Department of Information Science and Technology

Depth Extraction in 3D Holoscopic Images

João Diogo Gameiro Medeiros

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master in Computer Engineering

Supervisor:
PhD Tomás Gomes da Silva Serpa Brandão, Assistant Professor,
ISCTE-IUL

Co-Supervisor:
PhD Pedro Figueiredo Santana, Assistant Professor,
ISCTE-IUL

October, 2018

Resumo

A holoscopia é uma tecnologia que surge como alternativa aos métodos tradicionais de captura de imagens e de visualização de conteúdos 3D. Para o processo de captura é utilizada uma câmera de campo de luz que permite armazenar a direção de todos os raios, ao contrário do que acontece com as câmeras tradicionais. Com a informação guardada é possível gerar um mapa de profundidade da imagem cuja utilização poderá ser útil em áreas como a navegação robótica ou a medicina. Nesta dissertação, propõe-se melhorar uma solução já existente através do desenvolvimento de novos mecanismos de processamento que permitam um balanceamento dinâmico entre a velocidade computacional e a precisão. Todas as soluções propostas foram implementadas recorrendo à paralelização da CPU para que se conseguisse reduzir substancialmente o tempo de computação. Para os algoritmos propostos foram efectuados testes qualitativos com recurso à utilização das métricas Mean Absolute Error (MAE), Root Mean Square Error (RMSE) e Structural Similarity Index Method (SSIM). Uma análise comparativa entre os tempos de processamento dos algoritmos propostos e as soluções originais foi também efectuada. Os resultados alcançados foram bastante satisfatórios dado que se registou uma redução acentuada nos tempos de processamento para qualquer uma das soluções implementadas sem que a estimativa de precisão tenha sido substancialmente afetada.

Palavras-chave: Mapa de Profundidade, Câmera de Campo de Luz, Imagem Holoscópica, Paralelização da CPU.

Abstract

Holoscopy is a technology that comes as an alternative to traditional methods of capturing images and viewing 3D content. A light field camera can be used for the capture process, which allows the storage of information regarding the direction all light rays, unlike the traditional cameras. With the saved information it is possible to estimate a depth map that can be used for areas such as robotic navigation or medicine. This dissertation proposes to improve an existing depth estimation algorithm by developing new processing mechanisms which provide a dynamic balancing between computational speed and precision. All proposed solutions were implemented using CPU parallelization in order to reduce the computing time. For the proposed algorithms, qualitative tests were performed using the Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Structural Similarity Index Method (SSIM). A comparative analysis between the processing times of the proposed algorithms and the original solutions was also performed. The achieved results were quite satisfactory since there was a significant decrease in processing times for any of the proposed solutions without the accuracy estimate being substantially affected.

Keywords: Depth Map, Light Field Camera, Holoscopic Image, CPU Parallelization.

Acknowledgments

Aqui se finda a última etapa desta história. Foi um caminho atribulado, mas ao mesmo tempo o mais gratificante que alguma vez percorri. Não foi fácil, mas com o apoio de todos aqueles que me acompanharam neste processo, consegui superar todas as dificuldades que me apareceram pelo caminho. Durante esta dissertação, pude contar com o apoio de pessoas extraordinárias, sem as quais não teria conseguido concluir este capítulo. E é a todas elas que agradeço profundamente por tudo o que fizeram por mim. Em primeiro lugar quero agradecer e dedicar esta dissertação aos meus orientadores, ao Professor Tomás Brandão e ao Professor Pedro Santana, pela paciência e confiança que depositaram em mim e no meu projecto ao longo de todo este trajecto, e pela forma exemplar e objectiva com que orientaram esta dissertação. Em segundo lugar agradeço às pessoas que tornaram tudo isto possível, os meus pais e, em último mas não menos importante, a todos os meus amigos e à minha namorada pelo suporte que me deram para conseguir concluir a dissertação. Termino esta etapa da minha vida com a sensação de dever cumprido.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This Chapter provides a general overview of the motivation and the context for the chosen theme as well as to specify some details related to the investigation carried out in this dissertation. Section 1.1 presents the motivations that led to the research and Section 1.2 explores the context in which this dissertation fits. The research questions that serve as a guideline are addressed in Section 1.3. The proposed objectives and the chosen research method are described in Section 1.4 and Section 1.5, respectively. Finally, in Section 1.6 the dissertation structure and organization is addressed.

1.1 Motivation

With the market trend and the proven benefits of the use of three-dimensional content in several areas, there is a need to create new mechanisms to improve the 3D experience. The holoscopic imaging, also known as light field technology is one of the alternative 3D systems that recently emerged and promising to overcome some limitations from other systems. This technology uses a camera with a single aperture lens and comprising an extra layer of microlens array between the image sensor and the main lens, allowing it to store the light rays captured from the scene. The extra information provides the viewer with a full parallax scene without requiring any calibration as other 3D systems need. As a result, the eye strain often associated with conventional 3D technologies is not felt in holoscopic images [1]. Initially, the significant issues from holoscopic technology were the depth of field limitation and low-resolution images. However, as the microlens manufacturing process has improved over the years, [2][3] these problems were overcome. The image quality produced by light field systems vastly improved, making it an at-

tractive three-dimensional technology. The holoscopic technology can be a superior alternative to other solutions on the entertainment market because it has no uncomfortable symptoms, provides a more realistic and immersive 3D experience, and no need to wear special glasses to view content [4][5]. For these reasons, the number of studies about light field images is significantly increasing in order to capture and display the information as fast as possible [6].

The light field camera is the device that allows to capture holoscopic images and is very similar to a traditional digital camera. However, unlike a traditional camera that only records the light rays entering in each pixel of the scene, the light field camera stores the direction of light rays in the sensor through its microlenses allowing it to collect all the information from the scene. These microlenses are tiny cameras that view the scene from all over the main lens, in other words, they can take a picture of the back of the main lens from different perspectives [7]. The light field cameras store the micro-images taken by microlens encoded in 2D format. Each micro-image has a reduced size due to the microlenses low resolution. The set of images captured by the array of microlenses need to be decoded and processed. One of the processing methods possible to apply to light field images is the depth map estimation. The depth map information may be useful for feature extraction, coding, interactive content manipulation and other applications in areas related to computer vision. For the collection of depth information, the light field images can be an alternative to laser scanning, stereo vision or structured-light techniques. The depth extraction based on holoscopic images can circumvent the high price of the laser, and it is also a passive system that does not need any interaction with the environment to construct the 3D model, unlike IR structured light pattern projection or laser scanning. The light field images do not require external calibration processes and are more likely to avoid occlusions due to camera's microlenses when compared with stereo vision systems. However, using light field images to estimate the depth shows some disadvantages like the price of the camera (although cheaper than a good quality laser) and the processing time [8].

1.2 Context

The 3D technology is a new way of seeing content and its market is currently on a large expansion and constantly seeking new solutions to please the users [6]. In the 3D field, some cues give a depth feeling that is not present in the 2D images [9]:

- Stereo parallax – the user sees a different image in each eye which gives a sense of depth;

- Motion parallax or Movement parallax – the user sees different images according to his position concerning the screen;
- Accommodation – the lens of the eyes focuses on a particular object of interest;
- Vergence – the left eye and the right eye converge to an object of interest.

As it attempts to provide a realistic experience similar to real-life scenarios, the competition in the 3D entertainment field is increasingly pronounced. Although there is a recent research interest for 3D, the technology was invented by Charles Wheatstone in 1838 by discovering the stereoscope. This device displayed a different image for the left and the right eye and, through the brain's processing, gave a 3D illusion [10]. The 3D glasses work similarly to the stereoscope technology. There are several techniques to create an illusion of depth and process stereo images. The most commonly used commercial solutions are:

- Active 3D or Active Shutter 3D systems – They use shutter glasses that run on batteries and block the information for the eye to which the image is not intended. For example, if the display device wants to convey an image to the left eye, the right eye shutter closes and does not allow the capture of any visual information for the right eye. The display will need to have the ability to present information at least 60 frames per second for each eye so that the viewing experience is the most appropriate and comfortable as possible [11].
- Passive 3D or Polarized 3D systems - They use polarized glasses like those used in a 3D movie theater. The display applies a special filter to the image in which the left eye sees the odd lines and the right eye the even lines. These glasses are much cheaper than those used in the active 3D systems [11].

The stereoscopic image presents disadvantages and limitations common to all techniques such as the lack of depth naturalness, eye strain or the 3D scene always displayed from the same perspective [12]. Later, and to overcome the stereoscopy limitations, began to appear the auto-stereoscopy. This new technology did not require any special glasses to observe the 3D image and as a result, became a more pleasant experience for the viewer [9]. Holography, light field images, multiple views (e.g., the lenticular lens and the parallax barriers) and, volumetric displays are examples of auto-stereoscopic technologies. The auto-stereoscopic systems

allow viewing different images from different perspectives in the same plane due to their optical components [6].

The multi-view technique is an auto-stereoscopic system that uses a combination of stereo parallax and motion parallax to create a 3D sense[13]. The multi-view methodology involves subdividing the viewing plane in a finite number of small windows so that one image is visible for each eye (known as stereo parallax) and, if the viewer moves with respect to the screen plane, he will see a different image pair (motion parallax) [13]. However, when used for an extended period, this method still have symptoms of discomfort as eyestrain or headaches [4].

Holography is another alternative that is emerging and evolving in the field of auto-stereoscopic technology. Dennis Gabor was the inventor of holography, a fact that has not gone unnoticed and which earned him the conquest of a Nobel Prize [14]. To capture a hologram is required to send two coherent light sources (one as an illumination beam in the direction of the scene and the other as a reference towards the holographic plate) by the laser. Some of the illumination light that reaches the scene is scattered/reflected toward the holographic plate which intercepts and interferes with the beam used as a reference. Is necessary to emit a coherent light source in the direction of the holographic plate to decode the reconstructed beam. This three-dimensional technique produces a high-quality final result where it becomes difficult to distinguish from the original shooting scene. However, it is a technology that needs a lot of expensive materials to produce adequate results making it difficult, at least for now, its incorporation into the market [15].

The holoscopic system is another auto-stereoscopic technology that seems to get around some of the limitations from different systems. This method was created in 1908 by Gabriel Lippmann. The holoscopy can capture the entire scene's light field by a set of microlenses installed on the machine. Unlike traditional equipment which can only record the intensity and the color, light field devices allow to capture all the light beams directions from the scene. The 3D scene reconstruction uses the stored directions. The machine used for the reconstruction also has a microlens array in order to correctly displays the captured information. The holoscopic generated image has high quality and with an authentic sense of depth and it is not as expensive as holography. Moreover, due to the constant technological evolution which allowed increasing the quality of microlenses and reducing the processing time of the captured image some of the initial limitations are starting to be solved [6]. The option of not requiring special glasses for 3D viewing, the full parallax, the three-dimensional realism, the multiple viewpoints observed de-

pending on the position of the user, and the elimination of uncomfortable symptoms are among the factors that make the light field technology a significant source of interest for researchers [6].

Not only for the digital entertainment holoscopy proves to be useful. The use of holoscopic images to estimate the depth from a scene is one example from the many areas that light field imaging technology can be applied. For the depth calculation, there are three methods widely adopted:

- Laser scanning – It is the most reliable method for the depth map estimation. It is an active system, i.e., it needs to interact with the environment to build the depth map and also gives no information about the color. It is the most expensive depth estimation system.
- Structured light – Projects into the scene an IR pattern and calculates its deformations to estimate the depth. Similarly to laser scanning, employs an active system for gathering information. Its use is not suitable for outdoors because the lighting conditions may adversely affect the quality of the data collected. However, some researchers have found a solution to overcome this problem [16].
- Stereo vision - As the humans' eyes, it uses two cameras separated by a given baseline. To find the depth, calculate the disparity between the images is needed. This technique is permeable to occlusions and requires a calibration process.

Although overcoming some disadvantages from other methods, the depth estimation using light field images presents some problems such as the high processing time [8].

1.3 Research Questions

The research presented in this dissertation was guided and focused on the following research questions:

- What techniques and algorithms should be used to calculate depth from light field images?
- Which mechanisms can be introduced to depth extraction algorithms to allow dynamic management of the speed-accuracy trade-off?

1.4 Research Objectives

The goal of this dissertation is to improve an algorithm already described in the literature for the depth map estimation on light field images. The selected algorithm was one of the most accurate in the depth map estimation. Modifications were made from the chosen solution in order to provide it a dynamic management mechanism for the speed-accuracy trade-off. Quality and performance tests were made to the proposed algorithms. In summary, the objectives were:

- Benchmark and compare depth estimation algorithms based on light field images;
- Select and improve an existing depth extraction algorithm with a better speed-accuracy trade-off;
- Evaluate the proposed algorithms for performance and quality.

1.5 Research Method

For the research process, the chosen paradigm was the Design Science Research because the main goal involves the construction of one or more artifacts and not to answer questions. While Section 1.3 described the questions to be solved, these are not the main objective of this dissertation and are used only as a guide to the research. The use of Design Science in the dissertation follows an approach centered on the problem and is divided by the following six phases [17]:

- First Phase – Expose the problem and defend the proposed solution for it. To accomplish this, it must have knowledge of state of the art and the importance of the advocated solution.
  - Dissertation Context: collect relevant information about the light field images and discover the already implemented depth extraction algorithms. Choose an algorithm and improve it in order to try to overcome some of the limitations found in the solutions described in the literature.
- Second Phase – Describe the objectives outlined for the proposed solution.
  - Dissertation Context: develop algorithms with a better speed-accuracy trade-off than other existing methods.

- Third Phase – Determine the artifact's features and architecture. Implement the artifact.
  - Dissertation Context: develop the depth extraction algorithms to meet the goals proposed in the second phase.
- Fourth Phase – Demonstrate that the artifact can solve the proposed problem.
  - Dissertation Context: demonstrate the application of algorithms in light field images to extract the depth map.
- Fifth Phase – Assess whether the artifact was able to achieve the proposed objectives in the Second Phase.
  - Dissertation Context: compare the efficiency and robustness of the algorithms already implemented with the proposed solutions. The performance of the developed algorithms will be related to the quality of the generated depth map and its processing time.
- Sixth Phase – Report the problem and its importance as well as the artifact developed as a solution to the problem.
  - Dissertation Context: writing this dissertation in order to communicate the problems, describes the developed algorithms and analyze the obtained results.

1.6 Document Structure

This dissertation is structured as follows:

- In Chapter 2 concepts related to light field technology are introduced. Some projects and works developed for depth extraction using holoscopic images are also approached and analyzed.
- From the literature review, a depth extraction algorithm was chosen to be improved. Chapter 3 explains with detail all the algorithm processes until depth map estimation is obtained.
- From the presented algorithm in Chapter 3, a group of modifications are made in order to optimize the original solution. One of these modifications, which is common to all

proposed algorithms, is the use of the central processing unit (CPU) parallelization, presented in Chapter 4. The specific modifications for each one of the proposed algorithms are also discussed.

- In Chapter 5 the obtained results for the proposed algorithms are analyzed. A qualitative and performance comparison is made with the algorithm originally implemented.
- Chapter 6 discusses the conclusions reached with this dissertation and presented some ideas for future development.

Chapter 2

Literature Review

In this chapter, the holoscopic technique for content viewing and image capturing is introduced, and some algorithms described in the literature for depth map extraction are also analyzed. Section 2.1 provides the evolution of holoscopic techniques over the years. Section 2.2 explains how the holoscopic method works in the process of capture and reproduction. It also presents an overview of some essential holoscopic technology features that distinguish it from other existing techniques. Section 2.3 presents some depth estimation techniques described in the literature. Lastly, the Section 2.4 presents a summary of the content covered in this Chapter and links to what will be discussed in the following Chapters.

2.1 The history and evolution of holoscopic technology

Although only in recent studies the holoscopic images (also known as light field images) have become relevant in the 3D field and digital photography, the idea behind the concept was created in 1908 by Gabriel Lippmann. He argued that with the adoption of a lens array, images with full parallax both vertically and horizontally could be captured [6]. The parallax means that the viewer of the scene sees a different image depending on their position relative to the screen. The set of convex spherical microlenses was placed over the sensor and thus could record all the 3D data of the scene with a single capture. The microlenses were essential not only in the capture process but also in the reproduction process as shown in Figure 2.1. These set of lenses can detect and store the scene's information from different perspectives [7]. The portion of the scene that is recorded by each lens from the array is called a micro-image. Micro-images are particular 2D images that allow reconstructing the final holoscopic 3D image [6]. This initial

version proposed by Lippmann presented several problems in the reconstruction process such as depth of field limitations [4].

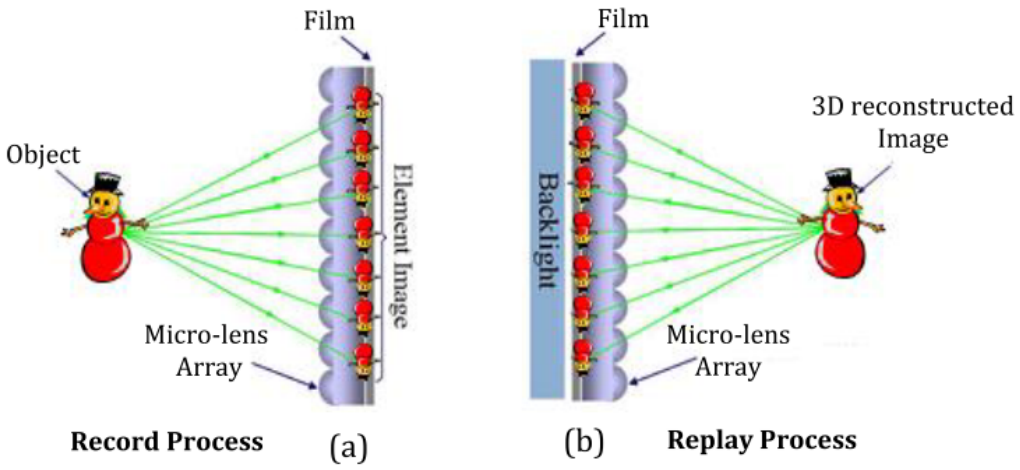


Figure 2.1: Principles of Holoscopic System: (a) scene capture process and (b) reconstruction of the recorded scene [6]

In 1911 Sokolov performed the first experience with light field images, but instead of lenses, he used the pinhole method for testing the system [18]. The test performed by Sokolov was satisfying because it could experience a sense of depth but has not been enough [19]. Image storage was one of the difficulties found in Sokolov experience. After the rebuilding process, the viewer saw an image with its inverted depth (pseudoscopic image) instead of the original captured scene (orthoscopic image). Ives noticed this problem in 1931 and also proposed the first approach to overcome this limitation [20]. The proposed solution was to capture the reconstructed image by passing it to the orthographic perspective. However, the method known as "two-step integral photography" caused the image to lose a substantial amount of resolution. In 1968 another solution was proposed to solve the problem of the pseudoscopic image by Chutjian and Collier [21]. Their proposal consisted of a computational reconstruction to create the orthoscopic image. Villums would later optimize this method in 1989 with the introduction of the objective lens as microlenses and the possibility to apply transparent color masks [22]. Despite constant efforts to improve the field of integral images, only recently it has started to recognize this technique as a reliable solution for the future. The machines processing speed, the improved quality of microlenses, and even the optimization in camera's sensors contributed to enhancing the quality of the holoscopic images [6]. In 2004 it was proposed to modify the lens shapes to ensure a higher viewing angle. However, this solution did not work because the curved lenses could not keep constant distances between them and it would be difficult to

estimate the disparity [23]. The conception of micro-images also changed, and since 2005 it became possible to produce higher resolution micro-images due to a new system [24]. The technique proposed to calculate the sub-pixel disparity between micro-images and the intensity values from the pixels to create a high-resolution micro-image. Other problems related to holoscopic images (e.g., converting pseudoscopic images to orthoscopic, limited depth of field or low-resolution images) were overcome due to the high number of researches in the field [25].

The developments allowed to discover domestic solutions that enabled the capture and the display of 3D holoscopic images. After the completion of its PhD about light field images in 2006, R. NG [7] launched the first light field camera on the market. It was called Lytro and could store all the light rays from a scene. With this extra information, the user could change the focal point on imaging post-processing. The company claimed that with the Lytro, blurred pictures would no longer be a problem [26].

With the technological improvement, the launching of light field cameras was generalized on the market. In the second generation of the Lytro camera, the information could be reconstructed in real-time causing the user to view the captured scene instantly. The machine also comes equipped with a lens that allowed manipulating the depth of field and optical zoom. The ability to view the image on a stereoscopy device and the quality of images are also features that served to improve the camera. The company plans to release the Lytro Immerge, a video camera that uses light field technology. This video camera will allow the capture of videos in 360 degrees and with full parallax. Also, each captured frame estimates the depth information which will be able to create a virtual reality environment [27].

In 2008, the Holovizio device was launched as the first domestic solution to reproduce light field content. For content playback, it used holographic technology, which, as holoscopic, it uses light fields to capture or display information. Each pixel from Holovizio emits light rays with different color and intensity in multiple directions. A light-emitting surface composed by these pixels creates a hologram to show 3D scenes. Fig. 2.2 shows the difference between the traditional 2D image and the technology used by Holovizio. This system does not cause eyestrain and does not require special glasses to view the 3D content. The latest versions of Holovizio already provide a horizontal parallax, i.e., if the viewer moves around the display, the image perspective also changes [28].

Although evolution is notorious, the equipment available for the public does not yet have the desired quality for the price it costs. The holoscopic technology has enormous potential,

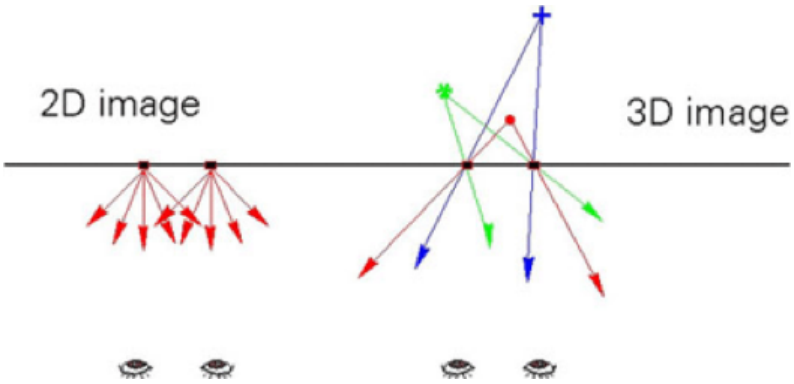


Figure 2.2: Main differences between 2D systems and 3D Holovizio systems [29].

however, still needs to be studied and improved to take greater acceptance by the market.

2.2 Principles and concepts behind the holoscopic images

The holoscopic technology allows the capture and the reproduction of a complete 3D scene by using the microlenses. The type of microlenses used may be a 1D lenticular sheet or a 2D microlens array [6]:

- **Unidirectional technique:** It uses a lenticular sheet to display 3D depth with horizontal motion parallax. Its micro-lenses are cylindrical.
- **Omnidirectional technique:** It uses a 2D micro-lens and gives the viewer a complete experience because it provides a depth 3D array with full motion parallax (the image shifts the perspective in all directions depending on the observer's position relative to the screen). Its micro-lenses are spherical.

2.2.1 Holoscopic Images Capture Process

The technology used in a light field camera intends to obtain all the information from the scene by storing all the light rays. The way it works is very similar to a traditional digital camera. However, the light field camera has an array of microlenses between the main lens and the sensor. The microlenses allow recording the luminosity and color as a standard camera does but also capture the direction of each light ray on the scene and store them in the CCD or CMOS sensor. Each of the captured images is called micro-image and correspond to photographs taken from different microlenses. Since the obtained micro-images are small, it is necessary to render higher resolution images [6]. To render images with greater quality it is required to arrange

each pixel in the same position from the obtained micro-images, which produces images from distinctive points of view. The total number of viewpoint images is equal to the number of pixels for each of the micro-images. Each of the generated pictures has a total of pixels equal to the number of microlenses in the array [30]. The reconstructed viewpoint images are orthographic projections rotated on a plane at a certain angle. Fig. 2.3 shows a summarized version of the orthographic projection for multiple images from distinct points of view. After obtaining the viewpoint images, it is possible to make changes in the post-edition process that were not achievable with the use of a conventional camera such as handling focal points, changing the depth of field, or shifting the perspective.

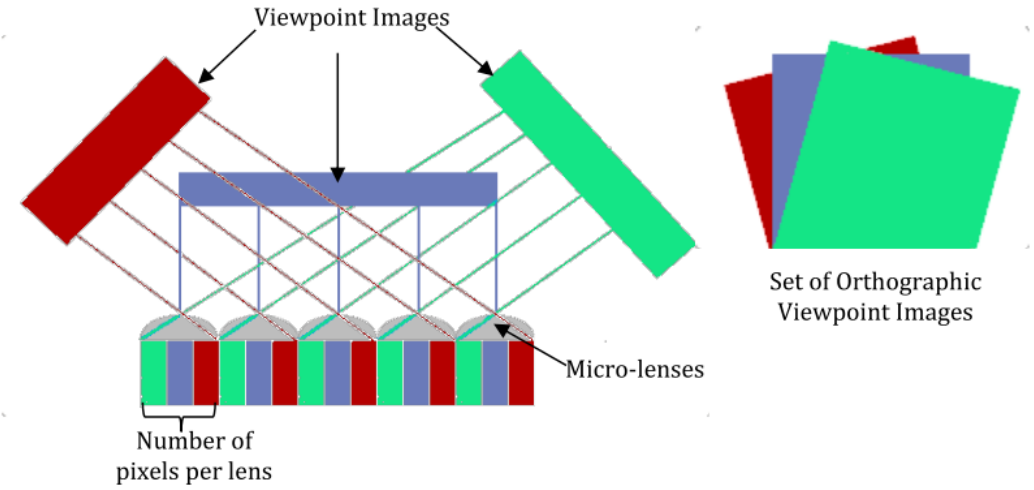


Figure 2.3: Orthographic projection of viewpoint images extracted from microlenses. Each viewpoint image consists of pixels in the same position in individually micro-image. For simplicity, it was assumed that each micro-image only has three pixels [6].

2.2.2 Holoscopic Images Playback Process

In the playback process, it is required to have microlenses similar to those used in the capture, because the technique used for reproduction is the reverse of the recording method [6]. The stored light rays make it possible to display the original scene on a screen prepared to support light field images. To select a holoscopic display system, the following features are considered:

- **Viewing zone** - The maximum viewing angle of the screen restricts the Viewing zone. A screen with a 180-degree angle is the ideal configuration because it shows all possible information regardless of the viewer's position.

- **3D resolution** - The smaller the distance between the center point of a microlens and its neighbor microlens (also known as pitch), the better will be the obtained 3D image resolution. When there is a shorter pitch, the resolution of the viewpoint images is higher because it means that there are more lenses in the array.

2.3 Depth estimation techniques in holoscopic images

A depth map corresponds to a 2D array (the image coordinates x and y matches the array row and column respectively) that stores the depth information for each pixel of the image (the z value). For the holoscopic images, the depth map is estimated based on the information captured by the light rays.

Over the years, related works focused on improving the generated image and in the viewing parameters for the light field images. The angle of vision improvement [31], the formation of an orthoscopic 3D image [32], as well as solutions for the depth of field limitation [33] and for the low-quality images [34] were some of the themes presented in the studies. Depth estimation applied to light field images is another commonly approached topic. The depth, as well as a valuable data to produce a better 3D reconstruction of the scene, can also be useful for computer vision areas, like virtual reality environments that blend information of real and synthetic images, robotics, medical imaging, and biometrics [4]. In the following sections, the main computational techniques to estimate the depth for images captured with a light field camera are analyzed.

2.3.1 Extracting Depth through the Point Spread Function (PSF)

The method to extract depth using the Point Spread Function was among the first to be adopted for holoscopic images, allowing to identify and match the intensity distributions between micro-images [35]. The PSF technique is applied to identify the holoscopic system settings, assuming the depth map calculation as an inverse problem (i.e., know the result and then calculate the causes, which, in this situation, is the depth map). However, estimating the depth as an inverse problem is susceptible to loss of information, caused by issues like extreme sensitivity to errors in the input parameters (i.e., a small error in a parameter may alter the outcome). This loss of information prevents the adoption of PSF approach to estimate the depth for real images due to the noise that the captured scene may contain [36]. Cirstea et al. [37] proposed two different

methods in order to avoid loss of information in the reconstruction process. One of the solutions used the Projected Landweber (PL) method, and the other followed the Constrained Tikhonov regularization process. Despite achieving good results, the tests performed for the implemented algorithms occurred only with numerical data simulations.

2.3.2 Extracting Depth through Disparity

Using disparity for depth estimation is widely adopted in the recent literature, due to the relationship between depth and disparity. The viewpoint images reconstruction is required to calculate the depth using disparity. Wu et al. [38] published one of the first studies using disparity to calculate depth. In this work also proposed a new technique that used various baselines, i.e., the variation between adjacent images was unequal. An improvement of the work was released a year later [39]. They proposed an improved algorithm which used the neighborhood values to estimate the depth in each pixel. The algorithm was tested with synthetic and real images, and the results demonstrated a shorter processing time compared to the solutions previously approached. However, if the background of the scene presents some noise, the depth map will not have the desired quality. For the proposed algorithm it is also required to identify the feature points manually, making the process error-prone and slower than an automatic process. Zarpalas et al. proposed two new methods to overcome the problems found in previous works [40]. Both approaches attempted to use two strong set of salient points (most known as anchor points) automatically identified and matched before 3D reconstruction. While avoiding some problems from previously implemented algorithms, the first method produced non-smooth depth maps and the second technique needed multiple optimization processes to estimate accurate depth maps, making the process computationally demanding. Also, the presented algorithms calculated the disparity with pixel accuracy, making both methods ambiguous for the feature points correspondence.

Stefan Heber and Thomas Pock proposed a methodology called Robust Principal Component Analysis (RPCA) to overcome some limitations from previously developed algorithms [41]. The disparity was calculated using a global features extractor as an alternative to local extraction technique. Instead of using a block of pixels around a pixel of interest, the global extractor uses the whole image to discover the correspondence between different views. All viewpoint images are compared with the viewpoint image in the central position of the array (know as central viewpoint image) and evaluated according to their similarity. The obtained

depth maps have a high accuracy rate. However, when there is a substantial depth variation, the estimation presents some noise.

2.3.3 Extracting Depth through Anchoring Graph Cuts

In 2012, Zarpalas *et al.* suggested a new algorithm based on graph cuts [42]. This algorithm uses the viewpoints images, apply a local descriptor to each of them, and establishes a features matching between images. Correspondences with a higher degree of certainty are chosen to anchor points. The algorithm uses the anchor points for the optimization method as input data. This improvement process aims to estimate the depth for each pixel as accurately as possible. The graph cuts are employed in the optimization process to find the depth faster and accurately. With synthetic images, is calculated a high-quality depth map. However, when applied in real-world scenarios, there are some problems in estimating the contours of the objects, especially the small ones, because of the viewpoint images low resolution.

Another algorithm emerged for the depth maps extraction [43]. The proposed technique uses two sets of slice images extracted from the captured micro-images. One of them is reconstructed based on a rectangular window and the other with a triangular window technique. The use of these two functions is necessary for generating different sets of images to be compared. A block of each image from the rectangular window set is compared with another block from the triangular window set at the same position. The mismatch between the blocks is calculated through the sum of absolute differences (SAD). After comparing all the images from the two sets, for each coordinate is chosen the lowest block matching error. The method was tested with real and synthetic holoscopic images, and the obtained depth maps were smooth, with high resolution and exhibited a low error rate.

In 2013, Yu developed an alternative algorithm using constrained Delaunay Triangulation for calculating the depth map [44]. The presented methodology finds the line segments in the light field image and performs a depth estimation for its endpoints using the line segment detector algorithm (LSD) [45]. In the next stage, the disparity is interpolated for the intermediate points of the line segment, and the disparity is calculated again for the endpoints. The estimated values are used as input parameters for the multi-view graph cut (MVGC) [46] that will try to figure out the best disparity label with the least cost. The results presented by this method are highly dependent on the estimation quality performed for the line segment disparity, that is, this solution will be highly error-prone if the value attributed to the disparity is not adequate.

More recently, a technique for calculating the disparity between viewpoint images using a sub-pixel accuracy [47] was proposed. This algorithm estimates the correspondences between each of the viewpoint images and the central viewpoint. The graph cuts are used as an energy minimization process to determine the best disparity for each pixel according to the detected matches. Lastly, the computed depth map from the graph cut step is subjected to an iterative refinement process for removing outliers and smoothing the image. The method was tested with both real and synthetic images, and the results were quite good compared to other techniques described in the literature. However, due to its various refining methods, it is a time-consuming process.

2.3.4 Extracting Depth through Focused Plane

It is possible to estimate the depth for the image through the focused plane. When the object is reconstructed in the correct depth plane, it appears focused. Otherwise, if the object is in a wrong depth plane, it seems out of focus. Due to the information the light field cameras can store, they can reconstruct the images with different focal planes and thus determine the depth for each object in the scene. Baasantseren *et al.* proposed in 2009 an algorithm that used the pixel weight factor (PWF) to estimate the depth from the focused planes [48]. The PWF is used to evaluate how much an object is focused and thus estimate the best depth for it by suppressing the incorrect depth planes. The results showed that the algorithm could estimate a quality depth map without noise. However, it has some difficulty in displaying the suppressed depth planes for each object because of the detected noise and object size limitations.

2.3.5 Extracting Depth through Optical Flow

Some problems related to the depth extraction from holoscopic images are the occluded regions in the scene for some viewpoint images. This limitation makes the produced depth map imprecise in the reconstruction of occluded objects and is called the quantization depth problem [6]. However, extracting the depth through optical flow can overcome the problem of occluded objects. The Optical Flow is a vector algorithm that estimates the motion between consecutive frames [49], [50]. An optical flow solution with sub-pixel accuracy was proposed for the depth estimation. This algorithm firstly creates the viewpoint images from the captured micro-images. The optical flow method is then used to calculate the motion vector with sub-pixel accuracy between consecutive viewpoint images. Upon completion of processing, a disparity

map is created and converted to a depth map. With the created depth map, the objects are clustered and segmented according to their depth information. These objects are then assembled in the central viewpoint image through point clouds. Thus, if there are occluded objects, they can be reconstructed. The reconstruction process is performed with triangular meshes. The algorithm was tested with real images and produced quality depth maps. It has also proven effective in the reconstruction of occluded objects.

2.3.6 Extracting Depth through Epipolar Plane Image

The depth extraction through Epipolar Plane Image was first explored in 1987 by Bolles [51]. A camera moving horizontally and capturing a scene from different perspectives was used. From the obtained estimation it is possible to build epipolar-plane images (EPI) which are intended to determine the motion between images. Each EPI is a geometry plane defined by the intersections between the cameras optical centers and a specific observation point in the image plane. The epipolar-plane images were used to calculate a depth map from the scene without some of the occlusion problems found in other systems like stereo vision or laser [52]. The system proposed by Bolles was improved and adapted by a research team linked to Disney [52]. The applied concept is identical to Bolles's work, however, it focused explicitly on light-field images where a depth map is generated using the parallel processing from the graphical processing unit (also known as GPU). The processing time has decreased, and the produced map became more accurate when compared to other existing solutions.

Tao *et al.* [53] proposed a solution where from the epipolar planes a depth map was estimated using defocus and correspondence. The technique was called Combinational Approach of Defocus and Correspondence (CADC) and can be divided into four distinct phases:

- Extract an epipolar image from the light field camera data;
- Displace every point from the epipolar image in a fixed direction. Shearing is the name of the process;
- Use the shearing image to calculate the angular resolution where each pixel has a higher contrast;
- Use the shearing image to calculate the angular resolution where each pixel has a lesser correspondence measure (in this case, the lower, the better);

- Combine the obtained information in the previous two steps with Markov Random Fields variables to produce a global depth map.

A window is used around a current pixel for the defocus and correspondence methods to increase the robustness of the result. Although the depth map has quality, the use of shearing causes an incorrect depth estimation for objects which are located far from the focal plane of the camera's main lens. The ambiguity of depth measurements due to the fixed window size is another of the found problems. For defocus, the out of focus zone sometimes is too large which also limits the correct measurement of the contrast [53].

In 2014, Sven and Bastian proposed a new depth extraction method called globally consistent depth labeling (GCDL) [54]. Unlike the other algorithms, this work did not perform any pixel matching between the viewpoint images. A local depth estimation is calculated for the vertical and horizontal gradients in all viewpoints, leaving each viewpoint image's pixels with two different alternatives for the disparity. The final depth map is then calculated by choosing the more reliable value for each pixel. The methodology was tested in both synthetic and real scenarios. Although it is computationally faster than the other compared solutions, the algorithm presented a significant error rate for objects that are too polished or with little texture. This approach also presents precision problems if the disparity between adjacent viewpoint images is less than two pixels.

2.3.7 Extracting Depth through Geometric Relations

In 2013, Heber *et al.* presented an alternative solution for calculating the depth map through geometric relations [55]. The methodology was based on the technique of active wavefront sampling (AWS) [56] allowing to capture the three-dimensional scene using a traditional camera and an AWS module. The AWS module consists of an aperture axis misaligned with the optical system axis and which, through its circular motion, captures the scene from several different angles. With AWS, it is possible to extract the depth from the scene with the radius information for each of the rotations made by the aperture system. After determining the rotation radius for each of the captured images, it is possible to predict the depth map based on geometric relations between the rotations. Heber *et al.* [55] proposed to use the rotation technique with light field images where the extracted viewpoint images were aligned as a circle around a central view. In this way, it was possible to calculate the radius for each rotation and thus generate a depth map. The algorithm was tested in both synthetic and real images, and the presented results are

satisfactory, although the error rate per pixel is higher than other depth extraction techniques [47].

2.4 Summary

Throughout this Chapter, several concepts related to light field technology have been introduced. Some of the advantages that can be obtained by using this type of technology, either for visualizing three-dimensional content or for the capturing process. Depth map estimation is one of the advantages presented in this chapter as an alternative to the most commonly used 3D scanning systems such as laser or stereo vision. Some of the techniques that can be used to calculate the depth using holoscopic images are also analyzed: Point Spread Function, Disparity, Anchorage Graph Cuts, Focused Plane, Optical Flow, Epipolar Plane Image and Geometric Relations. For each of the techniques, the algorithms already implemented and described in the literature are studied. All the approached solutions presented some limitations such as the lack of precision in the estimated depth map or the high computing time. Among the several algorithms, one of those that can estimate the depth accurately, although it presents some problems in the processing time, is the one proposed in the work of Jeon *et al.*[47]. This algorithm was chosen to be improved. In Chapter 3, its implementation is analyzed in detail, and in Chapter 4 proposes four different algorithms based on this work.

Chapter 3

Original Algorithm

A depth map is an image containing information about the distance from the objects at a scene to the camera, and it can be useful for several applications in areas such as medicine, robotics and, three-dimensional entertainment. Several mechanisms have been explored over the time such as laser or stereo vision, however, an alternative based on light-field images begun to emerge. Through the use of a particular camera, it is possible to store all the light rays present in a scene which allows computing the depth map.

Although it has immense potential, the light field technology still needs to evolve in the depth map estimation because the currently developed algorithms still have some disadvantages that need to be improved. One of the most common problems is the processing time for estimating the three-dimensional structure of the scene. None of the algorithms mentioned in Chapter 2 achieve a relatively low processing time for use in real time situations like robotic navigation. Still, the algorithm proposed by the research team of Disney [52] was the algorithm with better performance by using parallelization mechanisms in the GPU. Another issue often found in depth estimation algorithms is the lack of precision and the low resolution of the generated three-dimensional map [35], [38–40]. While not strictly limiting, most of these algorithms are not prepared to be used with images extracted from lenslet array cameras (like Lytro or Raytrix) as they were designed to be applied with unidirectional images datasets. These unidirectional images can be produced synthetically or with a normal camera when capturing multiple photos from different viewpoints along an imaginary horizontal line and with a specific spacing between them. In this situation, only the horizontal displacement is considered, unlike light field cameras that can take omnidirectional pictures, i.e., allowing to capture the horizontal and vertical displacement. Another difference between omnidirectional and unidirectional images is the

displacement between consecutive pictures. In unidirectional images taken with regular cameras, the displacement between the images is always greater than that for the omnidirectional images captured with a light field camera. The baseline between images is an essential factor to be considered for choosing the correct depth estimation algorithm when the disparity technique is applied. A developed algorithm for estimating the disparity in images with a larger baseline, such as unidirectional pictures or stereo vision, can not be used for a lower displacement baseline, such as omnidirectional images. The inverse is also true, i.e., an algorithm prepared for a narrow baseline cannot be used for the disparity estimation in wide baselines. The algorithms based on unidirectional images can precisely estimate the disparity with pixel accuracy while the omnidirectional images, due to its smaller displacement, require subpixel precision.

With the purpose to extract an accurate depth map from the data produced by the light field cameras, this dissertation focused on improving the algorithm developed by Jeon et al. [47]. This algorithm was chosen among several alternatives because this is the one that creates depth maps with good-enough quality and it is also prepared to be used with data extracted from light field cameras as it uses the disparity technique with subpixel precision. Throughout this chapter, all the original algorithm steps are explained.

3.1 Algorithm Overview

The algorithm developed by Jeon et al. [47] presents exciting results with a significant degree of accuracy. However, as the vast majority of algorithms for depth map processing, it has a high computational time. This dissertation aims to find a balance between the speed-accuracy tradeoff without having a substantial loss of quality for the depth map estimation. It is necessary to obtain a better understanding of how the original algorithm works to achieve the goals, and this is the purpose of the following sections in this Chapter.

Figure 3.1 depicts the general diagram of how the algorithm operates. Firstly, the data captured with the light field camera are converted into a set of viewpoint images. These viewpoint images are multiple reconstructions of the scene viewed from different perspectives. They are generated from the micro-images captured and stored in the photosensor of the light field camera. The viewpoint images are then converted to the frequency domain using the Fourier transform. With the application of the Fourier transform, it becomes possible to rebuild the shifted viewpoint images with sub-pixel precision. The applied value for the shifting is conditioned by

the image position in the set of viewpoints and the label for which it is being calculated. The label corresponds to a displacement hypothesis for the viewpoint image. A comparison is made between each of the shifted viewpoint images and the central viewpoint in order to find the correspondences. For the comparison, the sum of absolute differences (SAD) for RGB images and the sum of the gradient difference (GRAD) for the vertical and horizontal gradients are used. The SAD and GRAD for the viewpoint images of a given label are summed. At the end of the process, a cost volume array is obtained corresponding to the sum of the values for each of the labels. The total number of labels is an input parameter for the algorithm. In the next stage, the cost volume array is subjected to an outliers removal process through the use of a weighted median filter. This filter uses the values from the neighboring pixels for the outliers removal. In order to create a map with higher quality, an energy optimization method based on graph cuts is also used to propagate the most reliable labels for the points in the neighborhood with a higher degree of uncertainty. Finally, there is used an iterative refinement process in order to estimate a depth map with higher accuracy. In the following Sections, these procedures are detailed.

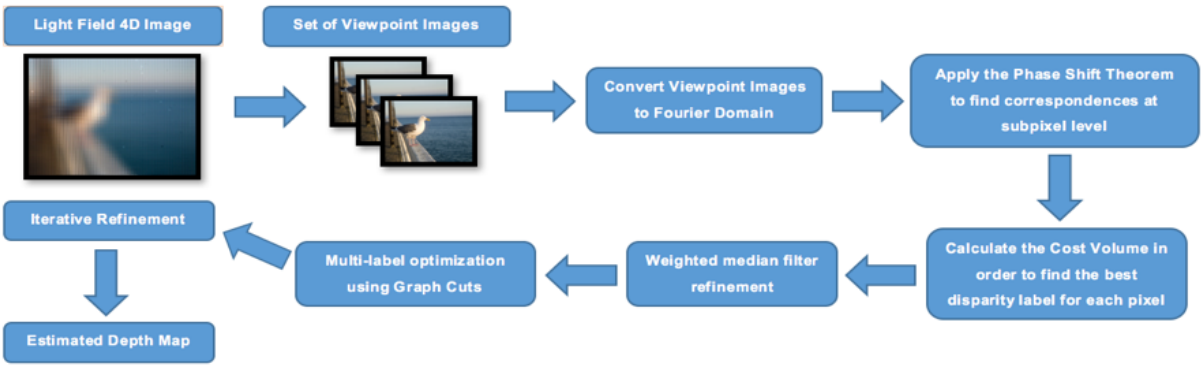
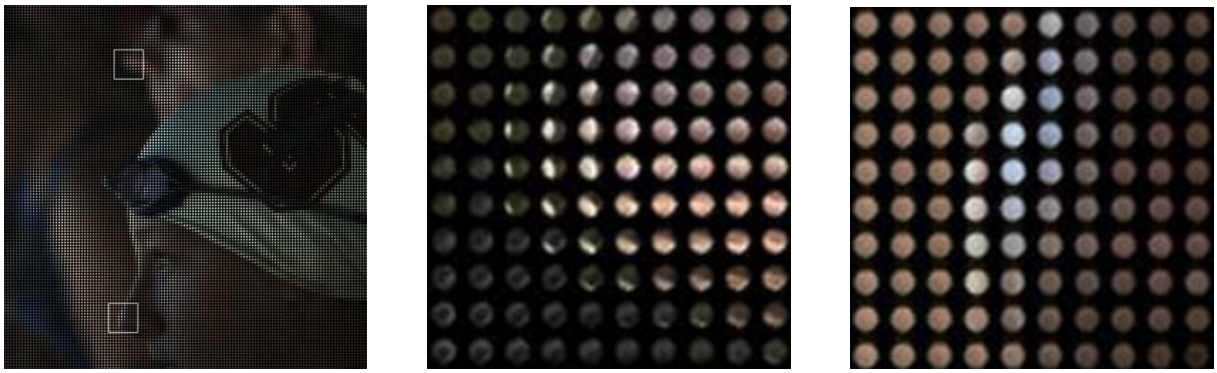


Figure 3.1: General Overview of the Algorithm

3.1.1 Extract the Viewpoint images from a Light Field Camera

When capturing a scene with a light field camera, the obtained raw file is not perceptible, as can be seen in Figure 3.2(a). The zoomed images (Figure 3.2(b) and Figure 3.2(c)) show that the image is composed of small micro-images each one of them captured by a different micro-lens belonging to the light field camera's array. While some of the algorithms for depth map calculation described in the literature use these micro-images to find matches, the algorithm proposed by Jeon *et al.* [47] needs to reconstruct the viewpoint images from the raw file to estimate the scene's disparity. It is necessary to obtain the pixel in the same position in all the

microlenses belonging to the array to reconstruct each of the viewpoint images. Thus, the total number of produced viewpoints is equal to the number of pixels for each microlens. In its turn, the viewpoint resolution depends on the number of microlenses belonging to the array. In the end, an ordered array of viewpoint images corresponding to the position of each pixel used for its reconstruction is obtained. For the work presented by Jeon *et al.*, the Light Field Toolbox was used to process the viewpoint images. This tool requires a calibration file with information about the capture device for an accurate extraction of the viewpoint images.



(a) (b) (c)

Figure 3.2: Holoscopic Image in RAW format (a) with two enlarged pictures ((b) and (c)) where it is possible to see multiple microimages captured by the microlenses of the light field camera [7].

3.1.2 Phase shift Theorem

Since the mechanism adopted to identify the correspondences is built for situations where stereo vision is used [57], it is necessary to apply a phase shift theorem with subpixel precision in order to increase the baseline. According to the theorem, if an image is shifted by Δx pixels, its phase shift in the frequency domain is given through its Fourier transform multiplied by an exponential term. When the multiplication is done, it is necessary to move the image back to the spatial plane. For this, the inverse of the Fourier transform is applied [47]. Thus, it is possible to obtain the shifted image I' by using the following equation:

$$I'(x) = I(x + \Delta x) = \mathscr{F}^{-1}\{\mathscr{F}\{I(x)\}e^{2\pi i \Delta x}\}. \tag{3.1}$$

By using the phase shift theorem, it is possible to construct shifted images with more detail and less noisy than with bilinear or bicubic interpolations [47], making the depth estimate more accurate. For the calculation of the displacement vector Δx the following formula is used:

$$\Delta x(s, l) = lk(s - s_{\mathrm{c}}), \tag{3.2}$$

where k is a value representing the baseline between consecutive microlenses. The values used for k are proposed by the work of Jeon *et al.* [47] according to the camera with which the light field image was captured. l is the index of each of the labels and s corresponds to the position of the viewpoint image in the ordered set of viewpoints and is subtracted by the relative position of the central image s_c. The label index and the viewpoint image relative position allow estimating different displacements with subpixel precision for the set of viewpoint images. The calculated values are grouped by label into a cost volume explained in the next section.

3.1.3 Cost Volume

To identify correspondences between the viewpoint images, the sum of absolute differences $\mathrm{C_A}$ (SAD), as well as the sum of gradient differences $\mathrm{C_G}$ (GRAD), are used for calculating the correspondences between the viewpoint images and the central viewpoint image. The calculated matrices from the sum of SAD and GRAD for all viewpoint images of a given label are grouped and placed in a cost volume array. The total number of labels chosen for calculating the cost volume has to be a small value so that the processing time and memory utilization does not increase substantially and yet high enough for the depth map to be accurate. Considering this trade-off, the number of labels used in the work of Jeon *et al.* was set to 75. The following equation determines the cost volume:

$$C(x, l) = \alpha C_{\mathrm{A}}(x, l) + (1 - \alpha) C_{\mathrm{G}}(x, l) \tag{3.3}$$

,

where $\alpha \in$ [0,1] is used to determine the relative weight of each term used in the cost volume function. The sum of absolute differences (C_{A}) is calculated using the following formula:

$$C_{\mathrm{A}}(x, l) = \sum_{s \in V} \sum_{x \in R_{\mathrm{X}}} min(|I(s_{\mathrm{c}}, x) - I(s, x + \Delta x(s, l))|, \tau 1), \tag{3.4}$$

where $I(s_{\mathrm{c}}, x)$ is the central viewpoint image, R_x is a region centered at x and V comprises the index coordinates for the viewpoint images array except for the central viewpoint image. The shifted viewpoint images from the set are represented by $I(s, x + \Delta x(s, l)$ and $\tau 1$ is only

a truncation value in order to ensure that the result of the function is not more than a given threshold. Equation 3.4 calculates the correspondence cost by comparing the central viewpoint image with the other sub-pixel shifted viewpoint images. The obtained result estimates a disparity map from a single point of view. For the cost volume calculation to be complete, it is necessary to find the sum of gradient differences:

$$\begin{aligned} C_{\mathrm{G}}(x,l) = \sum_{s\in V}\sum_{x\in R_{\mathbf{x}}} \beta(s) min(|I(s_{\mathrm{c}},x) - I(s,x+\Delta x(s,l))|,\tau 2) \\ +(1-\beta(s)) min(|I(s_{\mathrm{c}},y) - I(s,y+\Delta y(s,l))|,\tau 2). \end{aligned} \tag{3.5}$$

Equation 3.5 calculates the difference between the horizontal and vertical gradients from the central image and the viewpoint images. The weight of each one of the directional gradient differences is given by:

$$\beta(s) = \frac{|s-s_{\mathrm{c}}|}{|s-s_{\mathrm{c}}| + |t-t_{\mathrm{c}}|}, \tag{3.6}$$

where s_c and t_c represent the position of the central image in the viewpoint images array while s and t correspond to the viewpoint image position being compared.

The cost volume calculation generates an array of matrices whose size corresponds to the total number of labels supplied as input parameter. From the cost volume array, it is possible to extract a depth map using the winner-takes-all strategy, where, for each of the pixels, the minimum value present in the set is selected.

3.1.4 Weighted Median Filter (WMF)

Although the cost volume process mentioned in the previous Section is enough to extract a depth map from the light field image, the final result has some noise, as can be seen in Figure 3.3(b). To overcome this situation, submit the cost volume to a process of refinement and removal of outliers is necessary. The Constant Time Weighted Median Filtering proposed by Zyang Ma et al. [58] was used. This technique has a lower computational cost compared to traditional weighted median filtering because it avoids the use of a brute-force implementation and introduces a constant time value for algorithm processing [58]. For the calculation of weighted median filtering the following equation is used:

$$C'(\mathrm{x}, l) = \sum_{\mathrm{x'} \in N(\mathrm{x})} w(\mathrm{x}, \mathrm{x'}) C(\mathrm{x'}, l), \tag{3.7}$$

where$N(\mathrm{x})$ represent a local window around x and x' the median value for the neighboring points. In order to estimate the median values for the neighborhood, the box filter was applied. This technique sums all the values inside the window and divides by the total number of elements. $w(\mathrm{x}, \mathrm{x'})$ is the edge-aware filter applied to the cost volume matrix with l index ($C(\mathrm{x'}, l)$). For this case, the edge-aware filter chosen to be used was the guided filter that can be calculated using the following Equation [59]:

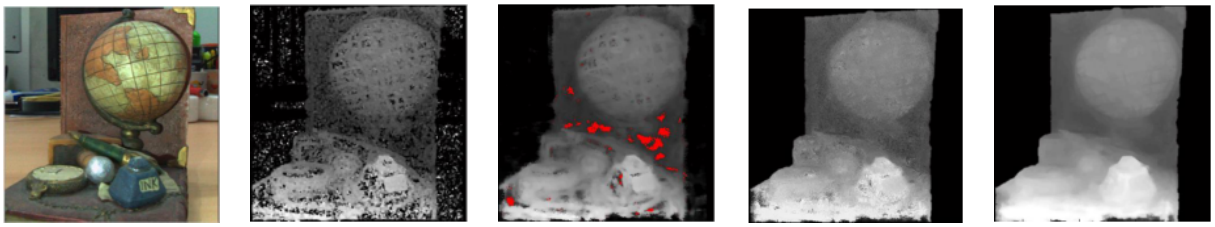
$$w(\mathrm{x}, \mathrm{x'}) = \frac{1}{|\omega|^2} \sum_{q:(\mathrm{x},\mathrm{x'}) \in \omega_q} \big(1 + (I(s_\mathrm{c}, \mathrm{x}) - \mu_q)^T (\Sigma_q + \epsilon U)^{-1} (I(s_\mathrm{c}, \mathrm{x'}) - \mu_q)\big), \tag{3.8}$$

where μ_q and $\sum_q$ represent the mean vector and covariance matrix of the guidance image (which in this case is the central viewpoint image $I(s_\mathrm{c}, \cdot)$) in the window ω_q centered at pixel q. The window size is given by $2r + 1$ X $2r + 1$ pixels where r represents the radius. U represents the identity matrix and $|\omega|$ it is the total number of pixels for the identity matrix. ϵ is a smooth parameter constant. The covariance matrix and the mean vector require the color information (RGB channels) of the guidance image so that they can be calculated. Therefore, the following Equations were used:

$$\Sigma_q = \begin{bmatrix} \sigma_q^{rr} & \sigma_q^{rg} & \sigma_q^{rb} \\ \sigma_q^{rg} & \sigma_q^{gg} & \sigma_q^{bg} \\ \sigma_q^{rb} & \sigma_q^{gg} & \sigma_q^{bb} \end{bmatrix} \text{ where } \sigma_q^{[1][2]} = \frac{1}{|\omega_q|} \sum_{\mathrm{x} \in \omega_q} I^{[1]}(s_\mathrm{c}, \mathrm{x}) I^{[2]}(s_\mathrm{c}, \mathrm{x}) - \mu_q^{[1]} \mu_q^{[2]} \text{ for } [1], [2] \in \{r, g, b\} \text{ and,} \tag{3.9}$$

$$\mu_q = \begin{bmatrix} \mu_q^r \\ \mu_q^g \\ \mu_q^b \end{bmatrix} \text{ where } \mu_q^{[1]} = \frac{1}{|\omega_q|} \sum_{\mathrm{x} \in \omega_q} I^{[1]}(s_\mathrm{c}, \mathrm{x}) \text{ for } [1], [2] \in \{r, g, b\}. \tag{3.10}$$

After the weighted median filter has been applied, it is possible to estimate the l_a depth map using the winner-takes-all strategy. Figure 3.3(c) shows the depth map estimate with the detected outliers marked in red.



(a) (b) (c) (d) (e)

Figure 3.3: The depth maps generated after each step where the Figure (a) is the central viewpoint image. (b) corresponds to the depth map generated after calculating the cost volume and without any type of optimization. (c) is the depth map after the weight median filtering algorithm was used while (d) and (e) are the generated depth maps after the multi-label improvement and the Iterative Refinement process respectively [47] .

3.1.5 Multi-label optimization using Graph Cuts

It is necessary to identify strong correspondence points between the central viewpoint image and the other images to use graph cuts for the multi-label optimization. In addition to the found matches in the cost volume calculation, the correspondences in the salient zones are detected using the SIFT algorithm principles[60]. For this particular problem, the feature points from the central image are calculated and an attempt is made to find candidates in other viewpoint images using the Euclidean distance. The points chosen are divided into clusters of at least three points and compared to the database of features extracted from the central image. Each group of points is approved if the object location, scale, and orientation are similar to the features present in the reference image otherwise, they are considered outliers. Using the cluster is more effective than using individual salient points because the probability of detecting outliers is higher. The next phase submits the candidate clusters to a computationally more demanding test by comparing the points with the pose calculated for the object in the central image. This pose comparison uses the least squares method. The outliers are removed again, and the remaining features are subjected to a final test for calculating the probability of wrong matches, according to how the features fit the salient points estimated for the central image. The points approved in all the tests are considered correspondences with a high degree of certainty. Matches for the same feature point may be in different viewpoint images. In this case, the mean values of the matches are calculated. This value is used to calculate the disparity confidence l_c.

A multi-label optimization process using graph cuts is applied for correct the pixel disparity. The following equation is used:

$$l_r = argmin_l \sum_{\mathrm{x}} C'(\mathrm{x}, l(\mathrm{x})) + \lambda_1 \sum_{\mathrm{x}\in P} \|l(\mathrm{x}) - l_a(\mathrm{x})\| \\ +\lambda_2 \sum_{\mathrm{x}\in M} \|l(\mathrm{x}) - l_c(\mathrm{x})\| + \lambda_3 \sum_{\mathrm{x'}\in N(\mathrm{x})} \|l(\mathrm{x}) - l(\mathrm{x'})\|, \tag{3.11}$$

where P represents the features calculated with the cost volume, $N(\mathrm{x})$ is a window around x and the M, contains the points with a high degree of confidence calculated through the use of the SIFT algorithm. The λ values are input parameters that allow to manage the relative importance of terms. For the energy minimization process of l_r, four different expressions are used:

- Accuracy of correspondence cost - (C'(x, l(x)));
- Information integrity - ($\|l$(x) - l_a(x)$\|$);
- Confident matching cost - ($\|l$(x) - l_c(x)$\|$);
- Local Smoothness - ($\|l$(x) - l(x')$\|$).

Figure 3.3(d) depicts an example of a depth map after using the multi-label optimization process. A comparative analysis is performed between the original image and the estimated disparity map, showing that the result is already quite acceptable; however, if an iterative refinement process would be applied, would be possible to improve the quality of the depth map further, especially in areas with higher salience.

3.1.6 Iterative Refinement

Although iterative refinement is an optional step to produce a depth map, it is a process that significantly reduces the error percentage. The algorithm proposed by Yang *et al.* [61] was adapted for this work to perform the iterative refinement step. An iterative refinement step is applied to the depth map. The depth map estimated by the energy minimization process is used as input parameter and is subject to an iterative refinement where the cost volume is calculated. An weight median filter is applied to the cost slices (each matrix from the cost volume array) to soften the most salient areas. After using the filter, a new depth map is calculated using the winner-takes-all strategy and serves as input parameter l_r for the following formula:

$$l* = l_r - \frac{C(l_+) - C(l_-)}{2(C(l_+) + C(l_-) - 2C(l_r))}, \tag{3.12}$$

where $C(l_+)$ and $C(l_-)$ are the values of the neighboring cost volumes and, $C(l_r)$ is the cost volume for l_r. The iterative refinement process is repeated four times. After the cycle ends, $l*$ corresponds to the final estimated depth map. Figure 3.3(e) shows the depth map quality improvements after the iterative process of refinement.

3.1.7 Results and Performance

The algorithm was originally developed using Matlab and was tested on synthetic datasets and in images extracted from light field cameras. Although the presented results have a lower percentage of error than the other existing algorithms and allows to generate reliable depth maps in both the synthetic and real world images, computation time is still the main drawback. The tests carried out by Jeon *et al.* [47] were performed on a 3.70 GHz CPU with 16 GB RAM and took about 25 minutes to generate depth maps for synthetic images and 6 minutes for images extracted from light field cameras.

Synthetic Images

The process developed by Jeon *et al.* [47] was compared to three other algorithms: 1. Active wavefront sampling (AWS) [55]; 2. Globally consistent depth labeling (GCDL) [54]; 3. Robust PCA [41].

The algorithm was also modified and tested with other sub-pixel shifting methods: 1. Bilinear; 2. Bicubic.

For the comparison of results, the methodology proposed by Heber *et al.* [41] was used. A relative depth error analysis was performed based on the percentage of pixels where the disparity error is greater than 0.2%. The percentage of GCDL errors was calculated based on the original source code provided by the authors. However, for both AWS and Robust PCA, the source code was not available and thus used the errors percentage mentioned in the work from Heber *et al.* [41].

As can be seen in Figure 3.4, for each of the synthetic images tested, the error percentage of the Jeon *et al.* [47] is always below any of the other algorithms. It is also possible to verify that both bilinear and bicubic approaches to sub-pixel shifting are inferior to the new method

proposed by the author [47].

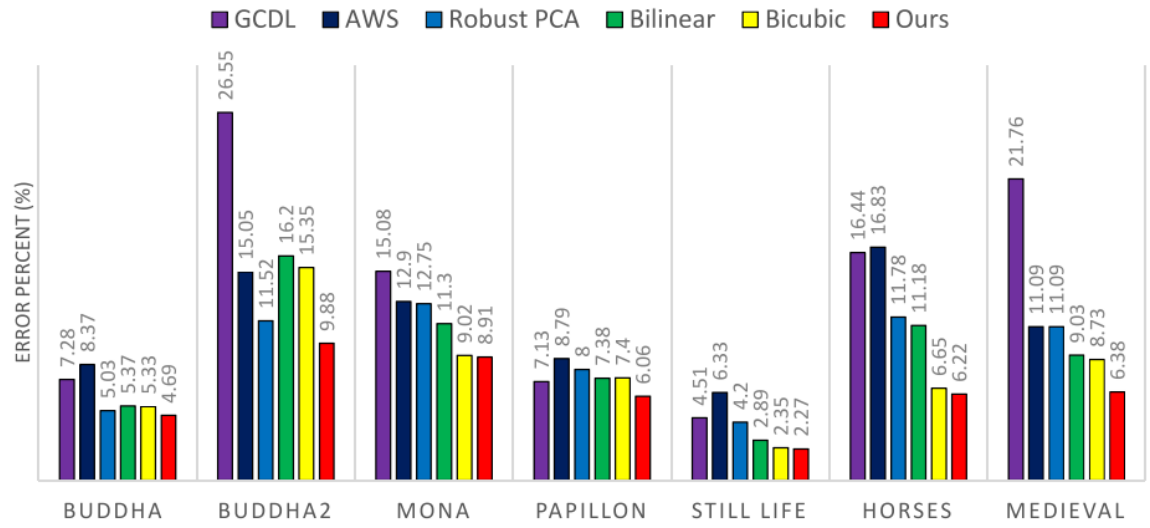


Figure 3.4: Comparison of results between the algorithm developed by Jeon *et al.*, AWS, GCDL and Robust PCA. It is also possible to compare the relative error rate between the solution developed by the original algorithm to calculate the sub-pixel shift with the bilinear and bicubic approaches.

A qualitative analysis was also performed between the GCDL and the Jeon *et al.* [47] algorithm. Figure 3.5 shows a lower error rate from the Jeon *et al.*[47] compared with the GCDL solution.

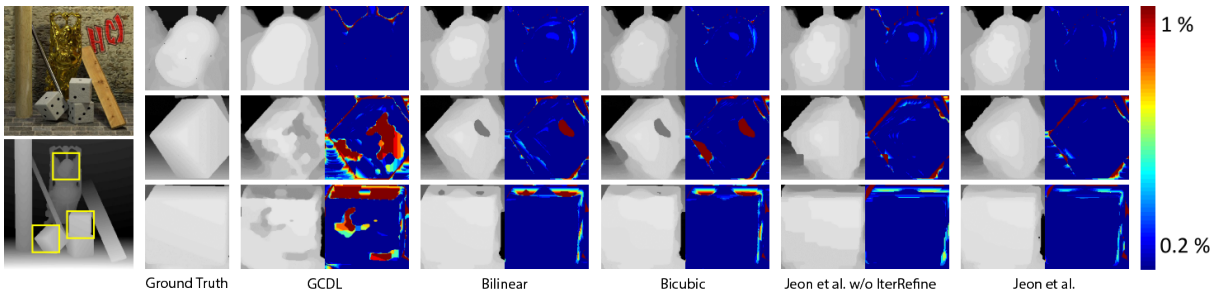


Figure 3.5: Qualitative comparison of results between the original algorithm and GCDL. It is also possible to compare the bilinear and bicubic approaches for sub-pixel shifting and the algorithm without the iterative refinement phase with the original methodology [47].

Real World Images

The method [47] uses three different light field devices in the tests, as the quality and the features of the camera are also important factors when calculating the depth of the scene:

- **Lytro camera** - As can be seen in Figure 3.6, a comparative analysis was made between the tested algorithms for two real world scenes captured with the Lytro camera. The

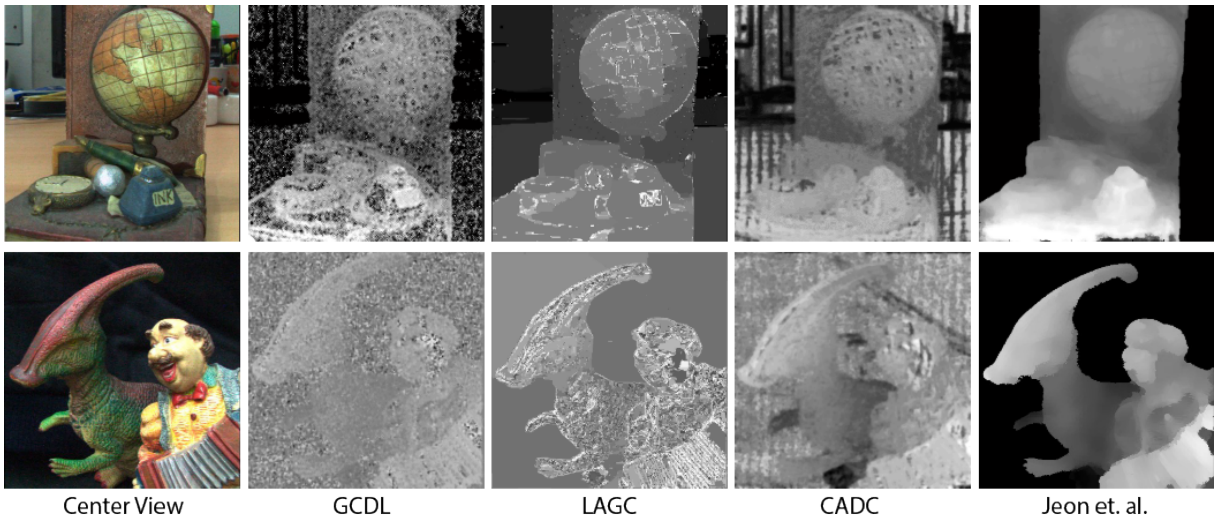


Figure 3.6: Comparison of results for light field images captured from Lytro camera. [47]

GCDL [54] presents some noise because the use of structured tensors in an image as complex as the one that the Lytro captures is not enough for an accurate calculation for the depth of the scene. The LAGC [44] also presents a poor quality due to the low resolution of the viewpoint images that are used as input parameters for the calculation of correspondences. The CADC [53] turns out to be the one that presents the best results among the algorithms that are compared with the method of Jeon *et al.* [47]. Making use of its ability to combine defocus with correspondence, it can generate reliable depth maps. However, the estimated depth becomes noisy in homogenous texture areas due to errors originated by the features matching procedure. As verified in the synthetic images, also in the real-world light field images, the algorithm developed by Jeon *et al.* [47] presents more reliable depth maps because it is the one that is better prepared for the detection and removal of outliers.

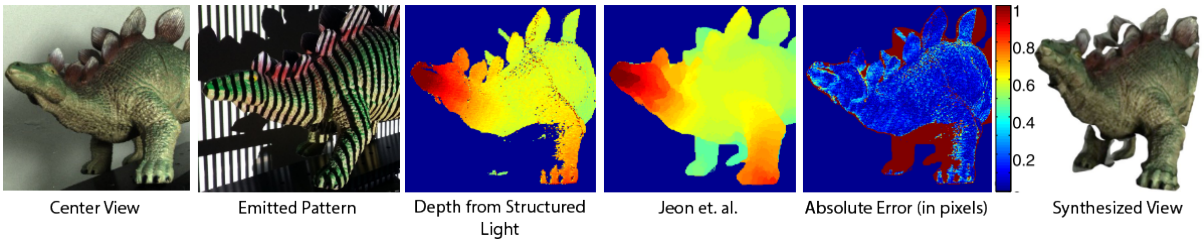


Figure 3.7: Comparison of the depth extracted from a structured light system with the algorithm of Jeon et. al. [47]

Figure 3.7 shows a depth map estimated with structured light technology that is used as ground truth to calculate the error per pixel of the Jeon *et al.* [47] method. It is possible

to verify that the error per pixel is inferior to 0.2 pixels of absolute error, where there are no occlusions.

- **Raytrix camera** - For this equipment the methodology studied [47] is compared with the depth estimation algorithm incorporated in the Raytrix software, GCDL [54] and, LAGC [44]. The Figure 3.8 shows that the Raytrix algorithm can not correctly estimate the boundaries of the objects present in the scenario because it only uses the SAD cost volume for the disparity calculation. For the GCDL, the estimated depth map is better when compared to the same algorithm applied to images captured with Lytro, although it presents some flaws estimating the edges disparity. Otherwise, the LAGC reveals an inverse behavior, i.e., it makes a good estimate for the edges; however, the depth estimation inside objects presents some problems. For the raytrix camera, the studied algorithm [47] was able to estimate a more accurate depth map than any other of the solutions to which it was compared.

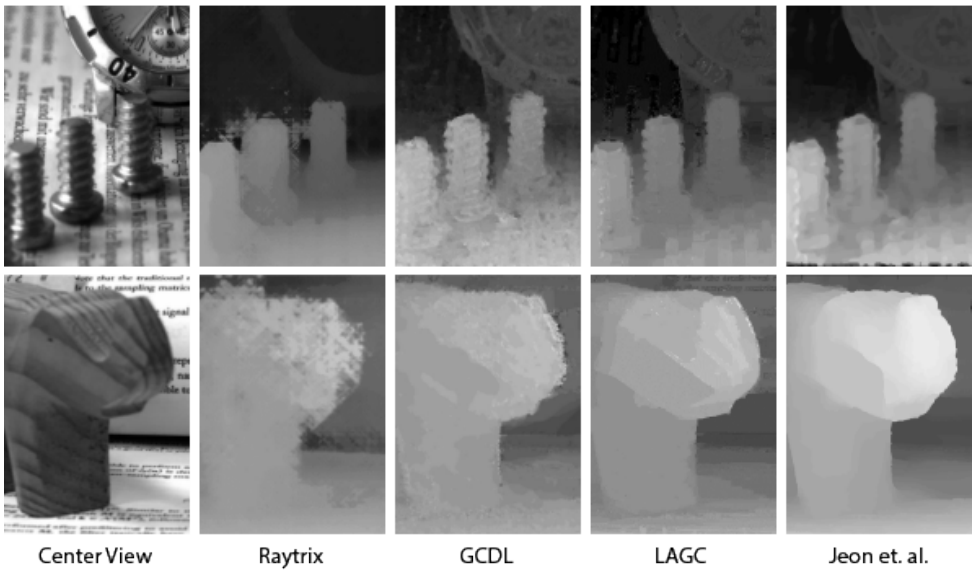
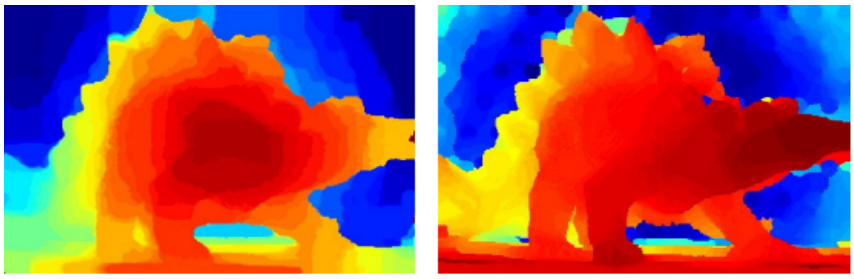


Figure 3.8: Qualitative comparison of algorithms using an image captured with a Raytrix camera [47]

- **Customized camera** - Jeon *et. al.* [47] transformed a conventional camera into a light field camera by inserting an array of lenses into the sensor of the machine in order to try to use their algorithm to estimate the scene's depth. The results were not the best because the captured images had too much noise which negatively affected the quality of the generated depth map. In order to obtain better results, it was necessary to apply the distortion correction algorithm also proposed in the work of Jeon *et al.* [47]. Figure

3.9 shows the difference between the estimated depth map. Without the application of the distortion correction, the depth map presents some problems when compared with the depth map generated with the corrected distortion.



Without distortion correction

With distortion correction

Figure 3.9: Comparison of depth maps with and without distortion correction of an image captured from a conventional camera [47].

3.2 Summary

In this chapter the steps of the algorithm developed by Jeon *et al.*[47] were described. The cost volume calculation is the first step analyzed and, through the use of SAD and GRAD can detect correspondences between the central viewpoint image and the other images. SAD performs the comparison between traditional RGB images while GRAD uses vertical and horizontal gradient images to make the comparison. At the end of the process, for all viewpoint images of a given label, their SAD and GRAD values are summed and grouped into a cost volume array. The next step is the application of the weighted median filter to remove the outliers and to soften the image. For this purpose, the values of the neighboring pixels are used to estimate the most appropriate value in a particular pixel of interest. In order to propagate and correct the disparity using the values assigned to the neighborhood, the graph cuts were used. Finally, an iterative refinement mechanism is applied with a new cost volume calculation and a weighted median filter process to remove the outliers and smooth the images. This algorithm was tested for both synthetic and real images, and the results presented were significantly better than those achieved by the competitors with which they were compared.

The algorithm presented in this Chapter serves as a starting point for the four solutions developed within the scope of this dissertation. The modifications made to the proposed algorithms are analyzed and described in Chapter 4.

Chapter 4

Proposed Algorithms

The depth estimates provided by the algorithm studied in the previous Chapter [47] are reliable. However, the algorithm is a time-consuming process. This dissertation presents some improvements made to the original algorithm to reduce its processing time without a substantial loss of accuracy.

In a first phase, the original algorithm presented by Jeon *et al.* and developed in Matlab, was implemented in C++ using the computer vision library OpenCV. C++ and OpenCV were used because they are more efficient and the computing time is lower than in Matlab [62]. The flexibility to choose the library to be used to run parallel code was another factor that influenced the choice of OpenCV [63].This solution will be used as a starting point for the methodologies that will be analyzed throughout this chapter.

4.1 Parallelism and multithreading

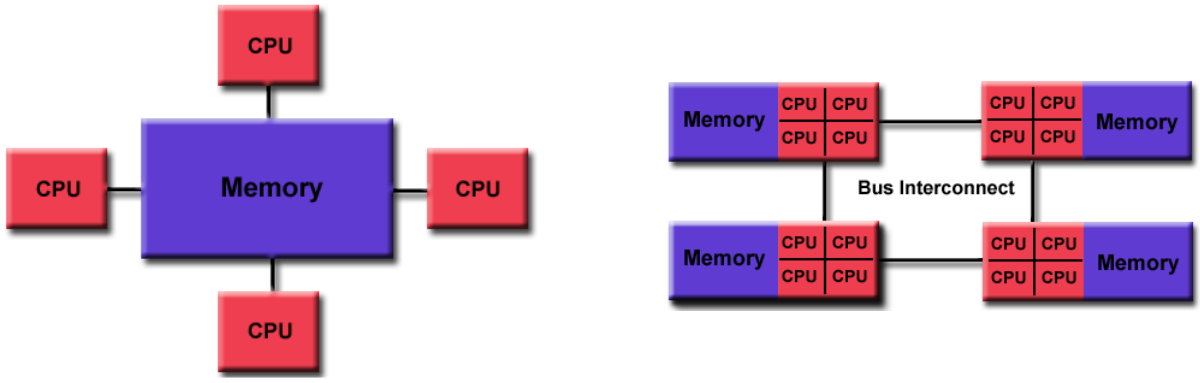
In order to increase the performance of the original algorithm, a parallel programming model approach was followed. This methodology uses the architecture of computational parallelization allowing to develop efficient solutions by taking advantage of the use of multiple computer resources simultaneously. Generally, software is developed with a serial computing approach, i.e., the set of specific instructions for its correct operation are executed one at a time, sequentially and only on a single processor. However, with the evolution observed at the computational level, parallel programming has arisen. Parallel programming allows to take advantage of multiple processors and thus perform tasks simultaneously. The set of instructions to be executed in each of the available processing units is extracted based on the division of the problem that

it proposes to solve. This approach requires a high level of coordination between processing units. Both central processing unit (CPU) and graphical processing unit (GPU) can be used in the parallel programming model.

4.1.1 Parallel Computer Memory Architectures

For the software to be executed using the parallel computing model, communication between the processing units through which the tasks are distributed is crucial. To occur the communication it is necessary to make memory accesses that can be classified as:

- **Shared Memory** - Memory is shared among all processing units, and every time a change happens, other processors can instantly see what changes were made. According to memory access, the classification of shared memory systems may be Uniform Memory Access (UMA), as seen in Figure 4.1(a), or Non-Uniform Memory Access (NUMA), as seen in Figure 4.1(b). While in UMA the processors share the same memory (although they may have different caches), in a NUMA architecture system each processor has its memory being able to access the memory of other processing units through a bus that interconnects the processors. The memory access time of the processors is not the same for all.



(a) (b)

Figure 4.1: Shared memory architecture with Uniform Memory Access (a) and Non Uniform Memory Access (b) [64].

- **Distributed Memory** - Each of the processors has its memory and are entirely independent of each other. If it is necessary to get or change data from another processing unit or ensure the synchronization of tasks, it must be done explicitly (see Figure 4.2).

- **Hybrid Distributed Shared Memory** - Hybrid memory is becoming increasingly used for computationally demanding processing. This method assumes the use of several com-

puters where the memory of the processing units in each of them is shared. However, if it is necessary to access the memory of a different machine, there must be explicit communication, such as in distributed memory systems (see Figure 4.3).

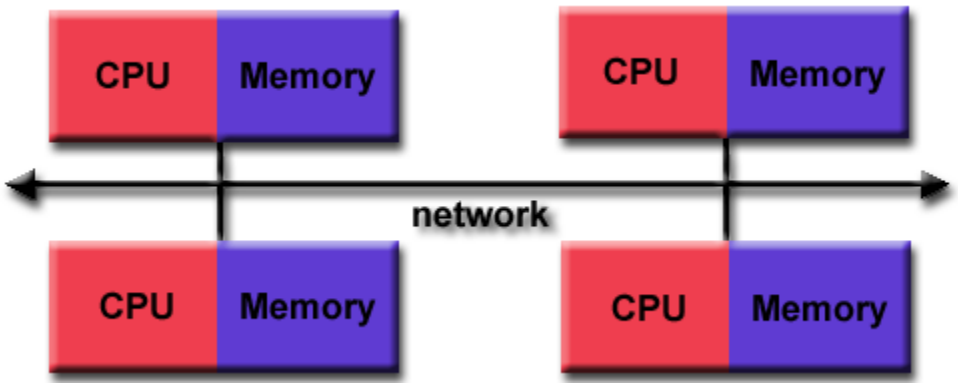


Figure 4.2: Distributed Memory Architecture [64].

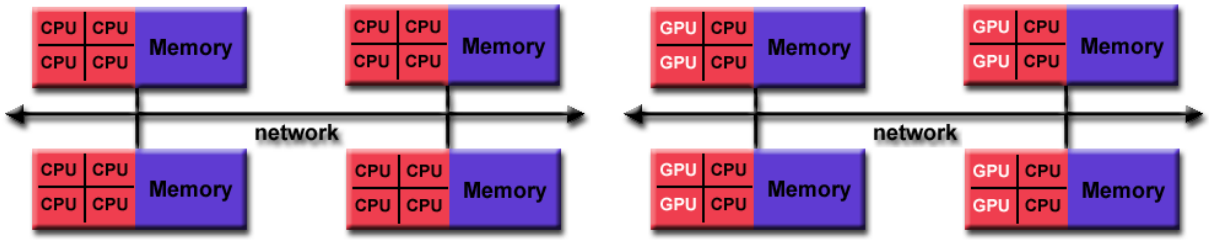


Figure 4.3: Hybrid Memory Architecture [64]

4.1.2 Parallel Programming Models

As with the architecture of memory access, parallel programming models can also be divided into different categories. The choice of model to use depends on the type of resources available. Parallel programming models may fall into one of the following categories:

- **Shared Memory without using Threads** - In this model, the processes use shared memory, and each one of them can perform read and write operations asynchronously (see Figure 4.4). It is the most accessible parallelism mechanism to implement since it does not require any modification in the code. Sometimes two or more processes may be blocked waiting for each other. This situation is called a deadlock and can be prevented using tools such as semaphores, which serve to control access to a shared resource.

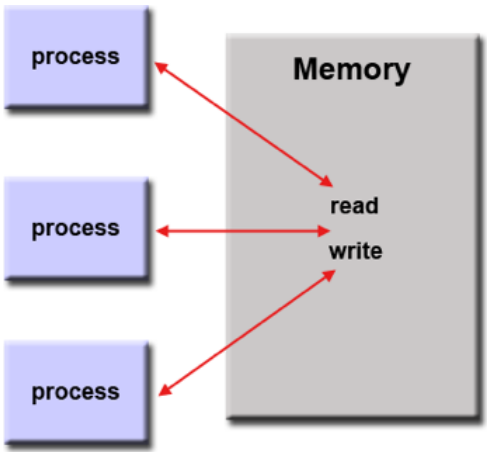


Figure 4.4: Shared Memory Model without using Threads where each process has direct read and write access to the global memory. [64]

- **Shared Memory with Threads** - With the threading model, the main thread executed by the operating system is responsible for creating new subthreads that can run in parallel and asynchronously (see Figure 4.5). The subthreads and the main thread can communicate through shared memory. However, each subthread also has data stored locally for faster access. Unlike the traditional shared memory model, the implementation process has to be explicitly made by the programmer through the libraries made available for this purpose.

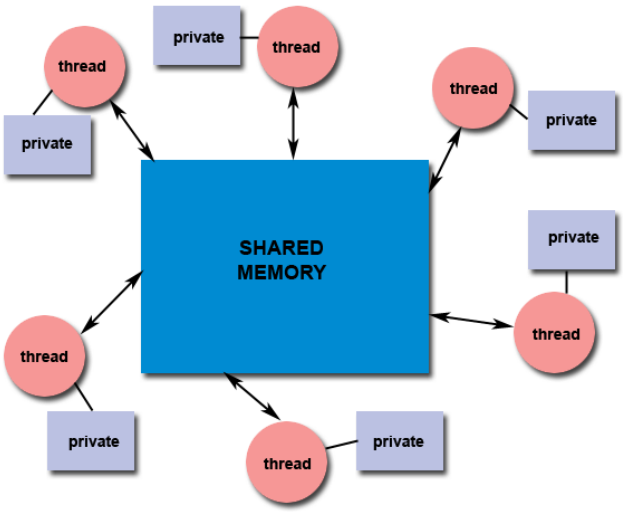


Figure 4.5: Shared Memory Model with Threads. The main process runs and places all the relevant data in shared memory. Each of the subthreads accesses this memory and loads locally the information it will need to perform a given task. [64]

- **Distributed Memory with Message Passing** - As in the distributed memory architecture, the distributed memory parallel programming model makes use of the independent

memory on each of the processing units. For the processing units communication, it is necessary to use the exchange of messages. This model is more challenging to implement because it requires high coordination between processors that could even be in different physical machines (see Figure 4.6).

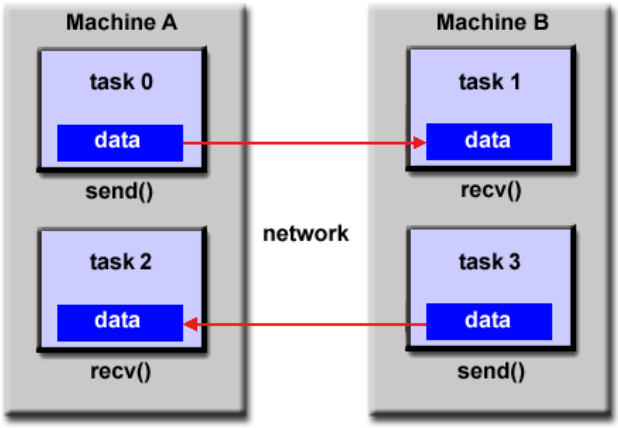


Figure 4.6: Distributed Memory model where each task uses its local memory to perform a certain processing. If there is a need for communication, there must be message exchanges, whether the processing units are on the same device or on different machines. [64]

- **Data Parallel Model** - Operations performed in parallel are executed on a given set of data that are usually represented in the form of an array. The data set is divided into partitions which are distributed by the available processes for the code execution (see Figure 4.7). This model can be adopted in both shared memory and distributed memory architecture. While in shared memory the data set is placed in global memory and accessed freely by all tasks, in the distributed memory architecture the data set is divided and stored in different memory spaces.

- **Hybrid Model** - The hybrid parallelism model combines more than one of the previously discussed methods. The combination of distributed memory with the thread-based model is one of the most commonly used solutions. Some of the tasks are executed by access only the local memory; however, there is always the possibility of accessing memory data from other processing units whether they are on the same or different physical machine. The hybrid model between the shared and the distributed memory is also the most complex solution to build since operations have to be carefully implemented so that there are no memory access issues. (Figure 4.8).

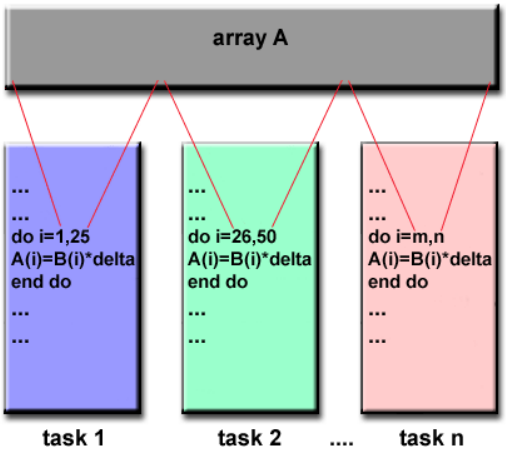


Figure 4.7: Example of a model in which the array is divided into several segments which in turn are assigned to the threads provided by the system. Each thread is responsible for running a particular piece of code in parallel. [64]

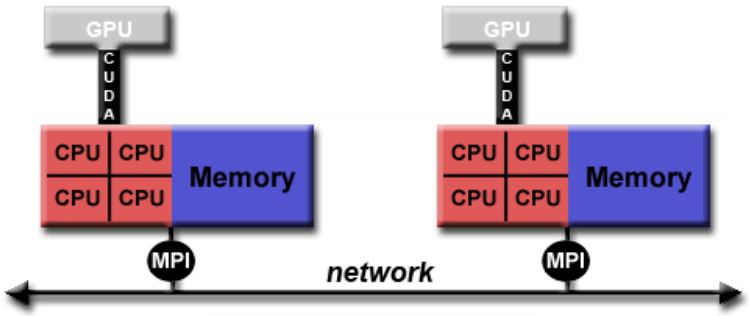


Figure 4.8: Model of parallelism using the hybrid scheme between distributed and shared memory. The CPU sends the information to the GPU for faster processing. When the GPU has completed the task, the information is sent back to the CPU. The distributed memory is used for communication between the different machines available to perform the calculations. [64]

4.2 Parallel processing using the CPU

One of the existing acceleration mechanisms for computing time to be reduced uses the CPU as a computational unit. This solution is the most popular, because unlike acceleration through the use of graphics cards, it does not require any specific hardware as all computers have a CPU to function properly.

The parallel programming model to adopt may be any of those mentioned in Section 4.1.2; however, a shared memory approach with threads have been used for the proposed algorithms in this dissertation. This type of implementation uses the CPU cores from one computer to perform parallel processing and needs to be explicitly programmed through a parallel acceleration library. For this dissertation, the proposed algorithms use the library developed by Intel®

called Threading Building Blocks (Intel® TBB). This library allows the use of a multi-core architecture without programmers needing to manage and maintain threads, making the solution as efficiently as possible. The TBB algorithms are fully compatible with modern processors and can be easily scaled to a custom number of cores. By default, the TBB library uses the maximum number of cores available in the machine.

4.2.1 Intel® TBB working principles

Although the parallel programming model to be used fits into the shared memory based on threads, TBB ends up using *tasks*, a higher abstraction level with a better performance according to Intel®. Tasks are performed through a scheduler supplied by the library, allowing to distribute the tasks by the available cores. Unlike the operating mode commonly used by the parallel libraries, where there is a global queue for the tasks, TBB uses a local pool in each of the threads with tasks to run.

The TBB scheduler executes jobs through the evaluation of a graph where each of the nodes corresponds to a task. Each task has associated a counter called *refcount* that stores the number of child tasks + 1 to be executed, as can be seen in Figure 4.9.

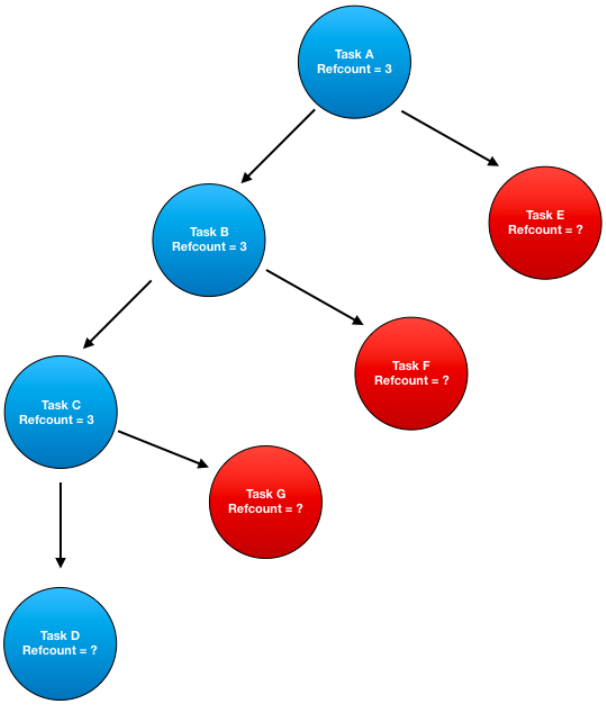


Figure 4.9: Example of a task graph. In blue are represented the tasks whose execution has already begun and in red are identified the tasks that await execution. Each of the tasks has a *refcount* that symbolizes the number of childs + 1 of each node.

Figure 4.9 shows in blue tasks that have already begun to be executed by the available threads, while in red pending execution tasks (E, F, G). The *refcount* value is unknown in case of task D because it did not spawn any child tasks and, as such, the counter has not yet been initialized.

The order of execution of the tasks is determined by the scheduler and aims to minimize access to memory and communication between threads without losing the ability to run in parallel. In order to achieve a solution as efficient as possible, there must be coordination between the two execution mechanisms used by the scheduler:

- **Depth-First** - It is applied for the execution of the tasks that are in the queue of the thread and recommended for the sequential execution of the solution. It does not need communication between threads, and it does not present problems in memory consumption.
- **Breadth-First** - Although the memory consumption is considerably high when this mechanism is used, its application guarantees the highest performance in parallel and the possibility to communicate between the queues of the threads. Its use is recommended to pick a task from a queue of a randomly chosen thread in order to keep the processor busy.

In summary, TBB uses the depth-first approach to execute tasks on the threads whereas uses breadth-first strategy to keep a thread busy by stealing a task from another thread's queue. While breath-first steals the oldest task from a queue of a random thread, depth-first executes the most recent spawn task because it is cached and, in theory, will be faster than the old tasks. Figure 4.10 depicts an overview of TBB behaviors in multiple situations. The oldest task, at the top of the thread's two local queue, is stolen to be executed by thread one whose queue is empty. In thread zero, it is also possible to verify that the newly created task goes to the bottom of the queue and in thread three is mirrored the normal behavior in which the most recent is the first one to be executed because it is in the cache and can be accessed faster by the CPU.

Knowing the two execution mechanisms used by the scheduler and under what circumstances they are applied, it is equally important to understand the preferred criteria for choosing the task to be performed. Thus, the choice of the task to be performed by a given thread is provided by the first fulfilled condition from the following list:

1. The task returned from the executed child task. If no task is returned, this criterion does not apply.

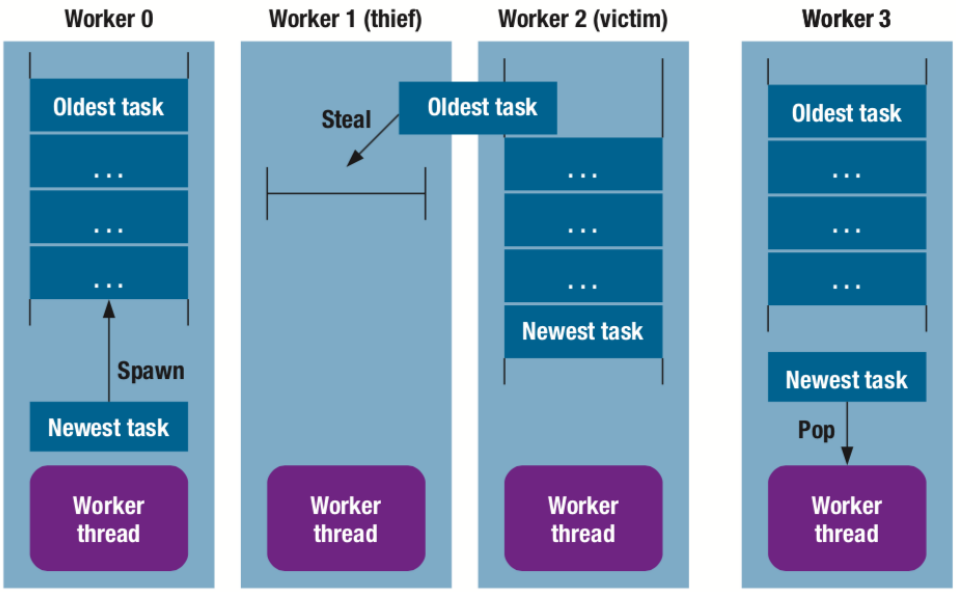


Figure 4.10: Several behaviors that each of the threads can assume when using TBB as a parallelization method [65].

2. The task most recently spawned to the thread queue. It does not apply if there are no queued tasks. The depth-first execution is used.

3. Pick a task from another randomly chosen thread's queue. If the chosen queue is empty, the criterion is applied again until it finds some thread with at least one task in the queue. The breadth-first execution is used.

4.2.2 Cycle Parallelization using TBB

Although the TBB provides a vast number of resources, parallelization of the algorithms proposed in this dissertation was performed using the `parallel_for` method, which is analyzed in more detail throughout this Section.

Let us assume that the algorithm intends to execute an iteration loop that applies a function to each element of an array. The iteration space is of type `size_t`, and it varies between 0 and `n`-1. The sequential code for this situation can be seen in Listing 4.1.

Listing 4.1: Example of a sequential iteration.

```
void SerialApplyFoo( float a[], size_t n ) {
    for( size_t i=0; i<n; ++i )
        Foo(a[i]);
}
```

The solution presented in Listing 4.1 works appropriately; however, the code is executed sequentially, i.e., it only uses one thread for processing. Assuming that the iterations are independent, i.e. that the code executed in each iteration does not depend on the result obtained in another iteration, it is possible to execute the same function by using the `parallel_for` template and make use of the threads provided by the system in order to decrease the computing time. `Parallel_for` is a generic solution offered by TBB in order to run a parallel cycle. The `Parallel_for` method accepts two input parameters:

- `range` - Iteration space for which the cycle is executed. The iteration space is divided into fragments to be distributed by the available threads for parallel processing.
- `body` - Function object where the `operator()` method processes a fragment of the problem.

To use `parallel_for` it is necessary to create an object compatible with the template. In the Listing 4.2 it is possible to see an example of a class prepared for the use of parallelism.

Listing 4.2: A class prepared to run with `parallel_for`.

```
#include "tbb/blocked_range.h"
class ApplyFoo {
    float *const my_a;
public:
    void operator()( const blocked_range<size_t>& r ) const {
        float *a = my_a;
        for( size_t i=r.begin(); i!=r.end(); ++i )
            Foo(a[i]);
        }
        ApplyFoo( float a[] ) :
        my_a(a)
    {}
};
```

Since each of the iterations is independent, it is necessary to create a class constructor that remembers the local variables declared outside the sequentially implemented cycle but used inside it. This constructor is the `body` argument and is copied to all the fragments in which the iteration space has been divided. Because it is a copy of the constructor, the values of the

variables should not be modified inside the `operator()` method, because there is a risk of not being reflected in the other threads used for processing. It is recommended to pass a pointer to the variables that are intended to be modified within the cycle as can be seen in Listing 4.2. It is possible to see that the `my_a` points to the first index position from the array passed as an argument in the `ApplyFoo` constructor method. By passing a pointer, it is ensured that changing the value stored in memory during an iteration has repercussion on the remaining iterations.

The `operator()` function is responsible for running the parallel code accepting as argument a `blocked_range <size_t>` corresponding to the space of iterations executed, in this case between 0 and `n`-1. In Listing 4.2, the `Foo` method is applied to each member of the array and is already running in parallel inside the `operator()` function.

After the construction of a class compatible with parallelization, the execution of the `parallel_for` method it is necessary. Taking Listing 4.3 as an example, it is possible to see that the template sends as input parameters `blocked_range<size_t>(0,n,grain size)` for range and `ApplyFoo(a)` for body. The range parameter has three arguments, the first two correspond to the interval between which the parallel loop is executed, and the third is the `grainsize` argument that represents the number of iterations to execute in a single processor. Its use is optional, however, if the `grainsize` is not defined, the value of `grainsize` = 1 is assumed, meaning that each performed iteration corresponds to a fragment. The value of the grain size can also be heuristically determined if the `auto_partitioner` is added as an additional argument in the function `blocked_range<size_t>(0,n), auto_partitioner()`. The `grainsize` can also be manually specified and must be chosen independently of the number of processors and iterations. Generally, this number manages to be discovered based on trial error. To find the ideal value for `grainsize` the best solution is:

1. Set a high `grainsize` value such as 10000;
2. Run the algorithm using only one processor;
3. Cutting the initial value down to half and check the slowdown percentage for running the algorithm.

A runtime deceleration between 5% and 10% using only one processor is the ideal value for the `grainsize` variable [66]. In case of doubt, it is better to keep a high value because although it reduces the parallelism, it does not cause overhead problems that may occur when the task's division is too small.

Listing 4.3: Example of using `parallel_for`

```
#include "tbb/parallel_for.h"
void ParallelApplyFoo( float a[], size_t n ) {
    parallel_for(blocked_range<size_t>(0,n,grainSize),
    ApplyFoo(a) );
}
```

The algorithms developed in the context of this dissertation were tested with the `grainsize` being automatically calculated.

4.3 Developed Algorithms using Parallelism

In this section, the main modifications performed to the implemented algorithms in this dissertation are described. The proposed algorithms are: 1. Parallelized Original Algorithm (POA); 2. Label Interpolation Algorithm (LIA); 3. Grayscale Image Algorithm (GIA); 4. Grayscale Image and Label Interpolation Algorithm (GILIA).

Their implementation aims to reduce processing time without affecting too much the accuracy of the estimated depth map when compared to the algorithm originally implemented by Jeon *et. al.* [47].

4.3.1 Parallelized Original Algorithm (POA)

The development of the original algorithm with a parallel programming model was the first solution that was chosen to be implemented as it could be used as the starting point for the development of the other proposed algorithms.

In order to develop an algorithm using parallel processing, it is important to analyze the hotspots of the solution, i.e., to verify where the application is taking the most time to execute and where parallelization can bring a real advantage. It is equally important to separate the code into tasks that can be executed without any dependency on each other.

The algorithm can be divided into four distinct phases, whose performance as well as its parallelization capacity are analyzed in the following subsections:

- Phase Shift Theorem and Cost Volume;
- Weighted Median Filter;

- Multi-label Optimization using Graph Cuts;
- Iterative Refinement.

For analysis purposes, the original algorithm [47] implemented in C++ and OpenCV without any recourse to acceleration mechanisms was used. At this stage, the algorithm was only tested on a set of 7x7 viewpoint images with 328x328 resolution each to verify the critical points of the source code. The results obtained are depicted in Table 4.1. It can be seen that the most time-consuming sections are cost volume and iterative refinement because of a large number of iterations performed. Although WMF and Graph Cuts are the fastest sections, they are also inefficient as far as processing time is concerned, because it must be considered that the test used a limited number of low-resolution viewpoint images.

Cost Volume (s)	**WMF (s)**	**Graph Cuts (s)**	**Iterative Refinement (s)**
211	60	44	172

Table 4.1: Computation time analysis of the original algorithm [47] using C++ and OpenCV without applying any acceleration mechanisms for a set of 7x7 viewpoint images with a resolution of 328x328 pixels each.

Phase Shift Theorem and Cost Volume Hotspots

According to the results depicted in table 4.1, the Cost Volume phase takes the longest time to process. The high computational time is mainly due to the cycle that shifts all the viewpoint images for each label not only in the original image but also for the directional gradient matrices. The subpixel shift method consists in multiply the Fourier transform and the exponential matrix to then calculate the inverse of the Fourier transform and put the image back into the spatial domain. The comparison between each of the shifted viewpoint images and the reference image to calculate the cost volume is also a computationally time-consuming process.

Since calculating the shift with subpixel precision for each of the viewpoint images does not depend on other viewpoints, it is plausible to adopt the parallelization. It is also possible to calculate the cost volume in parallel, as long as it is provided the label index for which the sum of GRAD and SAD is made.

In Listing 4.4 the `body` of the `parallel_for` method can be seen. This example is slightly different from the one previously discussed because an OpenCV binding is used for the TBB `parallel_for` method.

Listing 4.4: Snippet of the body argument for the Cost Volume calculation

```
void ParallelCostVolume::operator()(const cv::Range& range) const {
    ...
        for (int i=range.start;i<range.end;++i) {
            ...
            auxiliarFunctions::calculateFourierTransform(...);

               for (int j=1;j<numLabel;j=j++)
            auxiliarFunctions::calculateLabels(...);
    }
}
```

For the Listing 4.4, `i` corresponds to the iteration space which varies between 0 and the number of viewpoint images - 1 because the cost volume is not calculated for the central image. Within the main cycle are calculated the Fourier transform for the viewpoint image to be iterated. Images in the frequency domain are passed as an argument for calculating the cost volume that is performed inside the `auxiliarFunctions::calculateLabels(...)` function. The `j` corresponds to the label index for which the calculation is made. In this case, the fragmentation of the problem is done by the number of viewpoint images for which it is necessary to calculate the cost volume.

Weighted Median Filter Hotspots

For the use of the weighted median filter, it is necessary to calculate the inverted variance matrix for the image chosen as the reference, which in this case is the central viewpoint image. The box filter is one of the operations performed for the calculation of the inverse variance matrix. The box filter is applied in the reference image on each of the RGB channels individually. Therefore, since none of the channels is directly dependent on the other, parallelization can be used for this operation.

After the preliminary calculations employed to the central viewpoint image, it is necessary to apply the weighted median filter to the cost volume array. The cost volume matrices are independent of each other, and as such, the weighted median filter can be applied in parallel. Listing 4.5 shows the use of the weighted median filter implemented in parallel where the iteration space is between 0 and the size of cost volume array.

Listing 4.5: Implementation of `ParallelCostAgg` method

```
void ParallelCostAgg::operator()(const cv::Range& range) const {
     for(int i = range.start; i < range.end; ++i) {
            gfo.guidedFilterColorRunFilter(volumeCostMatrix.at(i),
            costAgg.at(i)); }
}
```

Multi-label Optimization using Graph Cuts Hotspots

The use of graph cuts to optimize the disparities according to the neighboring cost volumes is also a computationally demanding process as can be seen in Table 4.1. Although it is the phase whose processing has a shorter time, it must be considered that this preliminary analysis was tested with a small set of low-resolution viewpoint images. For the energy minimization process, the GCOptimization library developed by Olga Veksler and Andrew Delong was chosen [67–69]. With the use of this algorithm for the labels optimization, it is not possible to implement the graph cuts in parallel. However, before applying the energy minimization algorithm, an operation is performed with the goal to map the values for each pixel to a specific partition. This function can thus be used in a parallel programming mechanism since the values of the matrices are entirely independent of each other. Listing 4.6 shows the pseudo-code to assign the values of the input matrix to a given partition. The iteration space varies between 0 and the matrix size.

Listing 4.6: Method that allows to map the matrix to partition values.

```
void ParallelQuantiz::operator()(const cv::Range &range) const {
    for (int i = range.start; i<range.end;++i) {
        if the pixel i from the input matrix fits into a partition then
                output matrix[i] = partition value;
                goto indexFound;
    }
    indexFound:;
}
```

Iterative Refinement Hotspots

From the depth map estimation calculated by the graph cuts, the iterative refinement is applied as the final step of the algorithm. This calculation is also computationally quite expensive, as can be seen in table 4.1, where the time elapsed was in the order of 172 seconds.

The refinement requires the conclusion from the previous iteration so that it can advance to the next one. For this reason, it is not possible to parallelize this cycle; however, operations that occur within the cycle can use parallelism. The functions to be executed within the loop and which can be parallelized are:

- **Cost Volume Calculation** - The first phase of iterative refinement calculates a new cost volume array for a given depth map. The calculation of each cost volume matrix is independent of each other making it possible to use parallelism. The iterative space varies between 0 and the size of the cost volume array that is defined by the maximum value present in the input depth map.

- **Weighted Median Filter applied to the Cost Volume array** - The weighted median filter is applied to the cost volume array. As discussed in the subsection 4.3.1, also in this circumstance one can resort to the use of parallelism mechanisms since the matrices are independent of each other for this calculation. The iterative space varies between 0 and the size of the cost volume array.

- **Minimum Cost Selection** - The next step in iterative refinement aims to find the smallest value for each pixel among the set of cost volume matrices. The parallel programming model can also be used in this situation, by traversing for each of the pixels all the matrices present in the cost volume in order to identify the lowest value for each one. At the end of the processing, a matrix with the cost volume indices having the lowest value for each pixel is obtained. The iterative space varies between 0 and the total number of pixels in each matrix of the cost volume array.

- **Subpixel Refinement** - The indices map is used to calculate the depth estimate for each pixel using the values of the neighboring cost volumes. Parallelization can be used to perform this calculation. The iterative space varies between 0 and the total number of pixels in each matrix of the cost volume array. Listing 4.7 shows the `body` argument for calculating the refined value for each of the pixels. In all iterations, a comparison

is made between the value to be iterated and the indices matrix to verify if there is any correspondence. If so, the positions of the pixels containing the value are extracted and a calculation is performed using not only the cost volume matrix at position `i` but also the neighboring matrices. Finally, the optimized depth map is filled with the result of the calculation for each pixel.

Listing 4.7: Sub-pixel refinement method.

```
void ParallelSubPixelRefinement::operator()(
const cv::Range &range) const {
    ...
    for (int i = range.start; i<range.end; ++i) {
        binaryMat.setTo(1,idMap == disps.at(i));
        findNonZero(binaryMat, nonZeroCoordinates);

        if (nonZeroCoordinates.total() != 0) {
            costslice_m = costVolume.at(i-1);
            costslice_c = costVolume.at(i);
            costslice_p = costVolume.at(i+1);

            ...
            //Some calculations using
            the values from the
             cost volume matrices.

            for (int k = 0; k < nonZeroCoordinates.total();k++) {
                ...
                //Fill the refined depth matrix
       }
    }
}
```

4.3.2 Label Interpolation Algorithm (LIA)

As it was possible to analyze through the performed test whose results are presented in table 4.1, the most consuming section is the cost volume processing due to the need to calculate the

subpixel shift for all the viewpoint images in each label.

For each of the labels, the shifted viewpoint images are compared to the central image in an absolute way and with directional gradients in order to find correspondences. The computed results are summed, and a cost volume matrix is obtained. In the end, the result is a cost volume array with the size equal to the number of labels.

In order to optimize the performance of the solution and trying to ensure that the generated depth map maintains an acceptable level of quality, a label interpolation algorithm is herein proposed. In the proposed algorithm, some of the labels are not processed in the iterations applied to each of the viewpoint images. With this modification, some matrices from the cost volume array are initialized to zero.

The LIA solution introduced a new input parameter called `interpolationValue`. The value assigned to this parameter corresponds to the number of jumps for the calculation of each cost volume matrix. The minimum value is equal to one, meaning that for this situation there is no interpolation and the cost volume is calculated for all the labels. Skipping the labels is reflected in how the iteration is performed. In Listing 4.4, instead of `j=j++`, the next iteration is now calculated with `j=j+interpolationValue`. The initial and final cost volume matrices are always calculated, to ensure that the intermediate matrices where the interpolation is applied have references to use. After calculating the cost volume array, the weighted median filter is employed to remove the outliers and smooth the matrices. The filter is only used in the cost volume matrices that are filled, skipping those whose values are equal to zero. When the smoothing of the cost volume matrices is complete, the zero matrices are calculated using the following equation:

$$C(x,l) = \alpha C(x,l-1) + (1-\alpha)C(x,l+n), \tag{4.1}$$

where the interpolation of the null matrix is found by adding the previous matrix $C(x,l-1)$ with α weight to the next nonzero matrix $C(x,l+n)$ with the weight being equal to $(1-\alpha)$. The value of n corresponds to the difference between the index of the next matrix filled in the array and the index of the current matrix l. The α value can be calculated using the following Equation:

$$\alpha = \frac{n}{n+1}. \tag{4.2}$$

The weight of the matrix in equation 4.1 whose index is $l-1$, has a trend to assume greater importance than the matrix with index $l+n$. This situation occurs because it is at a shorter distance from the null matrix that is being subjected to the interpolation. The minimum value that can be assigned to alpha is 0.5, meaning that only when $n = 1$ the weight of the matrix with the index $l+n$ is equal to the matrix of index $l-1$. The degree of influence for the matrix with the index $l+n$ decreases the further away the next filled matrix is in the cost volume array.

When the interpolated matrices are filled, the calculation to be performed is the same as the original algorithm.

4.3.3 Grayscale Image Algorithm (GIA)

In order to begin the process of calculating the depth map, it is necessary to import the viewpoint images to the application from a specific folder on the disk. The folder is traversed through a cycle that loads the name of the files in memory. The array with the filenames is then iterated in order to import each of the images into the application. It is precisely in this process of loading the viewpoint images that this proposed algorithm begins to differentiate from the others previously presented because it loads a grayscale image instead of an RGB image. The approach of importing grayscale images over color images reduces the size of the matrices because for each of the imported viewpoint images only one channel is used instead of three. This means that each pixel is represented by a single intensity value instead of the traditional red, green and blue component values.

Taking as an example the images presented in figure 4.11 and assuming that the size of each one is 328x328 pixels, it can be said that the matrix refers to the grayscale image (4.11(b)) has a total of 107584 values (328x328x1) while the RGB image (4.11(a)) has 322752 values (328x328x3).

After uploading the grayscale viewpoint images, it will be necessary to make several changes to the original parallelization code because it was intended to be executed using the three channels of the RGB images. These changes are discussed throughout this subsection.

Cost Volume

For the cost volume calculation, the most significant differences between the Grayscale Image Algorithm and the Parallelized Original Algorithm are:

(a) RGB Image (b) Grayscale Image

Figure 4.11: Comparison between a three channel RGB image and a grayscale image with only one channel.

- In the Parallelized Original Algorithm, the calculation of the directional gradients in horizontal and vertical needs to be performed with the use of a grayscale image which implies there would have to be a conversion. Thus, with the Grayscale Image Algorithm, the directional gradients are applied directly to the image without the need to convert it. In this case, the use of grayscale images is better than the original algorithm because a computation operation is removed and the quality of the depth map is not affected by this modification.

- To pass the image to the frequency domain and then apply the subpixel shift it is necessary to calculate the Fourier transform for each of the viewpoint images and its respective directional gradients. Since the Fourier transform is implemented using the `cv::dft` method provided by OpenCV, and whose input parameter only accepts a matrix with one channel, the original solution requires to split the matrix into three parts, something that no longer happens in this situation because the input matrix is a grayscale image. The process of multiplying and applying the inverse of the Fourier transform to return the image to the spatial domain only requires to process one channel, instead of three. This modification results in a substantial decrease in processing time (see Section 5.2).

- For the calculation of the cost volume array, SAD and GRAD are used, and for the Parallelized Original Algorithm, they need to divide the RGB images into three matrices of only one channel to perform the computations. This division is not required for the

Grayscale Image Algorithm because the input matrix has only one channel and so can use both SAD and GRAD directly.

Weighted Median Filter

As discussed earlier, the weighted median mechanism uses a guided filter where the central viewpoint image is chosen as a guidance image in order to remove the outliers from the cost volume slices. The original algorithm presented by Jeon *et al.* [47] requires the color information of the pixels to be able to calculate the covariance matrix and the mean vector for the reference image. However, the approach must be modified to fit the presented algorithm. In order to the $w(\mathrm{x}, \mathrm{x'})$ of Equation 3.7 works for grayscale images, it is necessary to start calculating it as follows:

$$w(\mathrm{x}, \mathrm{x'}) = \sum_{q:(\mathrm{x},\mathrm{x'})\in\omega_q} \frac{1}{|\omega|^2}\left(1 + \frac{(I(s_\mathrm{c}, \mathrm{x'}) - \mu_q)(I(s_\mathrm{c}, \mathrm{x}) - \mu_q)}{\sigma_q^2 + \epsilon}\right), \tag{4.3}$$

where x' is a median value for the neighbors around a reference pixel x in the local window ω_q. $I(s_\mathrm{c}, \cdot))$ is the central viewpoint image used as a guide and ϵ is a smooth parameter constant. μ_q and σ_q^2 correspond to the mean and variance for the guidance image in ω_q and are calculated by the following equations:

$$\mu_q = \frac{1}{|\omega_q|}\sum_{x\in\omega_q} I(s_\mathrm{c}, \mathrm{x}) \text{ and,} \tag{4.4}$$

$$\sigma_q^2 = \frac{1}{|\omega_q|}\sum_{x\in\omega_q} I(s_\mathrm{c}, \mathrm{x}^2 - \mu_q^2. \tag{4.5}$$

Since only a channel is used to perform the processing, the efficiency for the removal of outliers and noise reduction is not as good as that of the original algorithm. In this circumstance, if the intention is to generate a depth map with the highest possible quality, it is important to use the RGB channels. However, since the aim is to find the ideal trade-off between performance and quality, this algorithm is used for the tests that are described in the next Chapter.

Graph cuts and Iterative Refinement

The graph cuts and iterative refinement steps were not subject to further modifications in the Parallelized Original Algorithm code, except when a new weighted median filter is applied in

iterative refinement phase. In this situation, the code to implement is similar to the one discussed earlier.

4.3.4 Grayscale Image and Label Interpolation Algorithm (GILIA)

For the development of this algorithm the changes addressed in subsections 4.3.2 and 4.3.3 are put together. In this way, a faster computation time is expected at the cost of a quality slightly decrease in the generated depth map (see also Chapter 5).

With the GILIA implementation, it is the cost volume phase that benefits the most from performance because the computationally longer processes are covered by the modifications made. The implementation of these changes greatly reduces its execution time. Not only because it is invoked fewer times due to the interpolation of the labels but also because it is within this method that some of the computationally more demanding functions are executed with only one color channel instead of three. The multiplication by an exponential term in the frequency domain and the consequent transformation of the image to the spatial domain using three channels is computationally quite demanding. Therefore, the calculation of the subpixel shift is one of the areas that takes more advantage in terms of computing speed when using the grayscale images.

As discussed in section 4.3.3, the weight median filter also benefits with the reduced of the computing time because of the use of grayscale images. However, a slight loss of accuracy for the estimated depth map has also been observed (see Section 5.1).

4.4 Summary

In this chapter concepts related to parallelism were introduced. Memory access architectures and parallel programming models were discussed. For the algorithms proposed in this dissertation, the parallel programming model with the use of threads was chosen. Unlike other solutions, this model requires code modifications to be made explicitly, usually through the use of a library. For this circumstance, the Intel Threading Building Block library was used.

The algorithms proposed for implementation are based on the work of Jeon *et. al* [47], and the primary objective is to find new solutions that allow a better balance of the speed-accuracy tradeoff. Therefore, the implemented algorithms were:

- Parallelized Original Algorithm (POA) - This algorithm is identical to the original imple-

mentation but with CPU acceleration through the use of the Threading Building Blocks (TBB) library. It is used as a starting point for the other three proposed algorithms;

- Label Interpolation Algorithm (LIA) - A label interpolation is implemented for the calculation of the cost volume array in which the nearest cost volume matrices are considered to do the estimation;
- Grayscale Image Algorithm (GIA) - Viewpoint images are loaded for the application only with the grayscale channel instead of the traditional RGB. This change requires that all the code only use one channel to perform the processing in the different phases;
- Grayscale Image and Label Interpolation Algorithm (GILIA) - This hybrid algorithm uses the label interpolation and the grayscale channel.

In Chapter 5 the results are analyzed for the developed algorithms. The criteria to be used are based on the computational time and the accuracy of the depth map obtained.

Chapter 5

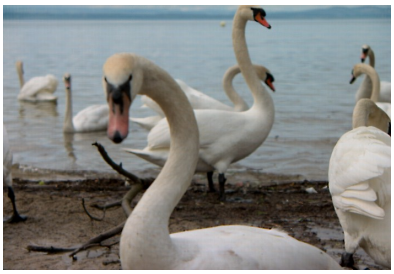
Experimental Results

Throughout this Chapter, the results obtained for the algorithms proposed in Chapter 4 are discussed. The algorithms are analyzed according to the computation time and the accuracy of the produced depth map. In order to validate the quality of the depth map, this Chapter starts with a visual assessment that compares the results with the solution originally implemented by Jeon *et al.* [47]. An objective analysis of the depth map quality is also provided. For this case, the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE) and the Structural Similarity Index Method (SSIM) were used for measuring the differences between the depth maps generated by the proposed algorithms with the ones generated by [47]. As for the processing times, the proposed algorithms were thoroughly analyzed and compared with the original algorithm developed in Matlab (MOA) and C++ (COA).

For the evaluation of the algorithms, a total of 75 labels were defined. The weight assigned to α in Equation 3.3 to determine the importance of GRAD and SAD was 0.5. The number of neighbors that influence a given pixel in the calculation of the weighted median filter was set to be an 11x11 matrix. Relatively to the graph cuts, the parameters $\lambda_1 = 2$, $\lambda_2 = 0.009$ and $\lambda_3 = 1$ for equation 3.11 were used. Finally, the iterative refinement process was executed four times so that in the end a high quality depth map is obtained. The tests were performed using a data set of images captured with the Lytro light field camera [70]. The central viewpoints of these images are depicted in Figure 5.1. The capture device has an array of 625 by 434 microlenses, each with a 15x15 pixel resolution. This means that a 15x15 2D array with 625x434 viewpoint images was obtained. Since the disparity between consecutive viewpoint images is less than 1 pixel, a value of $k = 0.01$ was used in Equation 3.2.

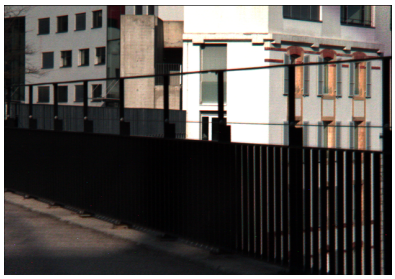
(a) Bench

(b) Sphynx



(c) Swans

(d) Bike



(e) Fence

(f) Diplodocus

(g) Friends

(h) Poppies

(i) Two Friends

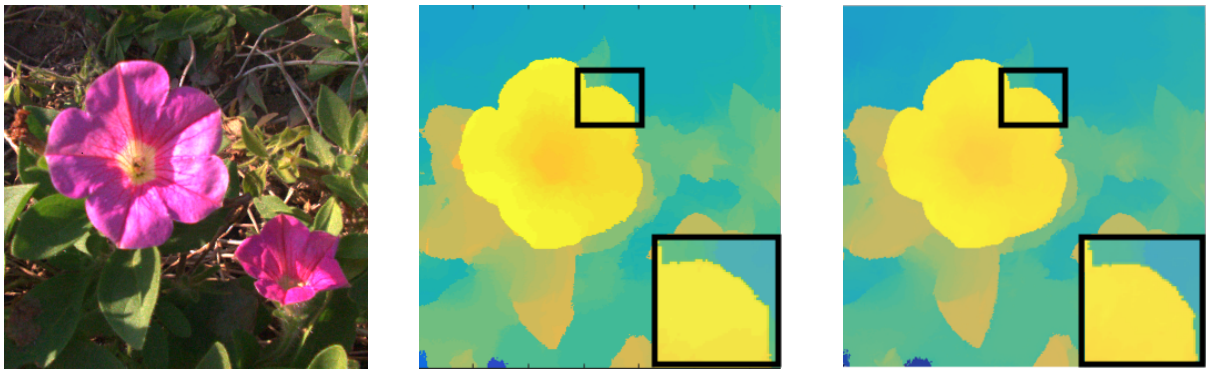
Figure 5.1: Central Viewpoint images of the data set used to evaluate the results.

5.1 Qualitative analysis

Although the primary objective of this work is to optimize the computing speed, the depth map accuracy cannot be neglected. Thus, this section analyzes the accuracy of the depth maps obtained with the developed algorithms and a comparative analysis with the results obtained in the original solution will be provided.

The first solution developed in this work was the implementation of the Parallelized Original Algorithm. This methodology, from the theoretical point of view, should generate a depth map equal to the produced by Matlab and developed by Jeon *et al.*[47]. However, numeric rounding in the computations applied to the matrices has slight differences, which leads to slightly different results. Figure 5.2 compares the Original Algorithm developed in Matlab and the Parallelized Original Algorithm.

Figure 5.2(b) and 5.2(c) display the produced depth map for the Matlab Original Algorithm



(a) Viewpoint Image (b) MOA Algorithm (c) POA Algorithm

Figure 5.2: Depth maps comparison between the Matlab Original Algorithm and the Parallelized Original Algorithm with a upsampling of an image zone. It is possible to see that in some areas there are slight differences between the maps.

and the Parallelized Original Algorithm respectively. Each of the images was zoomed in a particular area so as to perceive the in differences. The quality differences are more easily detected in the depth discontinuity zones as can be seen in the upsampled area from the Figure 5.2. This difference is related to the rounding of the decimal places hence these small discrepancies. Given that the quality difference between the two solutions is almost imperceptible, for future comparisons only depth maps calculated from the Parallelized Original Algorithm are used.

For the Label Interpolation Algorithm, the objective is to find an interpolation value in which there is a minimum quality loss of the produced depth map and whose calculation is carried out with as few labels as possible to accelerate the computing speed of the algorithm. As discussed earlier, this interpolation value represents the jump between which labels are calculated. A trial-and-error approach was used to find the best value for the interpolation. The necessary evaluations were performed and applied to the viewpoint images in order to find the highest value for the interpolation value (i) without the depth map precision being severely affected.

(a) (b) (c) (d) (e)

Figure 5.3: Depth maps comparison between the Label Interpolation Algorithm and the Parallelized Original Algorithm. (a) Viewpoint image. (b) POA Algorithm (c) LIA Algorithm with $i = 3$. (d) LIA Algorithm with $i = 5$. (e) LIA Algorithm with $i = 7$.

Figure 5.3 depicts typical results generated through the use of the Label Interpolation Algorithm for different values assigned to the label interpolation. Figures 5.3(c), 5.3(d) and 5.3(e)

show the estimate of a depth map using the LIA algorithm with $i = 3$, $i = 5$ and $i = 7$ respectively. When comparing Figure 5.3(b), which corresponds to the depth map of the scene without any interpolation, with the depth maps obtained through the interpolation mechanism, it can be seen that the one with the most approximate results is 5.3(c), which uses a value for the interpolation of $i = 3$. Although Figure 5.3(c) still omits some of the detail presented in the reference depth map, the estimate obtained is already quite satisfactory with no substantial loss of quality, especially when compared with Figures 5.3(d) and 5.3(e). Thus, given the results presented in Figure 5.3, it can be assumed that the value to be assigned for labels interpolation in which there is no high loss of quality would be $i = 3$. This value is the standard to be used for the LIA Algorithm in the comparison tests.

For the Grayscale Image Algorithm (GIA), as discussed in detail in the previous Chapter, only a single channel was used, instead of three, to estimate the depth map. Figure 5.4 compares the depth map extracted from the Parallelized Original Algorithm which uses the RGB channels and Grayscale Image Algorithm. As expected, the estimated map from GIA Algorithm has a lower accuracy when compared with the POA Algorithm. This situation is mainly due to the use of the guided filter as an edge-aware mechanism where the use of color was fundamental for efficient removal of the outliers and, consequently, for the smoothing of the image. Figure 5.4 shows a comparison between the Parallelized Original Algorithm in Figure 5.4(b) and the Grayscale Image Algorithm in Figure 5.4(c). In the swan located at the central position, it is possible to verify in Figure 5.4(c) that the edges have a higher amount of outliers than in the Figure 5.4(b) where the three color channels are used for the processing.

(a) Viewpoint Image (b) POA with RGB channels (c) GIA with a single channel

Figure 5.4: Depth maps comparison between the Parallelized Original Algorithm with three color channels (b) and the Grayscale Image Algorithm with only one channel (c).

The last algorithm to be analyzed is the Grayscale Image and Label Interpolation Algorithm (GILIA) that uses a hybrid mechanism that aggregates the interpolation of labels with the use of only a single channel to perform the depth map calculations. This solution from the accuracy

perspective is the one that presents worse results because it uses two techniques that remove relevant information from the processing.

A comparison of the results obtained for the implemented algorithms can be seen in Figure 5.5. A visual analysis shows that the POA, whose executed procedures are identical to the MOA, is the one that presents a better quality and as such, will serve as a basis of comparison to analyze the other solutions. At first glance, it is possible to affirm that the GILIA and the GIA solutions are what present a more different estimate relative to the POA solution. As for the accuracy of the results produced by the LIA, is the solution that visually seems to be the one that produces a more accurately depth map .

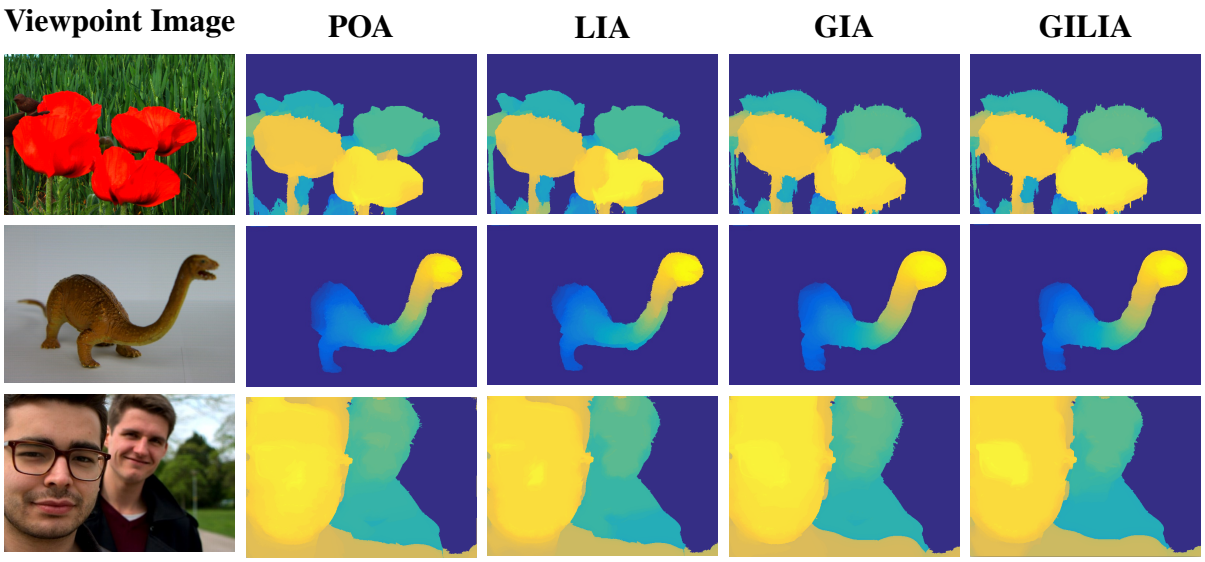


Figure 5.5: Depth map comparison between the proposed algorithms.

So far the quality of the depth estimates have only been evaluated through visual analysis; to be able to quantify the differences between the reference depth map calculated from the POA algorithm and the results obtained from LIA, GIA, and GILIA the following methods were used: 1. Mean Absolute Error (MAE); 2. Root Mean Squared Error (RMSE); 3. Structural Similarity Index Method (SSIM). In the next sections we analyze the results obtained for each of the metrics used.

5.1.1 Mean Absolute Error (MAE)

The Mean Absolute Error is one of the most commonly used metrics for comparing two depth maps. Its calculation is performed using the following equation:

$$MAE = \frac{1}{T}\sum_{p} |d_p - \hat{d}_p|, \tag{5.1}$$

where d_p and $\hat{d}_p$ represent the original and the predicted depth map respectively at pixel p. T corresponds to the total number of pixels for each of the depth maps. The obtained value for this metric does not fit into any scale because it varies according to what is being compared. In this case, as the correlation is made between two depth maps whose normalized values are in the range of 0 to 255, this will also be the range of values for the MAE. The smaller the value, the closer the proposed algorithm is to the original solution. Table 5.1 presents the results obtained for the comparison of depth estimates. The depth obtained from the POA was used as the reference and compared with the solutions produced by LIA, GIA, and GILIA. Taking into account the range of values between which the MAE can vary, the results are quite interesting. The LIA algorithm presented a very low absolute error between the estimated depth map and the reference depth map. As for the GIA and GILIA algorithms, they presented an average absolute error value also low, although higher than the one registered for the LIA. In the Fence and Friends images, the average absolute error is slightly higher for GIA and GILIA, however, given the range of values that can be assumed, the error is not significant.

Light Field Image	**LIA Algorithm**	**GIA Algorithm**	**GILIA Algorithm**
Bike	1,08	4,87	4,66
Bench	1,22	2,00	2,00
Fence	0,04	21,35	18,45
Poppies	0,55	1,48	1,52
Friends	5,61	22,12	23,14
Diplodocus	0,44	0,28	0,33

Table 5.1: Mean Absolute Error values resulting from the comparison of the Parallelized Original Algorithm with LIA, GIA and GILIA Algorithms.

5.1.2 Root Mean Squared Error (RMSE)

Root Mean Square Error is another widely used metric that can be used for measuring the accuracy of the predicted depth map with respect to its reference. For the calculation, the following formula is applied:

$$RMSE = \sqrt{\frac{1}{T}\sum_{p}(d_p - \hat{d}_p)^2}, \tag{5.2}$$

where the designations are identical to those presented for the MAE case. The MAE and the RMSE are quite similar because they both have the same range of values and, the lower the metric value, the higher the correlation between the depth maps. The difference between the two methods lies in the importance given to the detected errors. The RMSE assigns a higher weight to larger errors due to the calculation of the square before averaging. Therefore, this solution is recommended for situations where major differences are intended to be penalized. In the context of this dissertation, it is applied in the comparison between depth maps in order to accentuate the larger discrepancies. Table 5.2 shows the result obtained using the RMSE metric. Similarly to the MAE case, the Parallelized Original Algorithm estimation was used as a reference. Although the range is the same, the RMSE error is higher than the MAE due to the additional weight attributed to the comparisons whose differences are more pronounced. In most tests performed with the RMSE metric, the LIA is the one that estimates the depth most similar to the original algorithm. However, for the Fence image, it has the highest error value. If a comparative analysis is performed with the MAE for the Label Interpolation Algorithm, the Fence moves from the image with the lower MAE error to the one with the highest value in RMSE. This discrepancy between the results of the metrics means that although there is a low quantity of wrong correspondences, those that exist present a very marked difference. As for the GIA and GILIA algorithms, the results obtained are quite similar and also have a relatively low value.

Light Field Image	**LIA Algorithm**	**GIA Algorithm**	**GILIA Algorithm**
Bike	9,37	34,97	33,62
Bench	8,24	20,17	20,39
Fence	46,57	33,18	30,80
Poppies	6,44	30,52	30,62
Friends	10,15	26,93	27,76
Diplodocus	8,35	19,10	17,51

Table 5.2: Root Mean Squared Error values resulting from the comparison of the Parallelized Original Algorithm with LIA, GIA and GILIA Algorithms.

5.1.3 Structural Similarity Index Method (SSIM)

The Structural Similarity Index is widely used as an image quality metric because it is simple to compute and provides reliable quality results [71]. This mechanism operates not only for image comparison but also can be used to calculate the similarity between depth estimates as discussed in the work produced by Hossein and Peter [72]. SSIM multiplies the luminance, contrast, and structure of the image in order to calculate the similarity index [71]:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{5.3a}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{5.3b}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{5.3c}$$

$$SSIM(x,y) = [l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}, \tag{5.3d}$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. C_1, C_2 and C_3 are constants that avoid unstable SSIM values if the local mean or standard deviation are close to zero. The values assigned to these constants are usually small. To quantify depth map quality, the similarity index value will be used, which corresponds to averaging the local SSIM values calculated for each pixel in the image. The range of values that the index of similarity can assume is between -1 and 1. The smaller the value of the similarity index, the greater the difference between the compared images.

Figure 5.6 shows the SSIM maps (LIA SSIM, GIA SSIM, GILIA SSIM) generated by the comparative analysis between the depth map from POA (POA DM) and each other of the proposed solutions (LIA DM, GIA DM and GILIA DM). The SSIM maps have the colors reversed to make them easier to analyze. Therefore, the darker the color in each pixel the higher the degree of similarity between the compared images. In the opposite sense, the clearer the pixel, the less similarity exists between the two depth maps to be compared, meaning that the estimated value is incorrect if it is assumed that the solution from POA is the accurate estimation.

Analyzing the results from Figure 5.6, it is possible to verify that the LIA algorithm is the solution that most closely matches the result obtained from the Parallelized Original Algorithm. Still, it has some problems in estimating some of the backgrounds from the scene, as it is possible to see for the estimation made for the bench and the fence images. For the GIA and GILIA

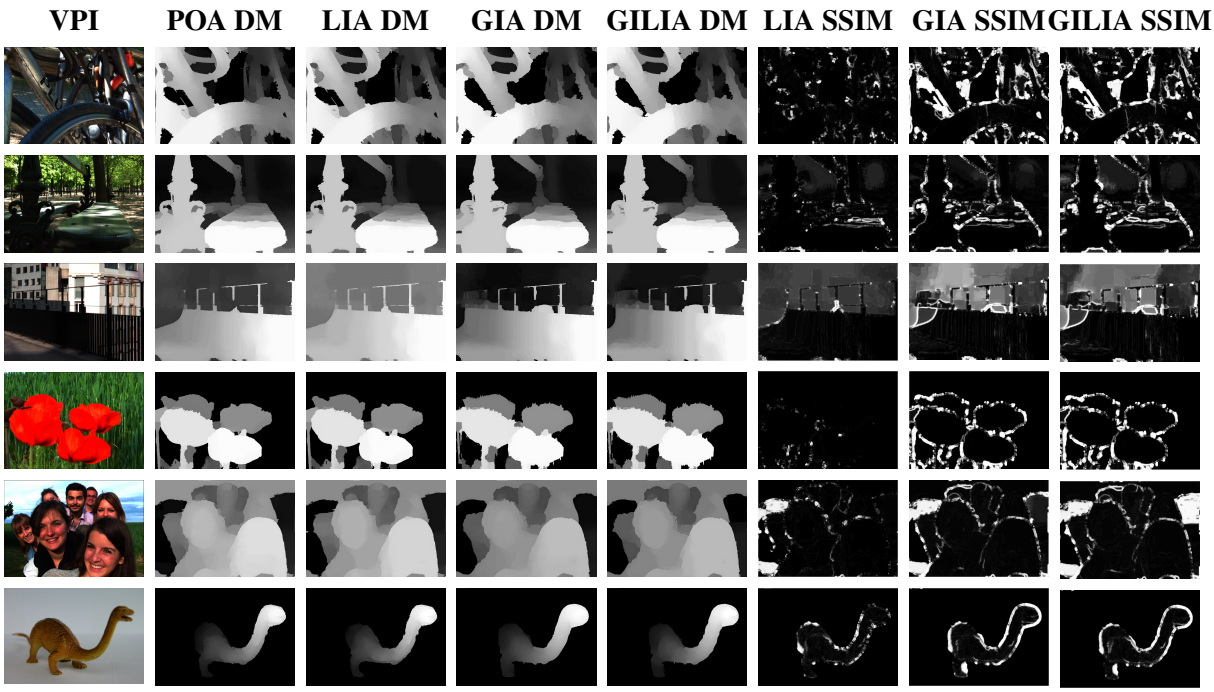


Figure 5.6: Structural similarity indexes maps with inverted colors for the LIA (LIA SSIM), GIA (GIA SSIM) and the GILIA (GILIA SSIM). The depth map from POA (POA DM) was used as reference and compared to the depth map of LIA (LIA DM), GIA (GIA DM) and GILIA (GILIA DM). The depth maps were estimated according to the central viewpoint image (VPI).

solutions, there is already a more significant difference when compared with the reference image. This loss of precision is due to the removal of the RGB channels for the calculation of the Guided Filter. With the use of the grayscale channel, it is less reliable to calculate the depth estimation especially at the edges of the images.

Light Field Image	LIA Algorithm	GIA Algorithm	GILIA Algorithm
Bike	0,95	0,82	0,82
Bench	0,94	0,90	0,89
Fence	0,87	0,76	0,80
Poppies	0,99	0,90	0,90
Friends	0,93	0,88	0,84
Diplodocus	0,97	0,94	0,95

Table 5.3: Global SSIM values resulting from the comparison of the Parallelized Original Algorithm with LIA, GIA and GILIA Algorithms.

Table 5.3 shows the overall structural similarity index for comparison between the POA and the other proposed solutions. This index corresponds to the average of the local SSIM values recorded for each of the pixels from the SSIM map. Given that the range of values that the global index can assume, it can be concluded that the decrease in quality is minimal and does

not preclude the use of any of the proposed solutions to estimate an accurate map. The LIA is the one that presents a precision closer to the results estimated by the POA. As for GIA and GILIA, they have a very similar value for the tested solutions and also estimate very accurate depth maps, although with inferior quality to the LIA algorithm.

5.2 Performance analysis (Run Time)

To evaluate computation performance, a comparative analysis was made between the four algorithms implemented with CPU parallelization (POA, LIA, GIA, GILIA) and the original algorithms implemented in Matlab (MOA) and C++ (COA). All tests were performed on a computer with a 2.9 GHz Intel i5 Quadcore CPU and 8 GB RAM. For LIA and GILIA a value of $i = 3$ for the interpolation value is used, because as mentioned in the previous section, it allows to calculate the depth map without substantial loss of quality. Any code that is executed in parallel has no limitation on the maximum number of processors to use, meaning that all available processors are used for the calculation.

For the performance tests the following approaches were considered:

- Original methodology developed in Matlab in the context of the work presented by Jeon *et al.* [47] (MOA);
- Original algorithm developed in C++ and with the help of the OpenCV library (COA);
- Solution with a thread-oriented parallel execution implemented using the Thread Building Block library from Intel (POA). This algorithm is explained in Section 4.3.1;
- Parallel execution method with a label interpolation mechanism for the cost volume calculation (LIA). The modifications made are described in Section 4.3.2;
- Use of a single channel for parallel depth map computation (GIA). For implementation details see Section 4.3.3;
- Hybrid approach that uses the label interpolation mechanism with the grayscale channel to perform the processing. (GILIA). The way the hybrid mechanism works is described in Section 4.3.4.

Figure 5.7 shows a comparative chart of the processing speed for each of the algorithms mentioned above. From the information on the chart it is possible to verify that:

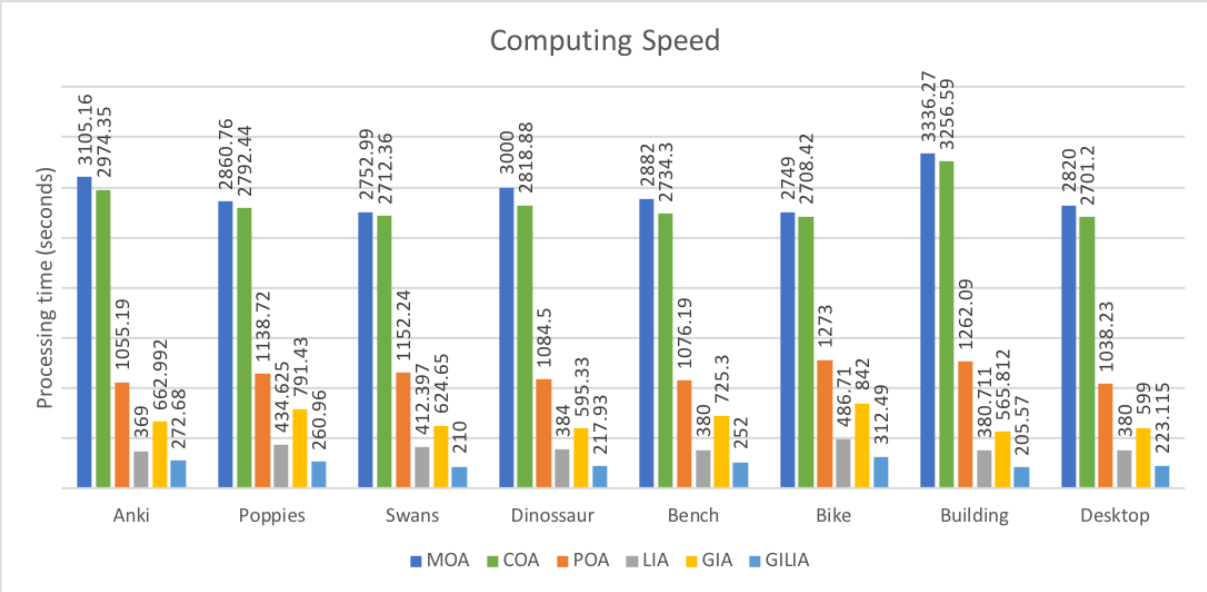


Figure 5.7: Comparison between the processing time of the algorithms. The results were analyzed for: Matlab Original Algorithm (MOA), C++ Original Algorithm (COA), Parallelized Original Algorithm (POA), Label Interpolation Algorithm (LIA), Grayscale Image Algorithm (GIA) and, the Grayscale Image and Label Interpolation Algorithm (GILIA).

- For the two algorithms without parallelization, the C++ Original Algorithm, implemented in the scope of this dissertation, is slightly faster than the Matlab Original Algorithm from Jeon *et al.* [47];
- The Parallelized Original Algorithm can be about three times faster in the calculation than the Matlab Original and C++ Original Algoritms;
- The Label Interpolation Algorithm is a more computationally efficient solution than the Grayscale Image Algorithm for processing. LIA can be about twice as fast as GIA;
- As for the Grayscale Image and Label Interpolation Algorithm, it is the one that presents better performance at the computational level.

With the results presented in this section, it can be stated that all proposed algorithms can reduce the computing time of the original algorithm. However, the purpose of this dissertation was not only to improve the processing time of the algorithm but also to ensure that the generated depth maps would not have a substantial loss of quality. This speed-accuracy tradeoff is analyzed in the next section.

5.3 Summary

Based on the information provided by this chapter, it can be stated that the POA substantially reduces the processing time without compromising the accuracy from the obtained depth map. POA can be about three times faster compared to MOA and also the COA. However, when compared to LIA, GIA and GILIA, it shows a quite slower processing speed. As for the Label Interpolation Algorithm, it is able to calculate accurate depth maps that are quite close to those produced by the original implementation. However, in some light field images the LIA can not estimate the depth for the background correctly. Regarding the processing time, LIA algorithm is able to process the information about eight times faster than the POA and two times faster than the GIA, just getting behind GILIA. Concerning speed-accuracy tradeoff, GIA was the one that presented the worst results because the estimated depth maps by this solution are identical to those produced by GILIA and for the computational speed is the slowest proposed algorithm after POA. Although the depth maps produced by GIA and GILIA are sufficiently reliable, among the proposed algorithms are those that present less accuracy especially in the estimation of the edges of the image.

Chapter 6

Conclusions and Future Work

The work described in this dissertation was motivated by the growing interest in the use of holoscopic images for digital entertainment and other fields that require image processing such as robotics or medicine. It is in this context that there is a need to find algorithms that estimate the depth of the scene through the use of light field images. As a technology that has only recently begun to be explored, solutions for depth estimation are not optimized due to the lack of precision and, above all, to the slow computation speed of the algorithms. Thus, this dissertation has as primary objective to explore new depth estimation mechanisms with a dynamic balance between precision and computational speed.

6.1 Conclusion

In order to meet the proposed objective, the algorithm of Jeon *et al.*[47] was used as a starting point for the development of alternative methodologies. This solution was chosen because, among the existing algorithms, this method presented interesting results regarding accuracy. The first step was to implement the C++ Original Algorithm (COA) using the OpenCV image processing library. After the implementation of the algorithm, it was decided to adapt the code to work in parallel in order to use the CPU processors to reduce computation time. For the parallel implementation to become effective, it was necessary to identify the critical points that took more processing time and which of them could be executed in parallel. With the critical points identified, it became necessary to adapt the code to fit the parallel programming model. To do this, the Thread Building Block (TBB) library from Intel® was chosen. TBB was adopted due to the innovative mechanism of tasks distribution. The Parallelized Original Algorithm

(POA) was then the first solution to be implemented in order to reduce processing time and, in this case, completely preserve the depth map estimation accuracy. Based on POA, three other solutions were developed in order to estimate the depth map in the shortest time possible and trying to ensure that the precision was not strongly affected:

- **Label Interpolation Algorithm** - In order to find correspondences between the viewpoint images and the reference image, it is necessary to create a cost volume array of matrices. For each viewpoint image the array has to be iterated, and it is within each iteration cycle that the most demanding operations for this step are performed, such as subpixel shift computation and the image transformation from the frequency to the spatial domain. For the computational time reduction, a solution was implemented in which not all cost volume matrices are needed to be calculated. For these matrices that are not subject to processing, a neighborhood-based interpolation method was applied.

- **Grayscale Image Algorithm** - The original algorithm estimates the depth map based on the RGB viewpoint images because it needs them for the image smoothing and the removal of the outliers carried out by a weighted median filter using the guided filter mechanism. This dissertation proposed to use a single channel - the grayscale channel - to estimate the depth map. It was necessary to change how the guided filter was calculated and to adapt the entire solution to handle only one channel matrix.

- **Grayscale Image and Label Interpolation Algorithm** - This hybrid algorithm uses the label interpolation applied to the cost volume array, and it uses the grayscale channel instead of the RGB channels to perform the calculations.

A data set of viewpoint images extracted from a light field camera was used to verify the effectiveness of the proposed algorithms. For each of the light field image, the corresponding depth map was generated and analyzed, not only in terms of computing time, but also on its accuracy with respect to the original algorithm from Jeon *et al.* [47]. The depth map's quality was evaluated using visual analysis and the following methods: 1. Mean Absolute Error (MAE); 2. Root Mean Squared Error (RMSE); 3. Structural Similarity Index Method (SSIM).

The results showed that there was a significant processing time decrease for the developed algorithms in relation to the Matlab Original Algorithm (MOA) and the C++ Original Algorithm (COA). From the proposed algorithms, the one that presented a better performance in terms of processing time was the Gradient Image and Label Interpolation Algorithm (GILIA) which

can be about twelve times faster than the MOA and COA solutions, followed by the Label Interpolation Algorithm (LIA). Although the GILIA is the fastest in terms of processing, it generates depth maps estimations with more noise than the LIA, mainly at the edges of the images. The Grayscale Image Algorithm (GIA) presents the same problem as the GILIA due to the fact that both only use the grayscale channel for processing the depth maps. Regarding quality, the implemented solution that provides the most guarantees is the Parallelized Original Algorithm (POA), where no loss of information was observed. However, this algorithm remains computationally demanding.

Based on the results, it can be concluded that the proposed algorithms can estimate reliable depth maps. Also, the computation time was substantially reduced due to the parallelization of the code through the use of the Thread Building Block library and the new calculation mechanisms that were developed for each of the solutions. These implemented mechanisms as well as the use of the Thread Building Block library allow to provide the developed solutions with a dynamic management of the speed-accuracy trade-off and thus achieve the main purpose for this dissertation.

6.2 Future Work

As a future work, it would be essential to try to find out new mechanisms for estimating depth maps from light field images which allow the computation to be carried out with a significantly shorter computing time, without neglecting the quality of the generated solutions. Thus, an interesting starting point would be the implementation of the algorithm addressed in this dissertation by taking advantage of the parallel mechanisms on GPUs, using graphical acceleration libraries such as CUDA, a solution provided by Nvidia to make use of graphics cards processing. Although the performance of the GPU is higher than that of the CPU, there must be additional communication between the memories of the two processing units that may have some weight in the total computing time. Therefore, the recommendation would be to load the necessary variables from the CPU to the GPU and perform the most significant number of operations on the graphics processing unit before converting back into the CPU memory space in order to access the information, since it is not possible to directly access the GPU memory. In order to use the GPU effectively, it is thus essential to ensure the smallest number of possible transactions between the memory spaces of the two processing units.

Bibliography

[1] W. A. IJsselsteijn, H. de Ridder, and J. Vliegen, "Effects of stereoscopic filming parameters and display duration on the subjective assessment of eye strain," *Proc. SPIE*, vol. 3957, pp. 12–22, 2000.

[2] J.-S. Jang and B. Javidi, "Improved viewing resolution of three-dimensional integral imaging by use of nonstationary micro-optics," *Opt. Lett.*, vol. 27, pp. 324–326, mar 2002.

[3] M. Martinez-Corral, B. Javidi, R. Martinez-Cuenca, and G. Saavedra, "Integral imaging with improved depth of field by use of amplitude-modulated microlens arrays," *Appl. Opt.*, vol. 43, pp. 5806–5813, nov 2004.

[4] A. Aggoun, "3D Holoscopic Imaging Technology for Real-Time Volume Processing and Display," in *High-Quality Visual Experience SE - 18* (M. Mrak, M. Grgic, and M. Kunt, eds.), Signals and Communication Technology, pp. 411–428, Springer Berlin Heidelberg, 2010.

[5] L. Onural, "Television in 3-D: What are the prospects?," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1143–1145, 2007.

[6] E. F. M. Alazawi and P. Sciences, *Holoscopic 3D Image Depth Estimation and Segmentation Techniques*. Doc thesis, Brunel University London, 2015.

[7] R. Ng, *Digital light field photography*. Doc thesis, Stanford University, 2006.

[8] G. Polder and J. W. Hofstee, "PhenoBot - a robot system for phenotyping large tomato plants in the greenhouse using a 3D light field camera," in *2014 ASABE and CSBE/SCGAB Annual International Meeting*, (Montreal, Quebec Canada), pp. 1–7, Wageningen University & Research, 2014.

[9] N. a. Dodgson, "Autostereoscopic 3D displays," *Computer*, vol. 38, no. August, pp. 31–36, 2005.

[10] "Stereoscopy." Retrieved from `https://en.wikipedia.org/wiki/Stereoscopy`, 2018-10-31.

[11] G. Morrison, "Active 3D vs. passive 3D: What's better?," 2012.

[12] D. M. Hoffman, A. R. Girshick, and M. S. Banks, "Vergence – accommodation conflicts hinder visual performance and cause visual fatigue," *Journal of Vision*, vol. 8, pp. 1–30, 2008.

[13] N. Dodgson, "Analysis of the viewing zone of multiview autostereoscopic displays," *Electronic Imaging 2002. International Society for Optics and Photonics*, pp. 254–265, 2002.

[14] D. Gabor, "A New Microscopic Principle," *Nature*, vol. 161, no. May, pp. 777–778, 1948.

[15] R. B. A. Tanjung, X. Xu, X. Liang, S. Solanki, Y. Pan, F. Farbiz, B. Xu, and T.-C. Chong, "Digital holographic three-dimensional display of 50-Mpixel holograms using a two-axis scanning mirror device," *Optical Engineering*, vol. 49, no. February, pp. 25801–25809, 2010.

[16] M. Gupta, Q. Yin, and S. K. Nayar, "Structured Light in Sunlight," *2013 IEEE International Conference on Computer Vision*, pp. 545–552, 2013.

[17] E. Cardoso, *Performance and Quality Management of HE programmes*. Doc thesis, ICSTE-IUL, 2011.

[18] A. Sokolov, "Autostereoscopy and Integral Photography by Professor Lippmann's Method," *Moscow State University Press*, p. 123, 1911.

[19] T. Okoshi, "2 - History of Three-Dimensional Imaging Techniques," in *Three-Dimensional Imaging Techniques* (T. B. T. T.-D. I. T. Okoshi, ed.), ch. 2, pp. 8–42, Academic Press, 1976.

[20] H. E. Ives, "Optical Properties of a Lippmann Lenticulated Sheet," *J. Opt. Soc. Am.*, vol. 21, pp. 171–176, mar 1931.

[21] A. Chutjian and R. Collier, "Recording and Reconstructing Three-Dimensional Images of Computer Generated Subjects by Lippmann Integral Photography," *Applied Optics*, vol. 7, no. 1, pp. 99–103, 1968.

[22] I. J. Vilums, "Optical imaging system using lenticular tone-plate elements," 1989.

[23] Y. Kim, J.-H. Park, H. Choi, S. Jung, S.-W. Min, and B. Lee, "Viewing-angle-enhanced integral imaging system using a curved lens array," *Optics Express*, vol. 12, p. 421, feb 2004.

[24] J.-H. Park, J. Kim, Y. Kim, and B. Lee, "Resolution-enhanced three-dimension/two-dimension convertible display based on integral imaging," *Optics Express*, vol. 13, pp. 1875–1884, mar 2005.

[25] R. Martinez-Cuenca, G. Saavedra, M. Martinez-Corral, and B. Javidi, "Progress in 3-D Multiperspective Display by Integral Imaging," *Proceedings of the IEEE*, vol. 97, pp. 1067–1077, jun 2009.

[26] "Lytro Official Page." Retrieved from `http://www.lytro.com`, 2017-12-22.

[27] A. Lima, "Introducing the World's First Professional Light Field VR Camera - the Lytro Immerge," 2015.

[28] K. Sherer, "Holografica's 3-D, goggle-free display," 2008.

[29] "Holovizio Official Page." Retrieved from `http://www.holografika.com`, 2018-10-31.

[30] A. Aggoun, E. Tsekleves, M. R. Swash, D. Zarpalas, A. Dimou, P. Daras, P. Nunes, and L. D. Soares, "Immersive 3D Holoscopic Video System," *IEEE MultiMedia*, vol. 20, pp. 28–37, jan 2013.

[31] M. Miura, J. Arai, T. Mishina, M. Okui, and F. Okano, "Integral imaging system with enlarged horizontal viewing angle," *Proc. SPIE*, vol. 8384, pp. 83840O–83840O–9, 2012.

[32] H. Navarro, R. Martinez-Cuenca, G. Saavedra, M. Martinez-Corral, and B. Javidi, "3D integral imaging display by smart pseudoscopic-to-orthoscopic conversion (SPOC)," *Optics Express*, vol. 18, no. 25, pp. 25573–25583, 2010.

[33] A. Castro, Y. Frauel, and B. Javidi, "Integral imaging with large depth of field using an asymmetric phase mask," *Optics Express*, vol. 15, no. 16, pp. 10266–10273, 2007.

[34] Y. Kim, J.-H. Park, S.-W. Min, S. Jung, H. Choi, and B. Lee, "Wide-viewing-angle integral three-dimensional imaging system by curving a screen and a lens array," *Applied Optics*, vol. 44, no. 4, pp. 546–552, 2005.

[35] S. Manolache, M. McCormick, and S.-Y. Kung, "Hierarchical adaptive regularisation method for depth extraction from planar recording of 3D-integral images," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 3, (Salt Lake City, UT, USA), pp. 1433–1436, IEEE, 2001.

[36] S. Cirstea, S. Kung, M. McCormick, and A. Aggoun, "3D-Object Space Reconstruction from Planar Recorded Data of 3D-Integral Images," *The Journal of VLSI Signal Processing*, vol. 35, no. 1, pp. 5–18, 2003.

[37] S. Cirstea, A. Aggoun, and M. McCormick, "Depth extraction from 3D-integral images approached as an inverse problem," in *2008 IEEE International Symposium on Industrial Electronics*, (Cambridge, UK), pp. 798–802, IEEE, 2008.

[38] C. Wu, A. Aggoun, M. McCormick, and S. Kung, "Depth extraction from unidirectional integral image using a modified multibaseline technique," *Proceedings of SPIE*, vol. 4660, no. February, pp. 135–145, 2002.

[39] C. Wu, M. McCormick, A. Aggoun, and S. Y. Kung, "Depth map from unidirectional integral images using a disparity algorithm based on neighbourhood constraint and relaxation," in *2003 International Conference on Visual Information Engineering*, (Guildford, UK), pp. 65–68, IET, 2003.

[40] M. G. Zarpalas, D., Biperis, I., Fotiadou, E., Lyka, E., Daras, P., & Strintzis, "Depth estimation in integral images by anchoring optimization techniques.," *Journal of Display Technology*, pp. 1–6, 2011.

[41] S. Heber, T. Pock, and L. Fields, "Shape from Light Field meets Robust PCA," *Computer Vision - ECCV 2014*, no. 836630, pp. 751–767, 2014.

[42] D. Zarpalas, E. Fotiadou, I. Biperis, and P. Daras, "Anchoring Graph Cuts Towards Accurate Depth Estimation in Integral Images," *Journal of Display Technology*, vol. 8, no. 7, pp. 405–417, 2012.

[43] H. Yoo, "Depth extraction for 3D objects via windowing technique in computational integral imaging with a lenslet array," *Optics and Lasers in Engineering*, vol. 51, no. 7, pp. 912–915, 2013.

[44] Z. Yu, X. Guo, H. Ling, A. Lumsdaine, and J. Yu, "Line assisted light field triangulation and stereo matching," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2792–2799, 2013.

[45] R. Grompone Von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[46] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," *Computer Vision—ECCV 2002*, pp. 82–96, 2002.

[47] H.-G. Jeon, J. Park, G. Choe, Y. Bok, Y.-W. Tai, and I. S. Kweon, "Accurate Depth Map Estimation from a Lenslet Light Field Camera," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1547–1555, 2015.

[48] G. Baasantseren, J. Park, N. Kim, and K. Kwon, "Computational integral imaging with enhanced depth sensitivity," *Journal of Information Display*, vol. 10, no. 1, pp. 1–5, 2009.

[49] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," *International Journal of Computer Vision*, vol. 92, pp. 1–31, mar 2011.

[50] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2432–2439, IEEE, jun 2010.

[51] R. C. Bolles, H. H. Baker, and D. H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *International Journal of Computer Vision*, vol. 1, no. 1, pp. 7–55, 1987.

[52] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross, "Scene reconstruction from high spatio-angular resolution light fields," *ACM Transactions on Graphics*, vol. 32, p. 1, jul 2013.

[53] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi, "Depth from combining defocus and correspondence using light-field cameras," *The IEEE International Conference on Computer Vision (ICCV)*, pp. 673–680, 2013.

[54] S. Wanner and B. Goldluecke, "Variational light field analysis for disparity estimation and super-resolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 606–619, 2014.

[55] S. Heber, R. Ranftl, and T. Pock, "Variational shape from light field," in *Energy Minimization Methods in Computer Vision and Pattern Recognition* (A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, eds.), (Berlin, Heidelberg), pp. 66–79, Springer Berlin Heidelberg, 2013.

[56] F. Frigerio, *3-dimensional Surface Imaging Using Active Wavefront Sampling. PhD thesis*. PhD thesis, Massachusetts Institute of Technology, 2006.

[57] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 504–511, 2013.

[58] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 49–56, 2013.

[59] P. Tan and P. Monasse, "Stereo Disparity through Cost Aggregation with Guided Filter," *Image Processing On Line*, vol. 4, pp. 252–275, 2014.

[60] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–11020042, 2004.

[61] Q. Yang, R. Yang, J. Davis, and D. Nistér, "Spatial-depth super resolution for range images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1 – 8, 2007.

[62] S. Matuska, R. Hudec, and M. Benco, "The comparison of CPU time consumption for image processing algorithm in Matlab and OpenCV," *Proceedings of 9th International Conference, ELEKTRO 2012*, pp. 75–78, 2012.

[63] "OpenCV."

[64] B. Barney and L. L. N. Laboratory, "Introduction to Parallel Computing."

[65] W. Kim and M. Voss, "Multicore desktop programming with Intel Threading Building Blocks," *IEEE Software*, vol. 28, no. 1, pp. 23–31, 2011.

[66] J. Reinders, *Intel threading building blocks - outfitting C++ for multi-core processor parallelism*. O'Reilly Media, 2007.

[67] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[68] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[69] V. Kolmogorov and R. Zabih, "What Energy Functions can be Minimized via Graph Cuts?," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[70] M. Rerabek and T. Ebrahimi, "New Light Field Image Dataset," *8th International Conference on Quality of Multimedia Experience (QoMEX)*, 2006.

[71] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[72] P. Corcoran and H. Javidnia, "Accurate Depth Map Estimation from Small Motions," *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, vol. 2018-Janua, pp. 2453–2461, 2018.