IEEE Workshop on Signal Processing Applications for Public Security and Forensics (pp. 1-8). IEEE.

[251] Davis, H., 2008. Practical artistry: light & exposure for digital photographers. O'Reilly Media, Inc.

[252] Crete, F., Dolmiere, T., Ladret, P. and Nicolas, M., 2007, February. The blur effect: Perception and estimation with a new no-reference perceptual blur metric. In Human Vision and Electronic Imaging XII (Vol. 6492, pp. 196-206). SPIE.

[253] MathWor, How do I calculate SNR from a single area of an image?, access date: 16th of June2022.

[254] Zhang, Y., 2008. Methods for image fusion quality assessment-a review, comparison and analysis. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37(PART B7), pp.1101-1109.

[255] Wang, Z. and Bovik, A.C., 2011. Reduced-and no-reference image quality assessment. IEEE Signal Processing Magazine, 28(6), pp.29-40.

[256] Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4), pp.600 612.

[257] Wang, Z., Bovik, A.C. and Lu, L., 2002, May. Why is image quality assessment so difficult?. In 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (Vol. 4, pp. IV-3313). IEEE.

[258] MathWorks, Image Quality Metrics, https://fr.mathworks.com/help/images/image-quality-metrics.html, access date: 18th of December 2022.

[259] Roetenberg, D., Luinge, H. and Slycke, P., 2009. Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors. Xsens Motion Technologies BV, Tech. Rep, 1, pp.1-7.

[260] Leisti, T., Radun, J., Virtanen, T., Halonen, R. and Nyman, G., 2009, January. Subjective experience of image quality: attributes, definitions, and decision making of subjective image quality. In Image Quality and System Performance VI (Vol. 7242, pp. 130-138). SPIE.

[261] Eskicioglu, A.M. and Fisher, P.S., 1995. Image quality measures and their performance. IEEE Transactions on Communications, 43(12), pp.2959-2965.

[262] Silverstein, D.A. and Farrell, J.E., 1996, September. The relationship between image fidelity and image quality. In Proceedings of 3rd IEEE International Conference on Image Processing, vol. 1, pp. 881-884). IEEE.

[263] Liu, Y., Yan, H., Gao, S., & Yang, K., 2018, Criteria to evaluate the fidelity of image enhancement by MSRCR. IET Image Processing, 12(6), 880-887.

[264] Zhou, G., Zhang, C., Li, Z., Ding, K. and Wang, C., 2020. Knowledge-driven digital Twin manufacturing cell towards intelligent manufacturing. International Journal of Production Research, 58(4), pp.1034-1051.

[265] Kritzinger, W., Karner, M., Traar, G., Henjes, J. and Sihn, W., 2018. Digital Twin in manufacturing: A categorical literature review and classification. Ifac-PapersOnline, 51(11), pp.1016-1022.

[266] Huang, T., Burnett, J. and Deczky, A., 1975. The importance of phase in image processing filters. IEEE Transactions on Acoustics, Speech, and Signal Processing, 23(6), pp.529-542.

[267] Heckbert, P., 1995. Fourier transforms and the fast Fourier transform (FFT) algorithm. Computer Graphics, 2, pp.15-463.

[268] Haynal, S. and Haynal, H., 2011. Generating and searching families of FFT algorithms. Journal on Satisfiability, Boolean Modeling and Computation, 7(4), pp.145-187.

[269] Fourier, J.B.J., 1822. Théorie analytique de la chaleur. Firmin Didot.

[270] Skarbnik, N., Zeevi, Y.Y. and Sagiv, C., 2009. The importance of phase in image processing. Technical Report; Technion—Israel Institute of Technology: Haifa, Israel, 2010; pp. 1–30.

[271] Oppenheim, A.V. and Lim, J.S., 1981. The importance of phase in signals. Proceedings of the IEEE, 69(5), pp.529-541.

[272] Wang, X., Tian, B., Liang, C. and Shi, D., 2008, May. Blind image quality assessment for measuring image blur. In 2008 Congress on Image and Signal Processing (Vol. 1, pp. 467-470). IEEE.

[273] Pardhu, T. and Perli, B.R., 2016, April. Digital image watermarking in frequency domain. In 2016 International Conference on Communication and Signal Processing (ICCSP) (pp. 0208-0211). IEEE.

[274] Crandall, R. and Fagin, B., 1994. Discrete weighted transforms and large-integer arithmetic. Mathematics of Computation, 62(205), pp.305-324.

[275] Baig, M.A., Moinuddin, A.A., Khan, E. and Ghanbari, M., 2022. DFT-based no-reference quality assessment of blurred images. Multimedia Tools and Applications, 81(6), pp.7895-7916.

[276] Chen, W.Y., 2008. Color image steganography scheme using DFT, SPIHT codec, and modified differential phase-shift keying techniques. Applied Mathematics and Computation, 196(1), pp.40-54.

[277] Pun, C.M., 2006, November. A novel DFT-based digital watermarking system for images. In 2006 8th International Conference on Signal Processing (Vol. 2). IEEE.

[278] Narwaria, M., Lin, W., McLoughlin, I.V., Emmanuel, S. and Chia, L.T., 2012. Fourier transform-based scalable image quality measure. IEEE Transactions on Image Processing, 21(8), pp.3364-3377.

[279] Zhang, S., Li, P., Xu, X., Li, L. and Chang, C.C., 2018. No-reference image blur assessment based on response function of singular values. Symmetry, 10(8), p.304.

[280] Fiete, R.D. and Tantalo, T., 2001. Comparison of SNR image quality metrics for remote sensing systems. Optical Engineering, 40(4), pp.574-585.

[281] MathWorks, Fourier Transform, https://uk.mathworks.com/help/images/fourier-transform.html, access date: 11th of December 2022.

[282] MathWorks, Practical Introduction to Frequency-Domain Analysis, https://uk.mathworks.com/help/signal/ug/practical-introduction-to-frequency-domain-analysis.html, access date: 13th of December 2022.

[283] Sripian, P. and Yamaguchi, Y., 2020. Hybrid image of three contents. Visual Computing for Industry, Biomedicine, and Art, 3(1), pp.1-8.

[284] Sripian, P. and Yamaguchi, Y., 2012, June. Shape-free hybrid image. In: Abstracts of symposium on non-photorealistic animation and rendering. Eurographics Association, Annecy, pp 11–19.

[285] Oliva, A., Torralba, A. and Schyns, P.G., 2006. Hybrid images. ACM Transactions on Graphics (TOG), 25(3), pp.527-532.

[286] Mark hedley Jones, 2018, Calibration Checkerboard Collection, https://markhedleyjones.com/projects/calibration-checkerboard-collection, access date: 6th of February 2023.

[287] Vagon, 2022, Top 10 CGI Material libraries for Blender, https://vagon.io/blog/top-cgi-material-libraries-for-blender/, access date: 7th of February 2023.

[288] Poly haven, https://polyhaven.com/, access date: 10th of February 2023.

[289] Poliigon, https://www.poliigon.com/, access date: 14th of February 2023.

[290] Mathematics, Viewing a circle from different angles – is the result always an ellipse?, 2019, https://math.stackexchange.com/questions/3103023/viewing-a-circle-from-different-angles-is-the-result-always-an-ellipse, access date: 17th of February 2023.

[291] AmbientCG, https://ambientcg.com/, access date: 20th of February 2023.

[292] Ehrlich, G., 1973. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. Journal of the ACM, 20(3), pp.500-513.

[293] Kenley, C.R. and Coffman, R.B., 1998. The error budget process: an example from environmental remote sensing. Systems Engineering: The Journal of The International Council on Systems Engineering, 1(4), pp.303-313.

[294] Selin, E., 2020, Blender: a Cycles render settings guide. https://artisticrender.com/blender-a-cycles-render-settings-guide/, access date: 22nd of February 2023.

[295] Price, A., 2021, Render engine speed comparison, https://www.youtube.com/watch?v=myg-VbapLno, access date: 24th of February 2023.

[296] MathWorks, Matlab, https://uk.mathworks.com/products/matlab.html, access date: 27th of February 2023.

[297] Whitehouse, D.J., 1997. Surface metrology. Measurement science and technology, 8(9), p.955.

[298] Lukas Stockner, 2019, Introduction to cycles, BCON19, https://conference.blender.org/2019/presentations/566/, access date: 28th of January 2022.

[299] Blain, J.M., 2019. The complete guide to Blender graphics: computer modeling and animation. AK Peters/CRC Press.

Appendix 1

Rendering parameters in Blender 2.9.2

The rendering parameters are found in: “Layout → Scene”, as shown on Figure Appendix1.1.

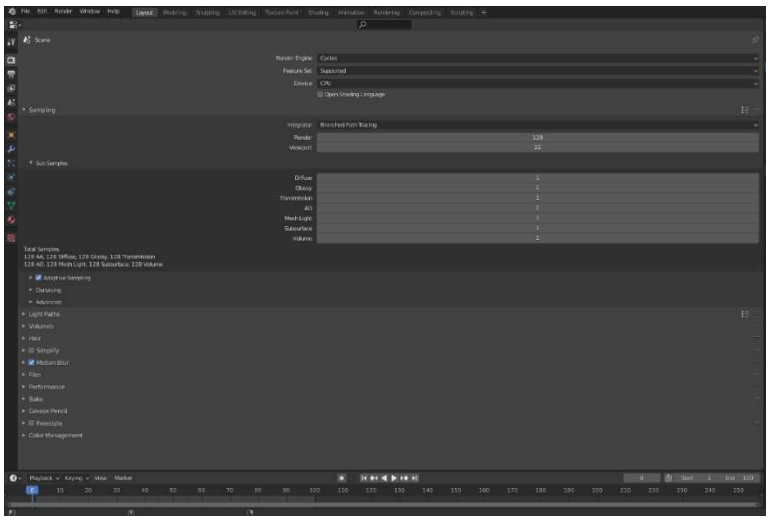


Figure Appendix1.1: Scene panel
(Source: Blender)

The first elements that can be modified concern the render engine. As we saw before, Blender offers three different rendering engines. In this session, the user can choice the one they want to use for the rendering.

- Render engine:
  Choose one of the three default render engines proposed by blender, which are:
  - Cycles: render for complex scene (management of the light).
  - Eevee: animation render.
  - Workbench: the pre-procession render.
- Feature Set:
  This section allows us to choose to have access to the features already tested and controlled (Supported), and experimental features (Experimental), namely, experimental and incomplete features that might be broken or changed in the future.
- Device:
  Allow to choose if the CPU or GPU will be used for the calculation. It is advised to use the CPU to have a better management of the RAM, and the quickest rendering [294-295].

Sampling category

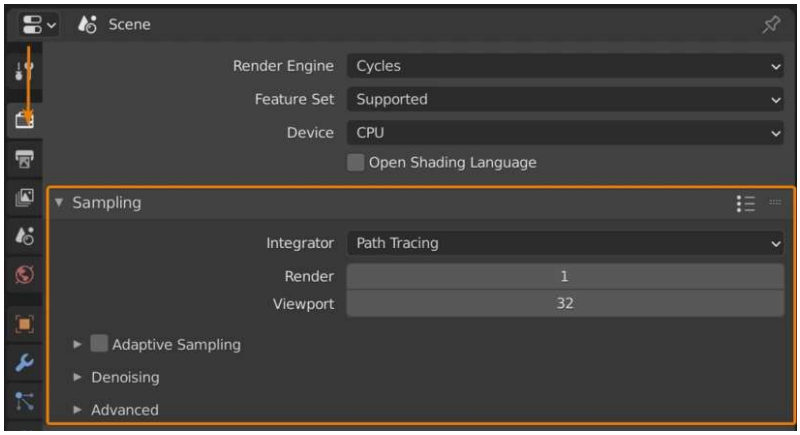


Figure Appendix1.2: Sampling category
(Source: [294])

This category is directly related to the light paths. The light path is the path of the light from the camera to the light source. The light on its way will bounce on the wall and the surface before hitting the light source. Each sample of the light will carry the properties and location of each material met on the way.

In this section and the next one, we have several options to control the samples:

- Integrator:
    - **Path tracing and Branched Path Tracing:**
      The path tracing integration is the more basic one. It gives us equal light bounces no matter what property the surfaces we hit have. It shoots off rays equally making each individual ray faster.
      However, for surface attributes that need more samples to clean up properly and remove noise, it may take longer than the alternative, branched path tracing.
      Branched Path tracing will split the shooting ray when it hits a material. The shooting ray will be divided among a certain number of rays, depending on the surface attributes and light, to determine the properties and location of the material.
    - **Render and Viewport:**
      With the renderer Cycles, the samples are known under the labels **render** and **viewport**. The render count is used for the final renders and the viewport samples are used in rendered viewport shading mode.
      To determine how many samples we need, we need to know how noisy the picture is, because noise comes from a lack of information, that is a lack of samples [294-295].

With the branched path tracing integrator, we can control how many rays Cycles uses for each material or feature hit:

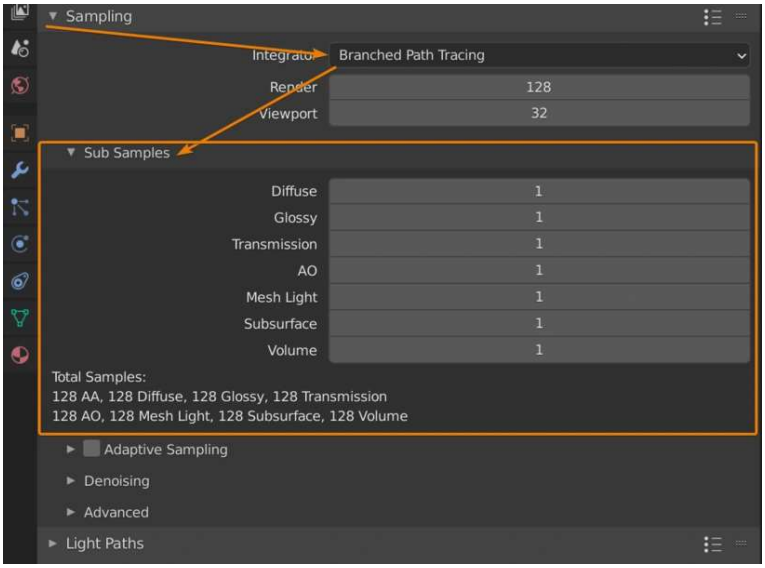


Figure Appendix1.3: Sub samples option
(Source: [294])

In sub samples, we have a list of different features. The number on the right, is the multiplying factor, so the final number of samples is the multiplication of the sample number by this number. For example, we have 100 samples and a factor of 2 for the diffuse feature. So, for the diffuse feature, the final number of samples would be 100 × 2 = 200. But if the other features have a factor of 1, they will stay at 100 samples [294-295].

The next section is **Adaptive Sampling.**

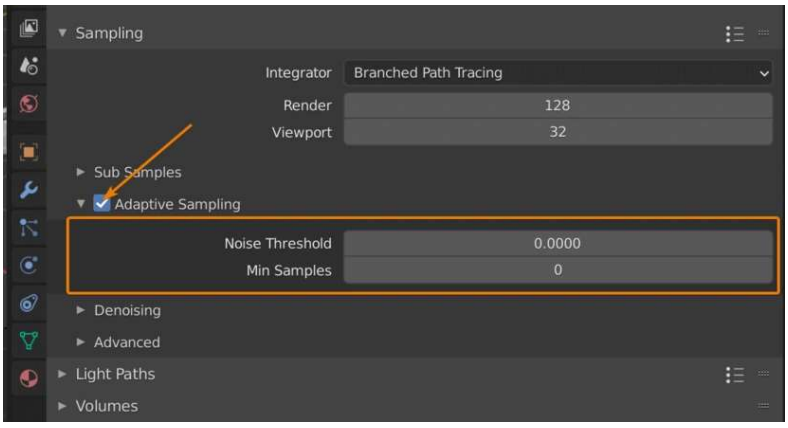


Figure Appendix1.4: Adaptive sampling
(Source: [294])

This section allows us to have control of the noise management. Blender will sense when the noise is reduced enough and therefore stop rendering the area while continuing to render areas that require more samples to become noise free.

With the **noise threshold** enabled, Cycles will keep rendering until it reaches the samples count or until the noise level has reached the threshold.

The **min samples** setting is the minimum number of samples that Cycles must use for the rendering. Cycles is not allowed to use fewer samples than specified here [294-295].

The denoising section sets a denoiser for the final renderer and for the viewport. The denoising section gets rid of noise in the final rendering. The viewport is the window where the final rendering picture is displayed.

Since the viewport is denoising in real-time for every sample, we can set a start sample so that the denoiser is not enabled before a set number of samples has already been calculated.

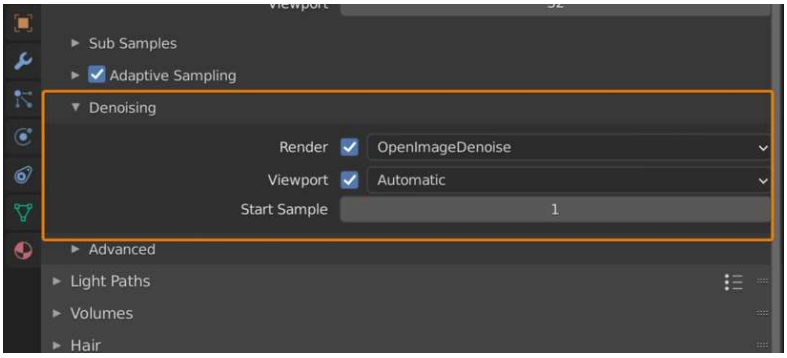


Figure Appendix1.5: Denoising section
(Source: [294])

For the render, three options are enabled:

- **NLM.**
  The traditional built-in denoiser. Not AI based denoisers.
- **Optix.**
  AI based denoisers.
- **OpenImageDenoise.**
  AI based denoisers. It is considered as the denoiser.

For the viewport, three options are also possible:

- **Automatic.**
  Default option but not really good.
- **Optix.**
  Faster option because it has higher priority, but requires a compatible Nvidia GPU.
- **OpenImageDenoise.**
  Default option if we do not have a compatible Nvidia GPU [294-295].

The denoiser also has an impact on what data the denoising data pass produces. **Optix** and **OpenImageDenoise** produce the same passes and look identical:

- Denoising Normal.
- Denoising Albedo.
- Denoising Depth.

NLM produces four additional passes:

- Denoising Shadowing.
- Denoising Variance.
- Denoising Intensity.
- Denoising Clean [294-295].

They are located in: “View Layer → Passes → Data”, as shown in the next Figure:

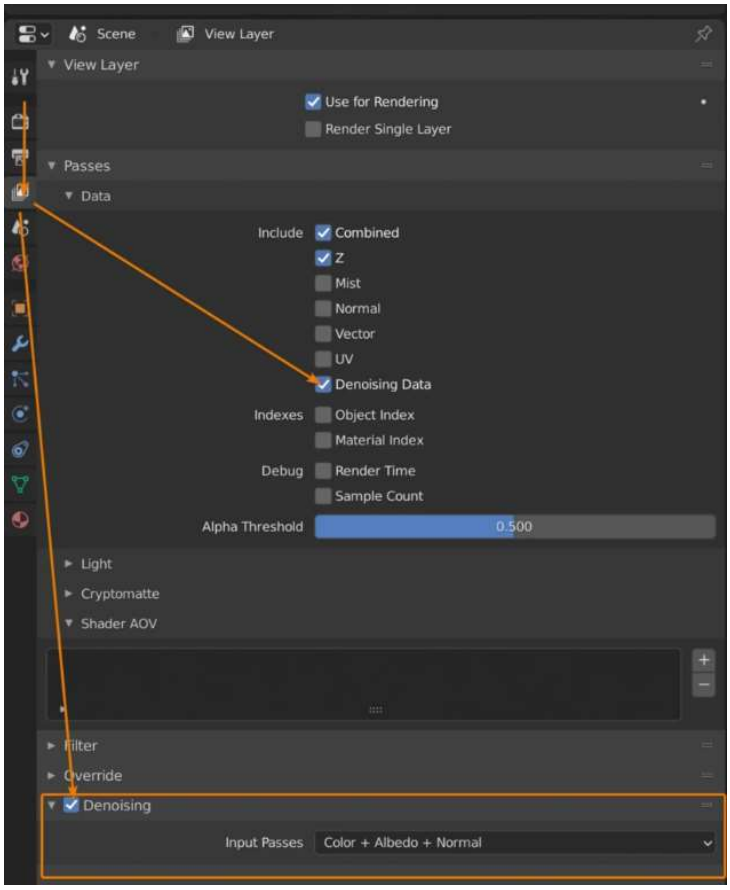


Figure Appendix1.6: Denoising passes
(Source: [294])

The last section in the Sampling category is the **Advanced** section:

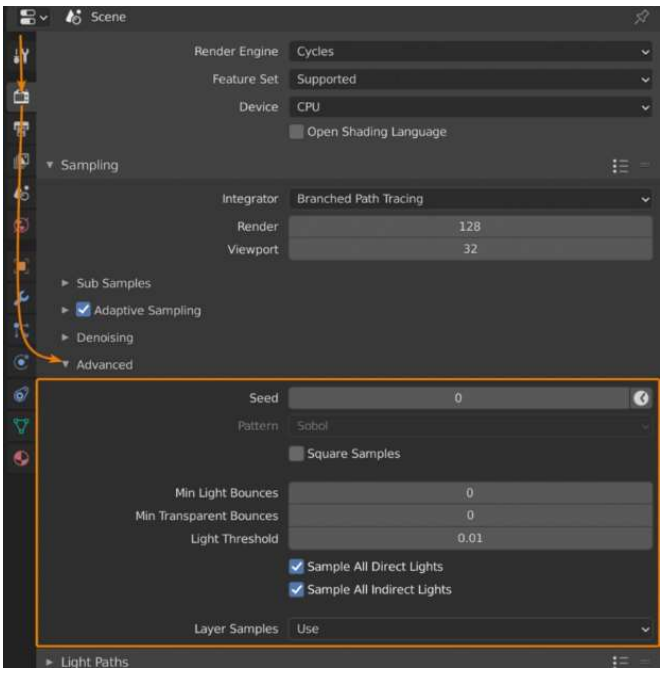


Figure Appendix1.7: Advanced section
(Source: [294])

We have eight options:

- **Seed.**
  Change the pattern of random distribution. In this case, it is the random distribution of the Cycles integrator. This will give us a different noise pattern across the image. The clock icon will change this value between each frame when rendering the animation.
- **Pattern.**
  It is the distribution of samples. Two patterns are available Sobol and multi-jitter. The different is subtle and most of the time un-noticeable.
- **Square samples.**
  Perform the mathematical operation of squaring the number of samples. It is a different way of calculating samples.
- **Min light Bounces.**
  We can override the lowest allowed bounces for all individual ray types. If it is set to zero, then it is disabled. It can help to reduce noise, especially in scenes with glass, liquids, and glossy surfaces. However, the render time will be affected considerably.
- **Min transparent Bounces.**
  Can also help to reduce noise with transparent texture.
- **Light Threshold.**
  Minimum amount of light that will be considered light. If the light is below this threshold, it will not be considered light.
- **Layer Samples.**
  Set the number of samples we want to calculate for this view layer separately [294-295].

Light path category

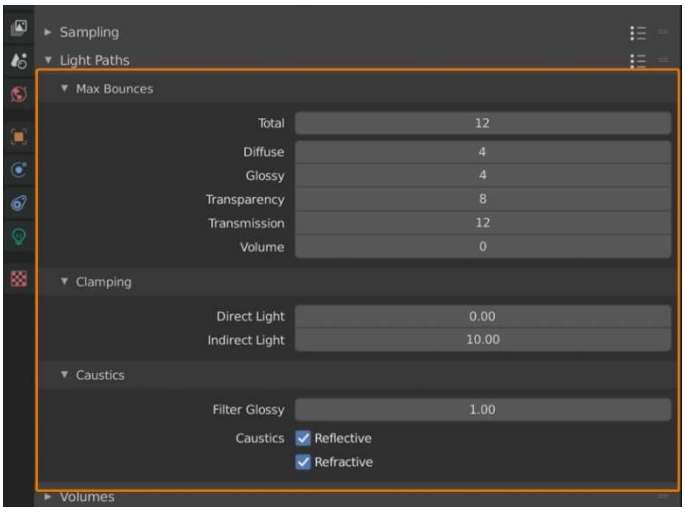


Figure Appendix1.8: Light path category
(Source: [294])

Here we decide the maximum number of bounces we want for each of the rays shot into the scene. The total is the number of bounces for the scene. When this number of bounces is met by any ray, we terminate it.

The subsection allows us to determine how many bounces we want according to the type of material. For example, if the scene is constituted of glossy material, the number of bounces for “glossy” will be higher to produce a better result. However, the render time will be longer [294-295].

In the **clamping** section, two options are possible, Direct and indirect light. These values limit the maximum amount of allowed light recorded by a sample.

This can help to reduce the light calculation error in a render. Light calculation errors are probably caused by a sample going haywire and recording an excessively high number, resulting in white pixels, called fireflies, in the rendering.

However, clamping breaks the accuracy of the light. So, it should be used as a last resort. To disenable it, a value of zero is enough.

The **filter glossy** setting is something useful because it gives glossy components a blur to reduce noise. It can enable us to get away with fewer glossy bounces and speed up renders with minimal cost to accuracy [294-295].

Film category

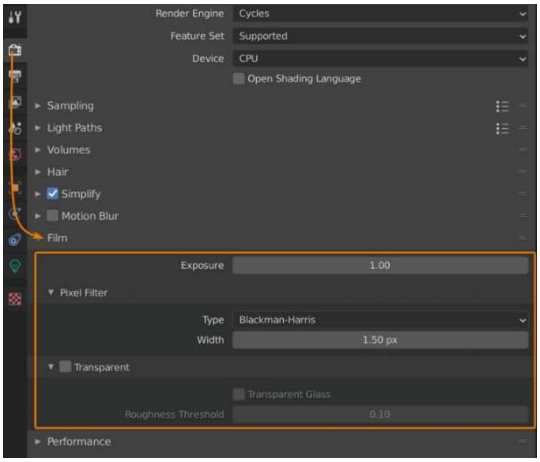


Figure Appendix1.9: Film category
(Source: [294])

This section begins with the **exposure** option. This option decides the brightness of an image. It allows us to either boost or decrease the overall brightness of the scene.

We carry on with the **pixel filter section.** This section is related to antialiasing. Antialiasing is the feature that blurs edges and areas with contrast for a more natural result, hiding the jagged edge between pixels.

The Blackman-Harris filter is based on the Blackman-Harris algorithm, and creates a natural anti-aliasing with a balance between detail and softness.

The Gaussian filter is a softer alternative while box disables the pixel filter.

The pixel filter width manages how wide the effect stretches between contrasting pixels.

The most useful section is **Transparent.** By checking the transparent section, the world background will be rendered transparent, but it will still contribute to the lighting.

When this option is enabled, we have access to **glass transparent.** This allows us to render glass over other surfaces. The roughness threshold will dictate at what roughness level the breakpoint is and we render the original colour instead [294-295].

Performance category

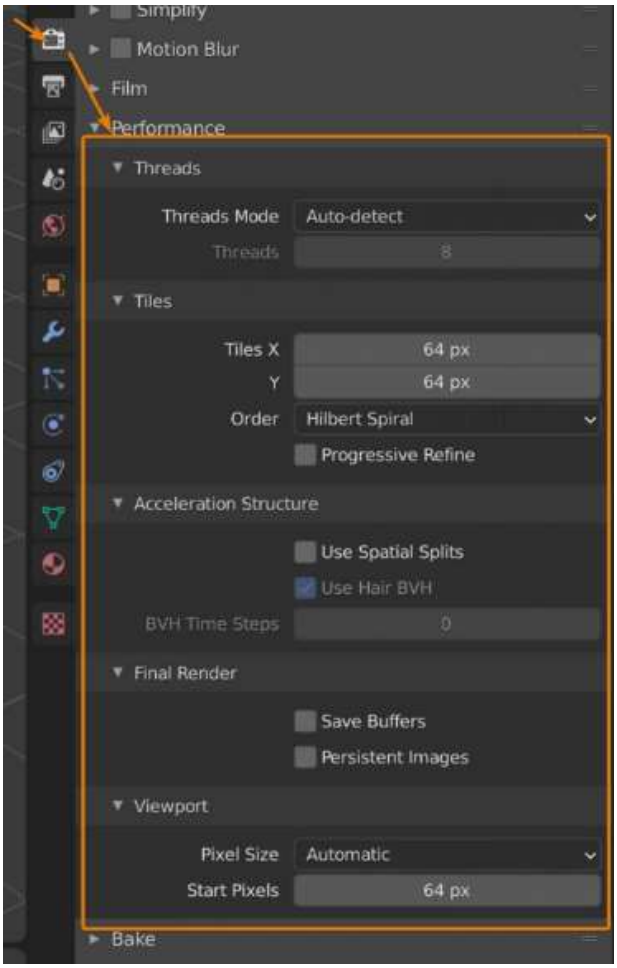


Figure Appendix1.10: Performance category
(Source: [294])

The threads section is only available with CPU rendering. It allows the user to choose how many of the available cores and threads we will use for rendering. By default, Blender will auto-detect and use all cores. However, if we want to keep some computing power to do other things at the same time, then it is useful to set the number of cores allocated at the rendering task [294-295].

The **Tiles** section sets the tile size. The tile X and tile Y values decide how large each chunk should be that is handled at a time by each computational unit. We can also set the **order** that tiles get picked.

Progressive refine helps us to render the whole image at once instead of a tile at a time. A predefined number of samples are not needed, because the samples are counted until the render is manually cancelled.

In the **Final render** section, we have **save buffer** and **persistent images**. **Save buffer** is normally used to save memory during the rendering process if many view layers are used. When it is turned on, Blender will save each rendered frame to the temporary directory instead of just keeping it in RAM. It will then read the image back from the temporary location. However, it seems that this does not really work.

Persistent images will tell Blender to keep loaded textures in RAM after a render has finished. This results in Cycles not having to load those images again, saving sometime during the build process while rendering.

The last section is the **Viewport.** Here two options are available, **pixel size** and **start pixels.**

Pixel size allows us to change the draw size of a pixel. So, the pixel can become bigger or smaller. For example, at 1x we render them at 1×1; and at 2x, we render them in 2×2-pixel sizes. The **start pixels** will set the size of pixels at the start, this will be refined to the pixel size setting over time [294-295].

Bake

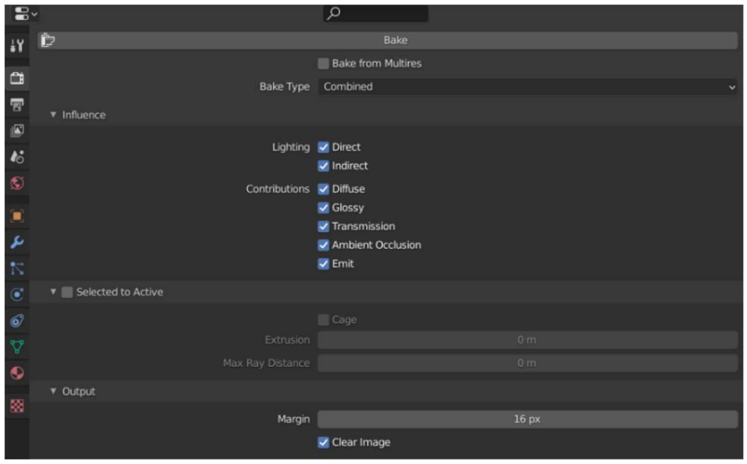


Figure Appendix1.11: Bake category
(Source: [294])

The **bake** category is used to save time during the rendering. It is the process of calculating and saving the Cycle light data directly on the objects themselves, namely, lights can be viewed in real-time.

The option is really helpful for game development, as well as camera fly through animations. It works by UV unwrapping the object. UV unwrapping is the process to map 2D assets like images/textures onto 3D objects. U and V refers to the x and y axes of the 2D texture that is projected onto the 3D model.

creating a blank image and baking the light data onto it; this saves every type of lights, such as the environment light, and the indirect and direct light.

However, if we want to save only one type of light, then we need to use the **influence** option. This option allows us to choose the type of light we want to bake and save [294-295].

Appendix 2

Details of the MATLAB based set of algorithms for the detection of the circle zones artefact

Blender generated images and the MATLAB box was finding the 3D coordinates of the circle zones, in the image, by using the photogrammetry method through a triangulation process.

The toolbox steps are:

- Loading the picture.
- Determining the contrast parameters needed for the detection:
  To find the centre of the circles of the calibration board, the function regionprops is used. The problem with real image is that the contrast between bright and dark colours is sometimes too soft, blurring the difference between them.

  To remove this problem, the function *imadjust* is used. This function adjusts the image intensity values or colourmap.

  The detection is done to let the user decide if the level of contrast in the picture is enough or not for the detection.

  The decision process is done through a dialog box, shown in the next Figure:

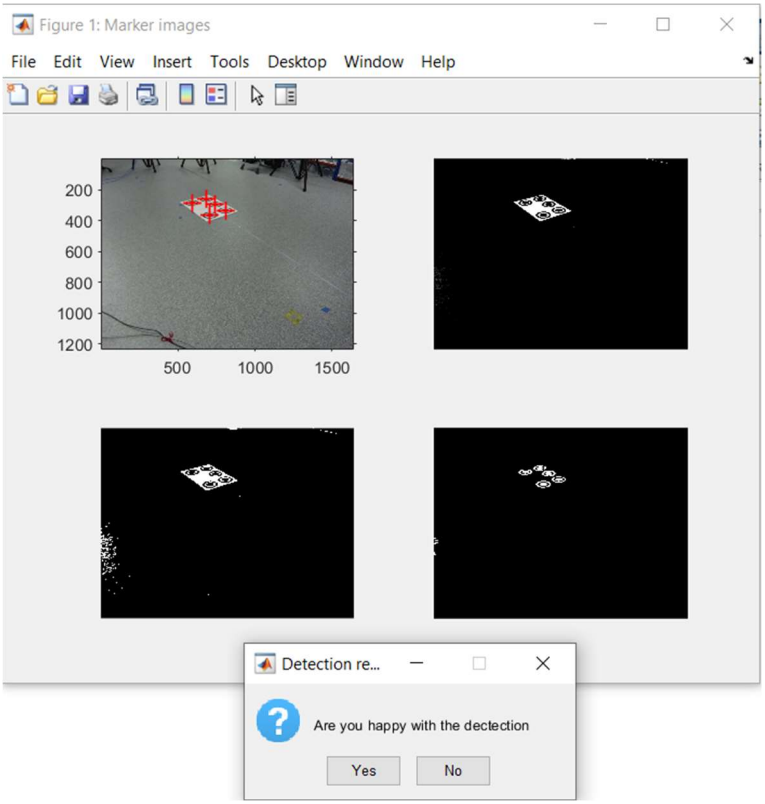


Figure Appendix 2.1: Detection process in Matlab

- Calculating of the camera's parameters (translation vector and rotation matrix):
    - Setting the focal length and size (width and height) of the camera.
    - Setting the size of the pixel on the camera sensor.
    - Setting the location of the principal point.
    - Finding the centre of the target circles:
        - Modifying the contrast level in the image with the parameters found previously.
        - Transforming the image into a binary one.
        - Creating the complement of the binary image.
        - Using regionprops on the binary and its complementary images to find the centre of the circles.
        - Using a threshold to keep only the interesting points.
        - Using kmeans to find the centre of the cluster.
        - Plotting the results with the real, the binary and the complementary images.

    The following picture shows the result of the detection:

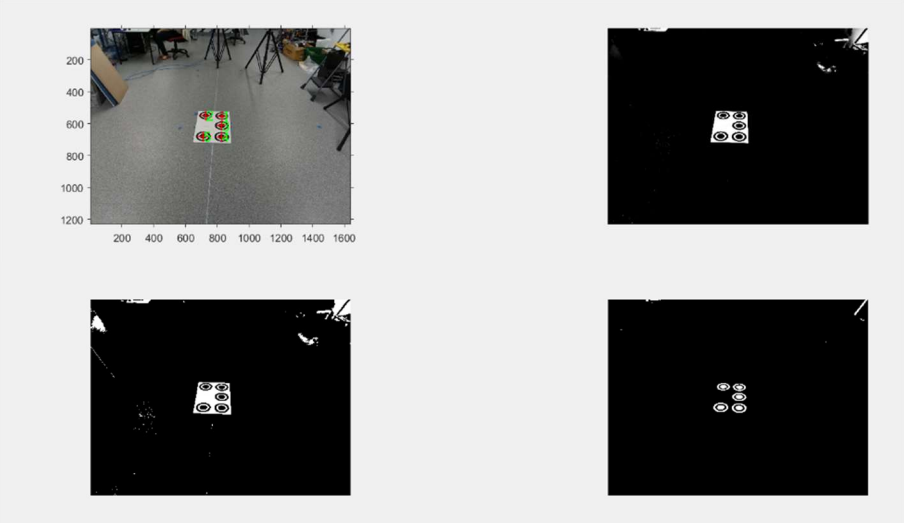


Figure Appendix 2.2 : Visualisation of the detection in Matalab

On the top left, the real image and the centres found are displayed. On the top right, it is the grey image.

On the bottom left, it is the binary image, and on the bottom left, its complementary.

- Checking the value of the centre of the target circles:
  The vector between the circles 4 and 2 is calculated by the cross product of the vectors between the circles 1 and 4, and 1 and 2. If it not null, the centres are correctly detected for these targets.

  If the sign of the vector is positive, the indices of the circle 1 and 4 are correct. Otherwise, 1 is at the place of 4 and vice versa.

  For the target 2 and 5, we check the angles. The first angle is between the vectors between 1 and 4, 1 and 2. The second is between 1 and 5, and 1 and 4. If the indices are correct, the angle between the vector 1-4 and 1-2 is higher than the other one.

  The angles are calculated using the following trigonometry formula:

$$\theta = \operatorname{acos}\left(\frac{(\vec{v}.\vec{u})}{\|\vec{v}\| * \|\vec{u}\|}\right) \qquad \text{Eq.78}$$

  Where θ is the angle between the two vectors, $\vec{v}$ and $\vec{u}$.

- Translation of the MATLAB camera parameters into the Blender coordinates frame.
  - Rotation
    - $R_{Bx} = R_{Mx}$
    - $R_{By} = R_{My}$
    - $R_{Bz} = R_{Mz}$
  - Translation:
    - $T_{Bx} = 180 + T_{Mx}$

- $T_{By} = -T_{My}$
- $T_{Bz} = -T_{Mz}$

Where:

(R_{Bx}, R_{By}, R_{Bz}) are the components of the Blender rotation matrix.

(R_{Mx}, R_{My}, R_{Mz}) are the components of the MATLAB rotation matrix.

(T_{Bx}, T_{By}, T_{Bz}) are the components of the Blender translator vector.

(T_{Mx}, T_{My}, T_{Mz}) are the components of the MATLAB translator vector.

- Calculating the 3D coordinates of each target centre by triangulation.
- Calculating the reprojection error (Equation 62, Chapter 2).
- Calculation the distance between each target centre.
- Conversing the camera's parameters calculated with MATLAB to a Blender format:
- Displaying the results:

  The results of the calibration are under two forms:

  - Reconstruction of the scene:

  In **Error! Reference source not found.** and Figure Appendix 2.3, we have the scene built from the information calculated by the tool box, and the verification of the results with the reproject vectors.

  These vectors are calculated for each point, and for each camera. So, each camera has reprojected vectors.

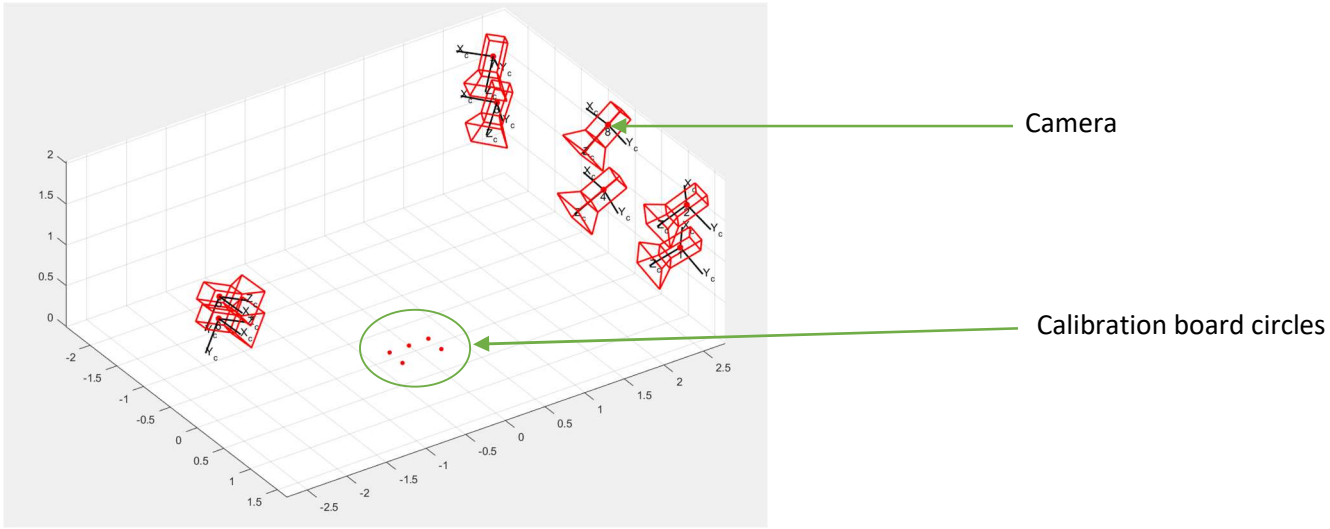


Figure Appendix 2.3: Scene rebuilt

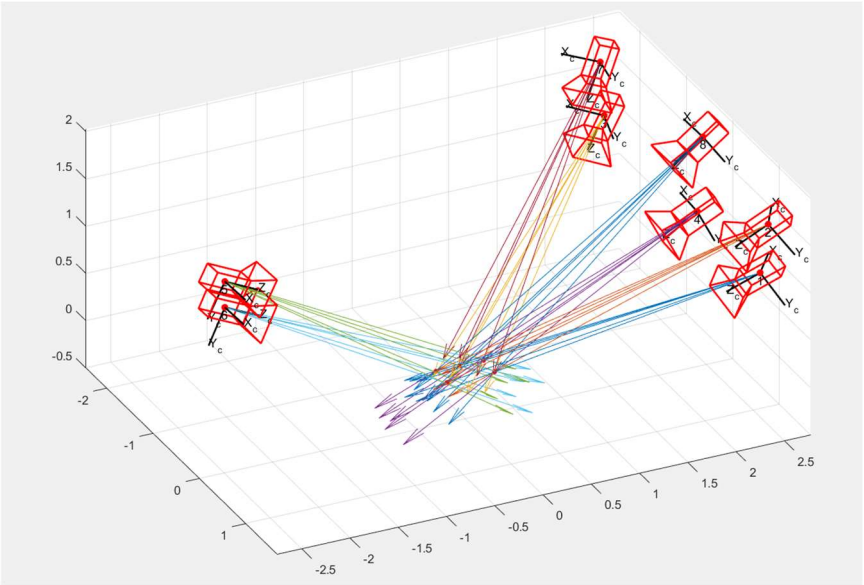


Figure Appendix 2.4 : Scene with the reprojection vectors

- Measurements of the targets' distances with the error on the measurement, the reprojection error for each target detection, and the parameters conversion between MATLAB and Blender.

Figure Appendix 2.5 and Figure Appendix 2.6 shown the results of the calibration:

```
The reprojection error for marker 1 is 2.065 mm
The reprojection error for marker 2 is 2.706 mm
The reprojection error for marker 3 is 2.494 mm
The reprojection error for marker 4 is 2.443 mm
The reprojection error for marker 5 is 2.610 mm

  1 -   2 : 0.249m +- 3.404 mm
  1 -   3 : 0.247m +- 3.238 mm
  3 -   4 : 0.247m +- 3.491 mm
  4 -   5 : 0.250m +- 3.575 mm
  2 -   5 : 0.495m +- 3.760 mm
  1 -   4 : 0.494m +- 3.199 mm
```

Figure Appendix 2.5 : Reprojection error for each circle and distance between each circle measured by MATLAB

```
Rotation
(-5.345242e+00, -1.817697e-02, 2.493467e+00),
(-5.391830e+00, 7.706315e-03, 2.612841e+00),
(-5.242892e+00, -1.151722e-02, -2.485091e+00),
(-5.189277e+00, -2.280992e-02, 3.108399e+00),
(-5.181027e+00, 2.406286e-02, 5.644219e-02),
(-5.157011e+00, -1.585614e-02, 1.766008e-02),
(-5.409751e+00, -9.139373e-03, -2.424744e+00),
(-5.338987e+00, -7.805272e-03, -3.117209e+00),

Translation
(1.470712e+00, 2.351984e+00, 1.207798e+00),
(1.469745e+00, 2.435938e+00, 1.701069e+00),
(-2.104738e+00, 2.448407e+00, 1.140214e+00),
(-1.404196e-01, 2.471265e+00, 1.058822e+00),
(8.062682e-02, -2.530319e+00, 1.569441e+00),
(3.971218e-02, -2.511232e+00, 1.272148e+00),
(-2.145217e+00, 2.427568e+00, 1.692404e+00),
(-1.545524e-01, 2.537342e+00, 1.825594e+00),
```

Figure Appendix 2.6 : Camera's location and rotation calculated to be used in Blender

Appendix 3

Determination of the optimum number of cameras

Experiment presentation

This work was done to determine the optimum number of cameras to perform object tracking. Due to equipment limitation and the COVID crise, this experimentation was performed in Blender. The system used was composed of 2D markerless camera located at 5 m from the chess board, on an aluminium gantry, and a chess board with a width of 445.0 mm, a length of 712.0 mm, and square size of $89\ mm$. These values were measured on the real board with a calliper, and a VICON system, and with a ruler in Blender.

96 pictures were used per camera, generated by the movement and rotation of the chess board by a Python script in Blender. The photogrammetry principal was used to perform the 3D reconstruction of the scene, based on the triangulation method, in MATLAB.

To determine the optimum number, 3, 4, 5, 6, 7, and 8 cameras were used. 3 was determined as the minimum due to the triangulation theory [296]; and 8 was the maximum because the whole set was composed of 8 cameras.

The criterion used in this work was the measurement of the length and width of the chess board reconstructed in the 3D world, in MATLAB.

System presentation and scene creation

The camera calibration system used in this experiment was different than the one used in the thesis. This system was composed of eight 2D markerless cameras with a resolution in the order of mm, and a focal length of $2/3$"; located on a 4 m x 3 m x 5 m aluminium gantry.

The 2D markerless system was considering at the beginning of this project as a first approach to determine the optimum location of cameras and markers. However, with the evolution of the work, and the willingness to have a mobile system, the Raspberry Pi V2 was chosen.

The Blender system is presented in Figure Appendix 3.1, and the image generated as output in Figure Appendix 3.2.

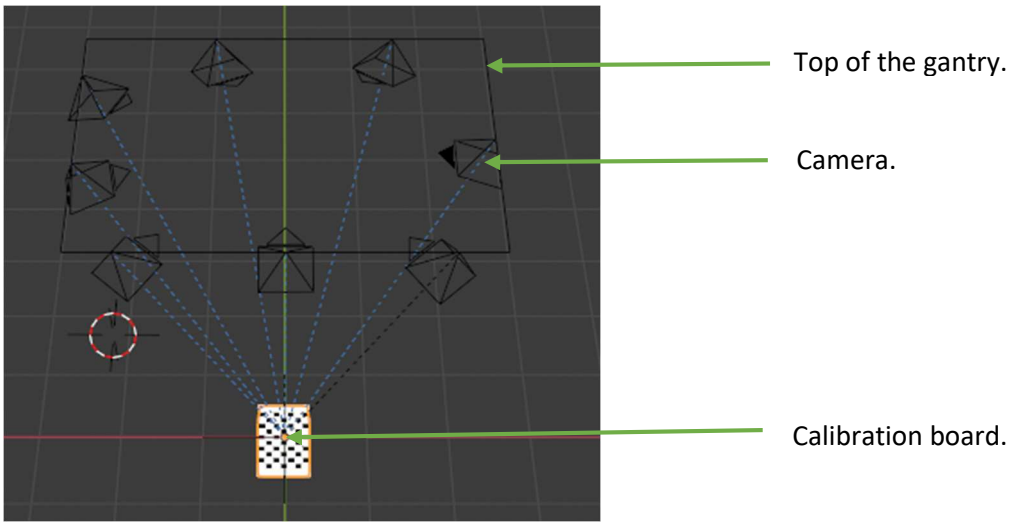


Figure Appendix 3.1: Blender scene

In Figure Appendix 3.1, the rectangle represents the top of the gantry, used as a reference to position the cameras at the correct height.

The calibration board used for the calibration is the board presented in Chapter 4, with a chess board pattern. This pattern was chosen due to the utilisation of the function *detectCheckerboard()* in the MATLAB script for the image treatment.

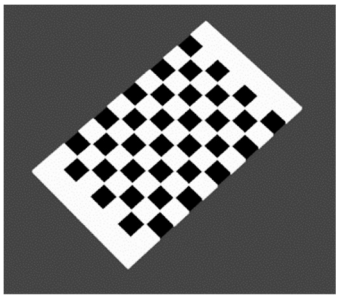


Figure Appendix 3.2: Picture generated by Blender

To generate Figure Appendix 3.2, the following rendering parameters were modified. They are accessible in the render properties menu of blender:

- Render: Cycles (Pinhole model render).
- Sampling:
    - Integrator: Branched, Path, Tracing
    - Adaptive sampling: OK
    - Render: 128 samples by pixels.
    - Viewport: 32 samples.
    - Denoising:
        - Render: OpenImageDenoise.
        - ViewPort: Auto.
        - Start sample: 0.

To be certain that the cameras were “looking at” the centre of the calibration board, an object constraint was applied through the constraint menu, accessible in the property menu.

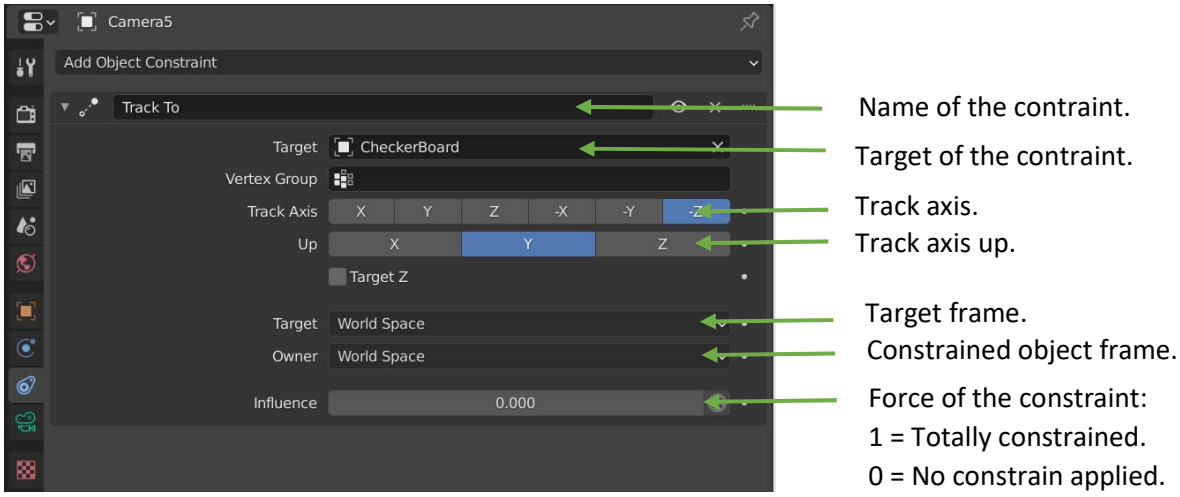


Figure Appendix 3.3: Constraint panel for camera 5 (Blender)

The constraint used was *Track to*, moving the camera according to the movement of the centre of the "target", here the calibration board. In this project, this constraint is used to "force" the cameras to look at the calibration board and to place them relative to the location of the object. However, this constraint has to be removed after the scene creation to avoid the cameras following the board when this one is moved to generate the set of pictures used for 3D reconstruction.

Pictures generation

The set of pictures was composed of 96 pictures per camera generated by the movement of the calibration board at 12 different locations in the scene delimitated by the camera, and rotated at each new position, 8 times clockwise. The calibration board was moved at 12 positions, and for each them 8 rotations of the board were performed.

The 12 positions were determined to have a position in the middle of the scene, where all the camera can perceive perfectly the board, and 11 around this central point in a circle at a distance of 1.5 m and 2 m; corresponding to the half of the length and width of the gantry.

The 8 rotations of the board were $0°, 90°, 135°$and $180°$ with the board horizontal. The board was also tilted to the side with an angle of $30°$, $-30°$, $40°$ and $-40°$.This tilt was done when the board was parallel to the width of the gantry, e.g., when its rotation was at $0°$.

Methodology

The general experiment was, in Blender, to locate the cameras on the gantry, and force them with the *Tracked to* constraint to "look at" the board in the centre of the gantry with no rotation (referred previously as a rotation of $0°$). The constraint was removed, and with a Python script, each camera, generated in the scene, was taking a picture of the board, before its rotation and translation through another Python script.

When the set of pictures was generated, in MATLAB, the image treatment was done with a similar MATLAB based set of algorithms used in this thesis. The only different was the function used to detect

the pattern (the MATLAB function *detectCheckerboard()*), and utilisation of the Bundle Adjustment (BA) to refine the results.

The Bundle Adjustment is an algorithm used to refine the 3D coordinates of the images 2D points reprojected in the image plane, and to recalculate the camera parameters of the cameras used to generate these images. This algorithm is based on the Levenberg-Marquardt algorithm explained in Chapter 3, section 3.4.1. Its general equation is given by Equation 53 in Chapter 2.

The first output of this experiment was a 3D reconstruction of the scene, as illustrated in Figure Appendix 3.4

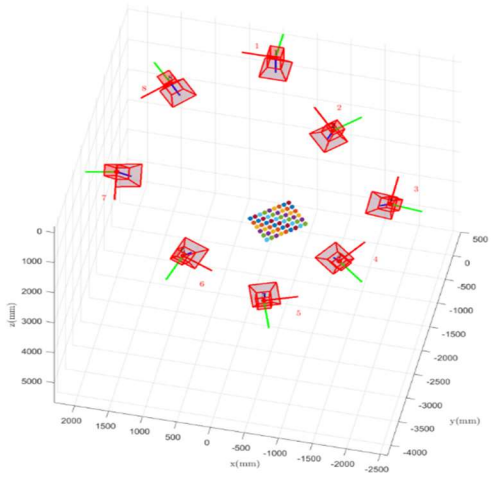


Figure Appendix 3.4: 3D reconstruction

To validate the position of the chess board in the scene, the projection vectors for each camera were plotted. For better readability, only the projection vectors for the four points represented the four corners of the chess board are presented in the following Figures.

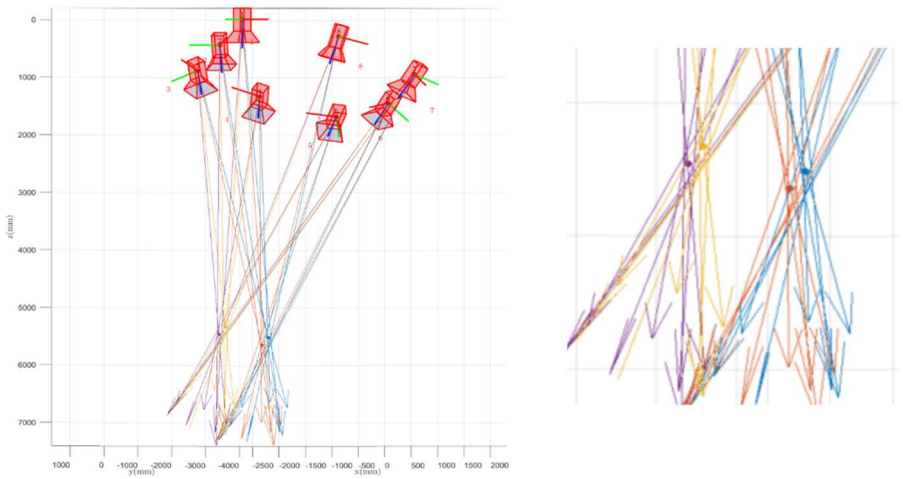


Figure Appendix 3.5: Projection vectors and a zoom on the points plotted

In Figure Appendix 3.4 and Figure Appendix 3.5, it is noticeable that the lines are not crossing on the dots. These differences might come from the detection in MATLAB.

The second output was the calculation of the dimensions of the chess board:

```
 1 -   6 : 444.4 mm +- 2.5 mm
 1 -  49 : 710.5 mm +- 3.1 mm
 6 -  54 : 710.4 mm +- 2.8 mm
49 -  54 : 444.6 mm +- 3.4 mm
```

Figure Appendix 3.6: Dimensions of the calibration board measured by Matlab

The result was provided by the following logic. The chess board was composed of 54 points, with 9 rows and 6 columns (counted in the real board, in the virtual board, and in MATLAB by checking the dimensions of the matrix stocking the image points detected).

In addition, 5 squares are composing the x axis, and 8 squares are composing the y axis (counted as well in the same way than the number of points constituted the board).

Then, the two top corners are defined by the point 1 and 6 due to the 5 squares between them. The two low corners are defined by the points 49 and 54 due to the number of rows and the 5 squares.

The width of the board is then measured between points 1 and 6; and points 49 and 54; while the length is measured between points 1 and 49; and points 6 and 54. In each case, both distances were measured to determine the impact of the lens distortion and the noise on the measurement.

Finally, another measurement can be used to check the results in Figure Appendix 3.6, the square size of the board. Knowing that the board length is composed of the 5 squares, and the board width of 9 squares, the measurement between points 1-6, and points 49-54 need to be divided by 5; and the measurement between points 1-49, and points 6-54 by 9. If the results are correct, the square size found will be 89 mm.

In MATLAB, the length and the width were calculated by taking the norm of the difference between the two points defining the length or the width. For example, the distance between points 1 and 6 is equal to the norm of the difference of these two points. The uncertainty was calculated by using the standard deviation defined by Equation 62 in Chapter 2.

This whole process was used for a set of 3, 4, 5, 6, 7 and 8 cameras.

Results

The results are given in Table Appendix 2.1:

Table Appendix 2.1: Results for the triangulation with a perfect system

Number of cameras used	Measures (in mm) no BA		Measures (in mm) BA	
3	1 - 6	443.6 +/- 30.8	1 - 6	443.7 +/- 7.5
	1 - 49	709.8 +/- 30.2	1 - 49	709.8 +/- 7.5
	6 - 54	709.8 +/- 29.1	6 - 54	709.9 +/- 6.0
	49 - 54	443.5 +/- 28.4	49 - 54	443.6 +/- 6.0
4	1 - 6	444.5 +/- 22.3	1 - 6	444.4 +/- 6.9
	1 - 49	710.6 +/- 23.3	1 - 49	710.3 +/- 9.2
	6 - 54	710.7 +/- 22.2	6 - 54	710.3 +/- 8.0
	49 - 54	444.5 +/- 23.2	49 - 54	444.5 +/- 10.1
5	1 - 6	444.0 +/- 15.0	1 - 6	444.0 +/- 3.5
	1 - 49	709.5 +/- 15.7	1 - 49	709.3 +/- 4.6
	6 - 54	710.0 +/- 14.6	6 - 54	709.9 +/- 3.7
	49 - 54	444.0 +/- 15.2	49 - 54	444.1 +/- 4.7
6	1 - 6	444.2 +/- 13.6	1 - 6	444.4 +/- 3.4
	1 - 49	709.9 +/- 13.7	1 - 49	710.2 +/- 4.2
	6 - 54	709.7 +/- 12.9	6 - 54	710.0 +/- 3.9
	49 - 54	444.5 +/- 12.9	49 - 54	444.7 +/- 4.6
7	1 - 6	444.0 +/- 10.6	1 - 6	444.2 +/- 3.1
	1 - 49	709.8 +/- 10.5	1 - 49	710.0 +/- 3.9
	6 - 54	709.6 +/- 10.0	6 - 54	709.8 +/- 3.6
	49 - 54	444.2 +/- 10.0	49 - 54	444.4 +/- 4.3
8	1 - 6	444.8 +/- 1.8	1 - 6	444.4 +/- 2.5
	1 - 49	711.3 +/- 1.7	1 - 49	710.4 +/- 2.8
	6 - 54	711.3 +/- 1.7	6 - 54	710.4 +/- 2.8
	49 - 54	444.9 +/- 1.6	49 - 54	444.6 +/- 3.4

According to Table Appendix 2.1, the results are getting better and better with the increase of the cameras number. According to the reference values (445.0 mm and 712.0 mm), the best results are given for a set 8 cameras. In addition, it is noticeable that the results are better after the used of the Bundle Adjustment except for the set of 8 cameras.

The Bundle Adjustment is used to refine the results by recalculating the camera parameters and the 3D reprojection of the 2D points in the 3D plane. Regarding Table Appendix 2.1, when the number of cameras is low, for instance, 3 cameras, the refinement of the results is more important than for a large number of cameras, such as 7 cameras. This shows that the less cameras used, less accurate is the measurement due to a lack of information.

Finally, the Bundle Adjustment is based on a minimisation problem solved by the Levenberg Marquardt algorithm. It seems that for a set of 8 cameras, the algorithm cannot converge, saying that the results are alright optimised.

Appendix 4

Surface texture measurements

All artefacts and calibration boards have been measured for surface texture. This was completed using the following equipment and conditions:

- Taylor Hobson CLI2000 instrument.
- UKAS calibration August 2023.
- CLA300 non-contact gauge.
- Traverse lengths between 5 mm (flat objects) and 9 mm (curved objects) – depending on object.
- 50 micrometres / second traverse speed.
- 0.5 micrometres data acquisition resolution.
- Maximum vertical measurement range 300 micrometres.
- Spheres randomly rotated between measurements.
- Calibration Boards – randomly selected white and black squares.
- Multiple measurements per artefact.
- Data processed in DigitalSurf MountainsMap v10.
- Form Removal (Spheres) using a standard cylinder removal process.
- Levelling using Least Squares operator.
- Waviness Removal (Gaussian Filter, Lambda S = 2.5 micrometres, Lambda C = 0.25 mm).
- R parameter outputs.

Surface texture value calculation

As explained in Section 3.2.2.2, surface texture is defined as the geometric irregularities present on a surface, and is defined by two characteristics, surface roughness and waviness (the measurement of the larger wavelength component (at the low frequencies) of the surface texture). In surface metrology, roughness is generally considered to be the high frequency, short wavelength component of a measured surface [297], describing a roughness profile. A common surface roughness parameter is therefore the average of the roughness profile, denoted R_a, and its calculation is explained using Figure Appendix 4.1.

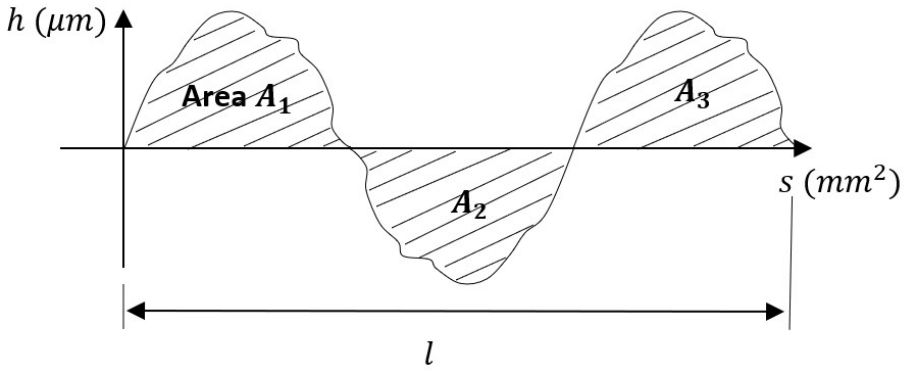


Figure Appendix 4.1: Diagram of the areas of the roughness profile of a random surface

In Figure Appendix 4.1, the x-axis corresponds to the length of the surface of the measured object, the y-axis to the surface roughness, and l corresponds to the sampling length taken for the measurement. The calculation of the surface texture of an object corresponds to the average value of one of the roughness pattern areas R_a, defined in Equation 79. Note that in this calculation, the absolute values of the area of each pattern along l.

$$\boldsymbol{R_a = \frac{\sum A}{l} = \frac{A_1 + A_2 + \cdots + A_n}{l}} \qquad \textbf{Equation 79}$$

In these experiments, both devices use Equation 79 to calculate the surface texture of the artefacts. The real process of measurement is illustrated in Figure Appendix 4.2, and has been performed three times on each artefact to check the repeatability of the measurement.

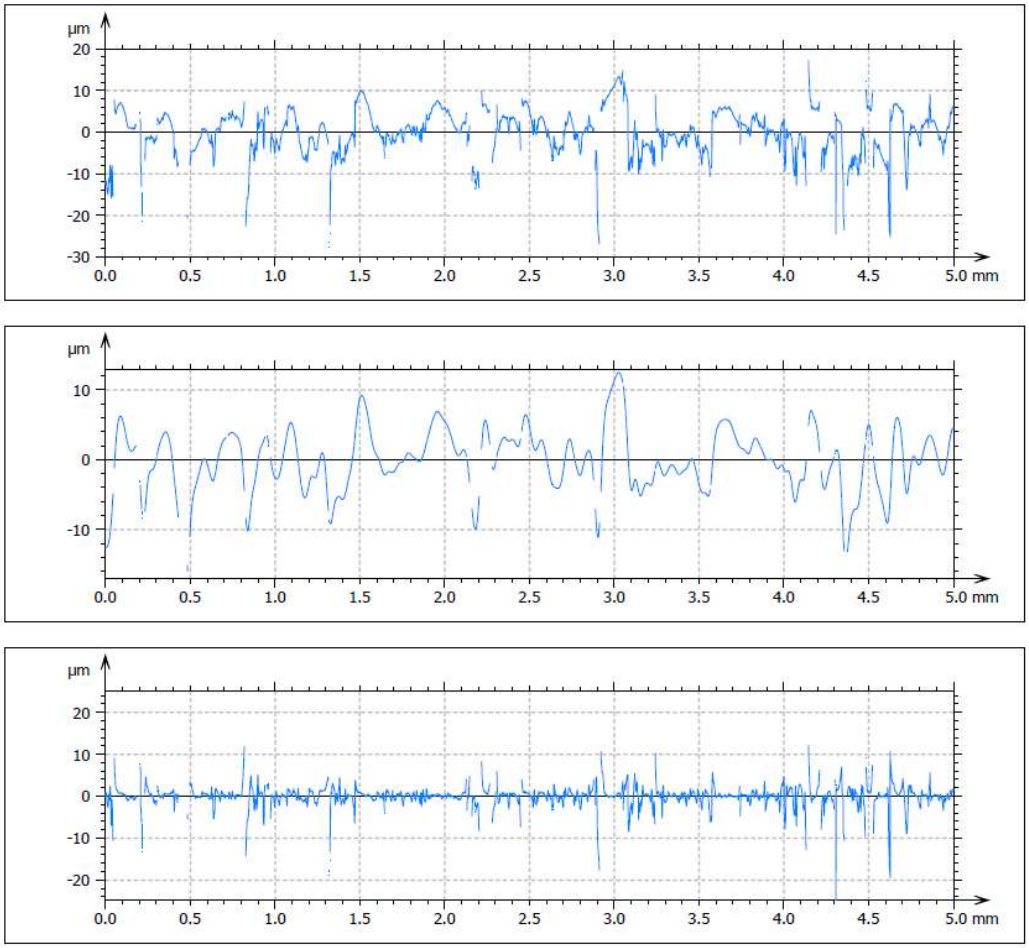


Figure Appendix 4.2: Example of a Ra measurement (performed on the large checkerboard).

In the real world, data are usually noisy, as illustrated in the top graph of Figure Appendix 4.2. Least squares algorithms are used to level the data. Finally, to obtain the roughness profile of the object, a Gaussian filter is applied as illustrated in the last graph of Figure Appendix 4.2. The frequency limits of this filter or cut offs (λ_c), has been defined according to the recommendations of the ISO 4288 standard in relation to the roughness profile traced by the equipment used.

Results

- Bar of the ball bar fibre carbon part (body)

Test	Surface Finish (μm)	Filter and cut offs
1	0.0614	Gaussian Filter, λ_c = 0.25 mm cut off
2	0.0552	
3	0.0517	
4	0.0581	
5	0.0699	
Average	0.0593	
Standard deviation	0.0062	

- Bar of the ball bar plastic part (extremities)

Test	Surface Finish (μm)	Filter and cut offs
1	1.1197	Gaussian Filter, λ_c = 0.8 mm cut off
2	1.3009	
3	1.3754	
4	1.2227	
5	1.1355	
Average	1.2308	
Standard deviation	0.0973	

- Ceramic spheres

Test	Surface Finish (μm)	Filter and cut offs
1	0.3408	Gaussian Filter, λ_c = 0.25 mm cut off
2	0.4246	
3	0.2961	
4	0.2765	
5	0.2898	
Average	0.3256	
Standard deviation	0.0540	

- Large zone circle board

	Surface Finish (μm)		Filter and cut offs
Test	Black circle	White circle	Gaussian Filter, λ_c = 0.8 mm cut off
1	2.6140	1.4120	
2	2.8072	1.5709	
3	2.7215	1.4219	
Average	2.7142	1.4683	
Standard deviation	0.0790	0.0727	

- Small zone circle board

	Surface Finish (μm)		Filter and cut offs
Test	Black circle	White circle	Gaussian Filter, λ_c = 0.8 mm cut off
1	2.5727	2.3722	
2	3.3205	3.3925	
3	2.8122	2.5757	
Average	2.9018	2.7801	
Standard deviation	0.3118	0.4409	

- Large checkerboard

Test	Surface Finish (μm)		Filter and cut offs
	Black square	White square	
1	3.1134	0.2836	Gaussian Filter, λ_c = 0.8 mm cut off
2	3.4590	0.2531	
3	3.3119	0.2396	
Average	3.2948	0.2588	
Standard deviation	0.1416	0.0184	

- Small checkerboard

Test	Surface Finish (μm)		Filter and cut offs
	Black square	White square	
1	3.5742	0.3205	Gaussian Filter, λ_c = 0.8 mm cut off
2	3.2375	0.2924	
3	3.6221	0.2754	
Average	3.4779	0.2961	
Standard deviation	0.1711	0.0186	

- 0.09 m sphere

Test	Surface Finish (μm)	Filter and cut offs
1	12.173	Gaussian Filter, λ_c = 2.5 mm cut off
2	10.694	
3	9.1681	
Average	10.678	
Standard deviation	1.2267	

- 0.1 m sphere

Test	Surface Finish (μm)	Filter and cut offs
1	8.6883	Gaussian Filter, λ_c = 2.5 mm cut off
2	9.8950	
3	9.3847	
Average	9.3227	
Standard deviation	0.4946	

- 0.2 m sphere

Test	Surface Finish (μm)	Filter and cut offs
1	12.658	Gaussian Filter, λ_c = 2.5 mm cut off
2	10.474	
3	4.2890	
Average	9.1401	
Standard deviation	3.5442	

- 0.07 m circle

Test	Surface Finish (μm)		Filter and cut offs
	Black background	White circle	
1	3.0864	4.2693	Gaussian Filter, λ_c = 0.8 mm cut off
2	2.9898	4.2712	
3	2.5329	3.052	
Average	2.8697	3.8642	
Standard deviation	0.2414	0.5743	

- 0.1 m cube

Test	Surface Finish (μm)	Filter and cut offs
1	0.1074	Gaussian Filter, λ_c = 0.8 mm cut off
2	0.1140	
3	0.1288	
Average	0.1167	
Standard deviation	0.0090	

- 0.2 m cube

Test	Surface Finish (μm)	Filter and cut offs
1	0.3466	Gaussian Filter, λ_c = 0.8 mm cut off
2	0.2928	
3	0.2035	
Average	0.2810	
Standard deviation	0.0590	

Appendix 5

Simulating the light path in Blender

The rendering process meets a given need: “Compute the image that a camera would see if it was placed in a given scene”. Its inputs are the components of the full scene, i.e., the geometry, the lights, the materials, and other, and its output, the image or the animation generated.

However, the need of computing an image similar to a photograph means simulating the real world with its complexity. One of the key characteristics is the light, which can be modelled through different models such as [183]:

- Diffraction equations.
- Dispersion equations.
- Relativity model.
- Quantum Mechanics.
- Non-Linear Optics.
- Polarization model.
- Wave-particle duality.

Unfortunately, it is impossible to simulate all of these. Nonetheless, it is not necessarily a requirement to have a perfect result. Consequently, the light modelling can be completed in a straightforward manner using geometric optics [183]:

- Light is emitted somewhere.
- Light travels along a straight line.
- Light bounces off objects.

Moreover, the light models cited above can still be modelled, by importing the equations/formulae which describe them. For example, an effect of dispersion can be simulated by using the geometric optics, if the equation which defining it is known [183].

Simulating the light path

Geometric optics is based on the path of light. The path of light is defined as light travelling along straight lines. Its simulation is a difficult problem because there are an infinite number of paths, which makes the problem intractable.

A good strategy could be to select a large number of random light paths and average them. An advantage of this strategy is the low noise level of the result, because the more light paths simulated, the less noisy the result will be. This method is called unbiased because, mathematically, the software will eventually average out to the correct result [298-299].

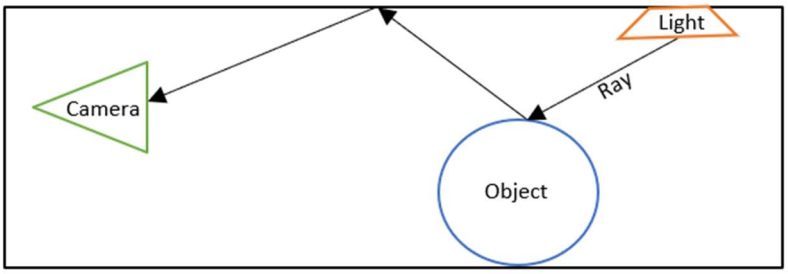


Figure Appendix 5.1: Example of a random light path from a light source

Figure Appendix 5.1 illustrates the unbiased method in a simple way. The computer simulates a light path from a light source whose position and direction are random. It follows the path of the light until an object is hit. If the object is a virtual camera, the light is recorded by the camera sensor, and the simulation stops; otherwise, a new light path is calculated from another random light source, and the same task is repeated.

Unfortunately, in practice this does not really work due to the small size of the virtual camera, leaving the light beam to bounce around until it hits a camera. In addition, the light loses energy with each bounce, so that only a small amount of light reaches the camera [298-299]. Noted that this affirmation is true in Blender according to [299].

However, in geometrical optics, one of the results is the reversibility of the light path. Thus, instead of starting from the light source, another solution could be to start from the camera, and perform the task as shown in the Figure Appendix 5.2.

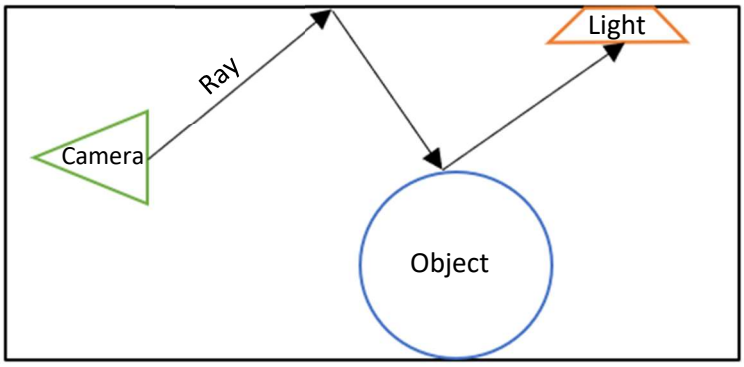


Figure Appendix 5.2: Example of a random light path from the camera

Unfortunately, this method does not work as well, due to the small size of the light sources. Thus, the camera and the light source are defined as tiny objects. To be sure that the light beam will be connected to both, the light path has to start from the camera and its direction has to be defined, instead of being random [298-299].

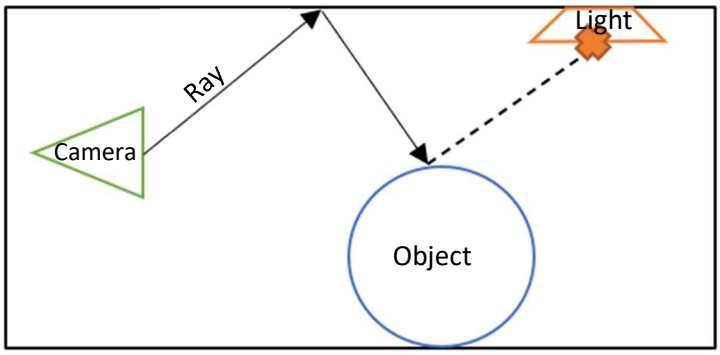


Figure Appendix 5.3: Example of a forcing light path

However, there is still a problem with the connection between the camera and the light. As shown in Figure Appendix 5.3, the light path hits an object that is not the light source and bounces back. However, as the connection between the light source and the camera has not been checked, the same problem as in Figure Appendix 5.1 may occur. An alternative could be to choose a random point and light source, but this would lead to an entirely different configuration without the certainty of hitting something. Therefore, the connection between the camera and the light source must be made to improve the result.

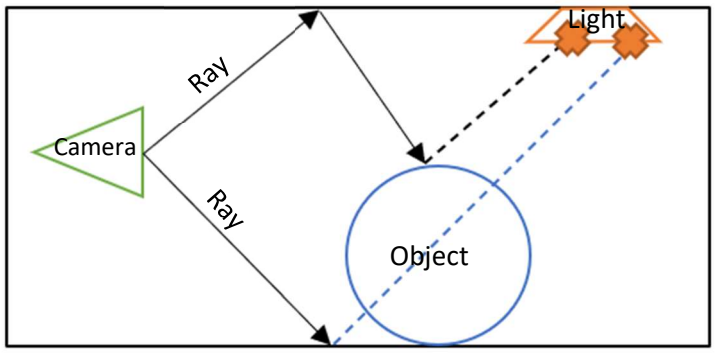


Figure Appendix 5.4: Example of connection

In Figure Appendix 5.4 the black dotted line shows a possible ray path because, when the light path is formed, the light source is connected to the camera. In contrast, the blue dotted line represents an impossible light path because of an object obstructing the path, making it impossible to connect.

As a reminder, the initial problem was to simulate a light path to generate an image. The simulation of a light path is a success, but insufficient to generate an image. However, the generation of multiple light paths, from a camera to a light source, consumes a lot of computing power. A solution to reduce the computing cost, is to use the bounces in the light path, as potential paths, by connecting them to the light source, as illustrated in Figure Appendix 5.5 [298-299].

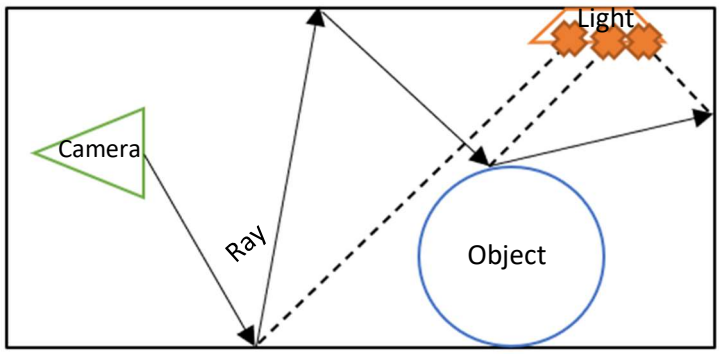


Figure Appendix 5.5: Generation of multi light paths

Figure Appendix 5.5 explains how to generate multiple light paths. However, this explanation does not take into account whether or not a light beam has touched an object, and the information about this interaction creating the image. Nevertheless, optical geometry is composed of triangles, making it possible to test whether a ray hits a triangle and to check the intersection. However, testing all of the potential paths is impracticable due to the large number generated.

Nonetheless, if it is assumed that a ray does not hit an object, then it will hit nothing. So, instead of testing the intersection, it is possible to check whether the light hit or did not hit an object [298]. The basic idea, behind this assumption, is to draw a box around an object, called a bounding box, and to check whether the ray hit the box or not. If the ray misses the bounding box, then the object inside is missed as well. alternatively, if the ray hits the bounding box, then it might hit the object as well, and this possibility is checked. This test is run for each light ray, illustrated in Figure Appendix 5.6.

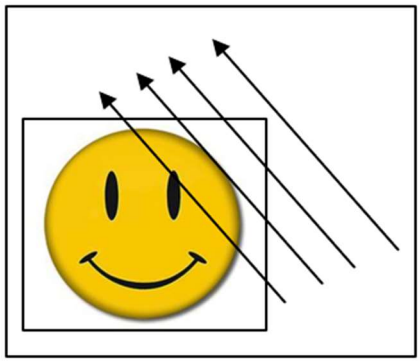


Figure Appendix 5.6: Bouncing box principle

The bounding box can be extended to the most complex situation by subdividing the scene into a hierarchy of boxes which each contain boxes, which contain boxes, and so on, as shown in Figure Appendix 5.7. The name of this technique is the Boundary Volume Hierarchy [298].

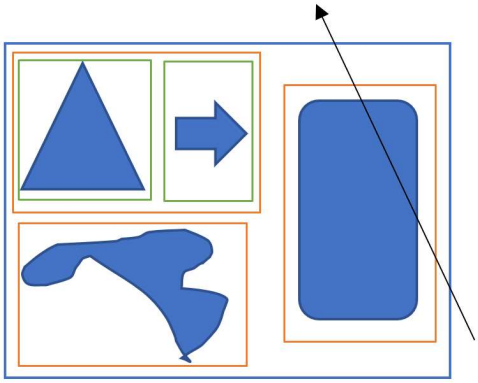


Figure Appendix 5.7: Illustration of the Boundary Volume Hierarchy

Ray tracing is based on connecting the source light to the camera, and to choose the right light paths direction to have this connection, before checking with the Boundary Volume Hierarchy to determine if the light beam hit an object or not. However, some material properties are difficult to handle with ray tracing, such as transparency. In Blender, transparency is unfortunately not equal to refraction. Glass can cast shadows but blocks the shadow rays because it is in the path of the light. The best solution will be to choose the light path direction as explained above. Nonetheless, this works only for diffuse materials (no specular reflection), but with mirrors and glossy material, the bounce direction is the same as the reflection direction.

Connecting the light source to the camera may therefore be the right option for glassy and glossy materials. However, if the light source is directly connected to the camera, the direction of the light will be fixed, because the position of the light source cannot be adjusted. Thus, the direction of the outgoing light cannot be chosen, leaving the possibility of choosing the wrong direction, for example, a direction where the mirror does not reflect, which leads to significant level of noise in the final rendering.

With bright, glassy materials and large light sources, the first solution should be considered. This solution consists of sending a light beam from the camera, in a random direction, with the hope that the light source will be hit.

Clearly one of these solutions will better than the other, depending on the context. However, to make life easier, the MSI (Multi Importance Sampling) method is generally used. This method is based on the two solutions explained above, but they are weighted according to the probability of success [298-299].

Image generation

Boundary Volume Hierarchy is the complete solution to solve the problem of simulating light in a virtual environment. Ray-tracing-based rendering engines (which will be presented in Chapter 4, Section 4.1.1) generate images related to the following method.

For each pixel composing an image, a certain number of light paths are traced. For each path, the engine will:

1. Follow it until it hits an object.
2. If nothing is hit, stop.
3. Evaluate the lighting interaction at the surface or of the volume, at the hit point.
4. Choose a random point on a light.
5. Try to connect to that random point.
6. Bounce into a new direction.
7. Go back to 1.

In Blender, its realistic rendering engine, Cycles, will translate the previous method as follow:

1. Synchronize Data from Blender.
2. Load GPU (Graphics processing unit) kernels if the GPU is used.
3. Pre-process data into the form that the renderer needs.
4. Send tiles to each computing device.
   Each device looks at a tile, at each pixel, and traces paths [298-299].

In Blender, tiles are small boxes that appear on the screen during the rendering process. These boxes form a grid and store all the information about the image in order to save memory and reduce rendering speed.