DEPTH MAP ESTIMATION USING FOCUS AND APERTURE BRACKETING FROM A MODIFIED CANON 600D CAMERA

Palomo Domínguez, Àlex
Curs 2016-2017

Director: JUAN FRANCISCO GARAMENDI BRAGADO
GRAU EN ENGINYERIA EN SISTEMES AUDIOVISUALS

Treball de Fi de Grau
DEPTH MAP ESTIMATION USING FOCUS AND APERTURE BRACKETING FROM A MODIFIED CANON 600D CAMERA

Àlex Palomo Domínguez

BACHELOR THESIS UPF
DEGREE IN AUDIOVISUAL SYSTEMS ENGINEERING
YEAR 2017

SUPERVISOR
Juan Francisco Garamendi Bragado
Image Processing and Computer Vision Research Group
Als meus pares i al meu germà.
ACKNOWLEDGEMENTS

Aquest treball ha estat possible gràcies a moltes persones, sense les quals no hagués aconseguit molts dels propòsits d’aquests últims anys. Vull agradir sincerament i de tot cor:

Als meus pares, per haver-me donat sempre les millors oportunitats i recursos per poder aconseguir totes les meves aspiracions. Sense el seu suport i la seva estima, no hauria pogut fer de la meva vida, el que és avui en dia. Gràcies per entendre’m sempre i per donar-me la mà incondicionalment.
Al meu germà, per demanar-me ajuda sempre que ho ha necessitat, tant en l’àmbit acadèmic, com en el personal. Per fer-me sentir un bon germà i per ser-ho tu amb mi. Òscar, t’anoixo a seguir estudiant i descobrir tot el que puc arribar a aprendre.
A la meva família en general, per haver-me fet sentir especial en cada pas i per sentir-se orgullosa de mi en molts aspectes de la vida.
Al supervisor d’aquest treball, Juan Francisco Garamendi Bragado. Estic molt agraït per tota la dedicació i temps que ha invertit en mi i el meu projecte. Mai ha dubtat en ajudarme i en donar-me tots els recursos possibles. Els seus ànims també han estat imprescindibles. Gràcies també per ser un dels professors en fer-me descobrir el meu interès en el camp de Visió per Computador. Juan, des del principi vaig saber que havia fet una bona elecció.

Als meus companys i amics de carrera, en Joan i en Roger. Per les incomptables hores a la biblioteca lluitant per el títol d’enginyer. Per totes les nits que ens hem passat desperts entre apunts i ordinadors. Per ser el meu recolzament en molts àmbits de la vida. Per haver estat unes de les persones més importants d’aquests quatre anys. Roger i Joan, gràcies per ser tan bons amics, no us vull perdre.

A tots els meus bons amics de carrera, Arnau, Martí, Marc, Javi, Gerard, Albert, Nico, Guillem i Alejandro. De tots ells m’emporto un record inoblidable que espero continuar mantenint vivu en el futur.
I per últim a tu, Núria, per ser la persona més especial d’aquests últim anys. Per totes les experiències que m’has regalat i per totes les que desitjo que vinguin. Per les incomptables vegades que m’has fet mirar endavant i oblidar tot el que no té importància. Per la teva estima i les teves ganes de viure. Per fer-me sentir especial. Núria, gràcies petita.
Abstract

In this work, we propose a variational model for depth estimation from an image sequence of a combined focus and aperture bracketing. In order to acquire the bracketed images, we modified a Canon 600D DSLR camera. We model the depth estimation problem as a minimization of an energy functional with a data fidelity term that takes into account the focus measures from different apertures. The energy to minimize is completed with a regularization term based on the Total Variation. Depth estimation using focus measures relies on local contrast. Homogeneous regions of the image have low local contrast, independently if they are focused or not, so this affects to the rightness of the estimated depth map. To overcome this problem, we propose a measure of reliability of the depth map and use inpainting techniques to improve the depth values on those areas with low reliability. The work is completed with the computation of an all-in-focus image. Finally, we also show experiments over different focus/aperture bracketings from various scenes and evaluate the behaviour of the algorithm by contrasting certain parameters.


Resum

En aquest treball proposem un model variacional per estimar la profunditat d’una seqüència d’imatges obtingudes a partir de la combinació de “bracketings” de focus i obertura de diafragma. Per tal d’obtenir aquesta seqüència d’imatges hem modificat el firmware d’una càmera DSLR Canon 600D. Modelem el problema d’estimació de profunditat com una minimització d’un funcional d’energia amb un terme de fidelitat de les dades que té en conta les mesures de focus de diferents obertures. L’energia a minimitzar es completa amb un terme de regularització basat en la Variació Total. L’estimació de profunditat mitjançant mesures de focus té en conta el contrast local. Les regions homogènies de la imatge tenen un contrast local baix, independentment de si estan enfocades o no, per tant això afecta a la credibilitat del mapa de profunditat estimat. Per afrontar aquest problema, proposem una mesura de credibilitat del mapa de profunditat i utilitzem tècniques d’”inpainting” per millorar els valors de distància d’aquelles zones amb una mesura de credibilitat baixa. El treball es completa amb el càlcul d’una imatge “all-in-focus. Finalment, també mostrem experiments sobre diferents “bracketings” de focus/obertura de diverses escenes i avaluem el comportament de l’algoritme contrastant certs paràmetres.

CONTENTS

Figures list ..................................................................................................................... xi

Tables list ...................................................................................................................... xii

1. INTRODUCTION ........................................................................................................ 1
   1.1. Objectives ............................................................................................................. 2
   1.2. Depth Estimation Strategies .................................................................................. 2
       1.2.1. Active Systems .............................................................................................. 3
       1.2.2. Passive Systems ........................................................................................... 3
   1.3. Motivation and Goal ............................................................................................. 6
       1.3.1. Camera modification ...................................................................................... 7
       1.3.2. Improved approach ....................................................................................... 8
       1.3.3. Inpainting model for homogeneous regions .................................................... 8
       1.3.4. Application of the depth map ........................................................................... 9
   1.4. Outline ................................................................................................................. 9

2. STATE OF THE ART ................................................................................................... 11
   2.1. Bracketing Using Magic Lantern .......................................................................... 11
   2.2. Depth Estimation Methods Using Focus Cues ....................................................... 14
   2.3. Variational Calculus (Energy Functional) ............................................................. 17
   2.4. Inpainting ............................................................................................................. 19
       2.4.1. Patch-based methods ..................................................................................... 19
       2.4.2. PDE and variational methods ....................................................................... 21

3. DEPTH ESTIMATION FROM DUAL BRACKETING ............................................... 23
   3.1. Dual Bracketing Module Implementation in Magic Lantern .................................. 23
       3.1.1. Basic camera functionalities ......................................................................... 23
       3.1.2. Programming Environment ......................................................................... 24
       3.1.3. Dual bracketing implementation and camera execution ................................. 25
   3.2. Depth Map Computation Method ......................................................................... 31
       3.2.1. New approach ............................................................................................... 31
       3.2.2. Preprocessing ............................................................................................... 33
       3.2.3. Focus measure operator ............................................................................... 35
       3.2.4. Gaussian fitting .......................................................................................... 38
       3.2.5. Energy functional ......................................................................................... 38
       3.2.6. Credibility map and inpainting ................................................................. 43
       3.2.7. Application ................................................................................................. 44
4. RESULTS AND EVALUATION .............................................................................. 45
  4.1. Dual Bracketing Module Evaluation .......................................................... 45
  4.2. Depth Map Computation Method Evaluation .......................................... 46
      4.2.1. General review .................................................................................. 47
      4.2.2. Preprocessing ................................................................................... 54
      4.2.3. Focus measure matrices ................................................................... 56
      4.2.4. Depth map weights .......................................................................... 60
      4.2.5. Credibility measure ......................................................................... 61
      4.2.6. Denoising parameter ........................................................................ 62
      4.2.7. Number of images in the sequence .................................................. 63
      4.2.8. All-in-focus image ............................................................................ 64

5. CONCLUSIONS AND FURTHER WORK ......................................................... 67
  5.1. Conclusions .............................................................................................. 67
  5.2. Further Work ............................................................................................ 68

BIBLIOGRAPHY ................................................................................................ 70
## Figures list

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Pg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Image projection of a real scene</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Image formation in a simple camera system</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Focus function</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Kinect</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Kinect depth map</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Project outline</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>Magic Lantern menu</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Dual bracketing focus function of a pixel</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>Magic Lantern install console</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>Magic Lantern uninstall console</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>Magic Lantern code main folder</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>Depth From Focus problem</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>Low-textured regions problem</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>Inpainting experiment by Bertalmio et al.</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>Efros and Leung’s method</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Ashikhmin’s method</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>Inpainting problem</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>Diaphragm aperture (f-number)</td>
<td>24</td>
</tr>
<tr>
<td>19</td>
<td>Aperture, shutter speed and ISO</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>Movie recording error</td>
<td>27</td>
</tr>
<tr>
<td>21</td>
<td>Live View error</td>
<td>27</td>
</tr>
<tr>
<td>22</td>
<td>Modules folder within Magic Lantern main folder</td>
<td>28</td>
</tr>
<tr>
<td>23</td>
<td>Magic Lantern modules menu</td>
<td>28</td>
</tr>
<tr>
<td>24</td>
<td>README file viewed on camera</td>
<td>28</td>
</tr>
<tr>
<td>25</td>
<td>Magic Lantern shoot menu</td>
<td>29</td>
</tr>
<tr>
<td>26</td>
<td>Dual bracketing menu</td>
<td>29</td>
</tr>
<tr>
<td>27</td>
<td>Dual bracketing image sequence</td>
<td>32</td>
</tr>
<tr>
<td>28</td>
<td>Focus surfaces</td>
<td>32</td>
</tr>
<tr>
<td>29</td>
<td>Camera set during the testing of the dual bracketing module</td>
<td>33</td>
</tr>
<tr>
<td>30</td>
<td>Luminance normalization example</td>
<td>34</td>
</tr>
<tr>
<td>31</td>
<td>Image registration example</td>
<td>34</td>
</tr>
<tr>
<td>32</td>
<td>Image registration using intensity-based registration method</td>
<td>35</td>
</tr>
<tr>
<td>33</td>
<td>Focus measure matrix</td>
<td>36</td>
</tr>
<tr>
<td>34</td>
<td>Focus quality of lens depending on the aperture</td>
<td>37</td>
</tr>
<tr>
<td>35</td>
<td>Focus quality of lens depending on the aperture (our lens)</td>
<td>37</td>
</tr>
<tr>
<td>36</td>
<td>Gaussian fitting example</td>
<td>38</td>
</tr>
<tr>
<td>37</td>
<td>Focus measure matrix in three different types of pixel</td>
<td>43</td>
</tr>
<tr>
<td>38</td>
<td>Inpainting example</td>
<td>44</td>
</tr>
<tr>
<td>39</td>
<td>Focus measure matrix of a textured pixel</td>
<td>48</td>
</tr>
<tr>
<td>40</td>
<td>Focus measure matrix of a pixel within the white region of the box</td>
<td>49</td>
</tr>
<tr>
<td>41</td>
<td>Defocus effect on a draw and its corresponding focus measure</td>
<td>49</td>
</tr>
<tr>
<td>42</td>
<td>3D representation corresponding to the second analyzed scene</td>
<td>50</td>
</tr>
<tr>
<td>43</td>
<td>Focus measure matrix of a pixel closer than the min. focus distance</td>
<td>51</td>
</tr>
<tr>
<td>44</td>
<td>Focus measure matrix of a pixel farther from the hyperfocal distance</td>
<td>53</td>
</tr>
<tr>
<td>45</td>
<td>Gaussian fitting through a focus measure matrix</td>
<td>58</td>
</tr>
<tr>
<td>46</td>
<td>Gaussian fitting through a focus function</td>
<td>59</td>
</tr>
</tbody>
</table>
Tables list

Table 1. Virtual Box configuration settings.......................................................... 25
Table 2. Dual bracketing module performance...................................................... 46
Table 3. Ideal scene depth map............................................................................. 47
Table 4. Ideal scene with low-textured regions depth map..................................... 48
Table 5. Unplanned scene depth map................................................................. 50
Table 6. Depth map of a scene presenting the hyperfocal distance problem........ 52
Table 7. Large space scene depth map................................................................. 53
Table 8. Focus bracketing method depth map...................................................... 54
Table 9. Luminance normalization effect in dual bracketing method.................... 55
Table 10. Image registration effect in dual bracketing method.............................. 56
Table 11. Comparison between different window sizes........................................ 57
Table 12. Illustration of the gaussian fitting effect............................................... 59
Table 13. Illustration of the weighting effect....................................................... 61
Table 14. Comparison between different credibility measures values............... 62
Table 15. Depth map using different denoising trade-off parameters............... 63
Table 16. Depth map computed using less images in the sequence.................. 64
Table 17. All-in-focus images from all the evaluated scenes.............................. 65
CHAPTER 1

1. INTRODUCTION

Nowadays, one of the most important goals of Information Technologies is to design and implement computational systems to obtain a high-level understanding from digital image and video content. This have gained an important research space in the scope of Computer Science, Mathematics, Physics, Signals Processing, and Engineering in general. Such interdisciplinary technologies have come to drive innovation in so many engineering areas. A great number of studies on the lasts decades has proved its huge potential in terms of automatizing human visual system tasks. These technologies are grouped together under the Computer Vision field, which eager to solve technical problems based on acquiring, processing, analysing and understanding digital images.

Computer Vision is becoming a fundamental component of a great percentage of systems, such as Image Retrieval based on the Content, Interpretation and Automatic Annotation of Videos, Three-Dimensional Information Extraction from Different Views, Improvement of Image Content or particular applications such as Autonomous Vehicle Guidance or Video Surveillance Intelligent Systems. It is known that such a field is highly related with the field of Artificial Intelligence since one of the reasons that explains the last decade exponential grow of the area advance is the fact that computational learning techniques have largely increased the performance of image information automatic extraction algorithms. This image data can take many forms, such as video sequences, views from multiple cameras or multi-dimensional data and as a technological discipline, computer vision attempts to apply its theories and models so that to achieve automatic visual understanding.

Specifically, this project deals with the first step to achieve one of the most typical tasks on the computer vision field: Depth Estimation. Depth estimation refers to the set of techniques and algorithms, that aim to reconstruct the spatial structure of a concrete scene. In other words, these methods seek to collect the third dimension of the real scene understood as the distance between the image capturing system and a specific point of the real scene. The input information which with computational systems must deal is commonly a two-dimensional representation of the captured scene, meaning that it has occurred a three-dimensional to two-dimensional transformation (i.e. image projection) (Fig. 1\(^1\)). During the image capturing process the set of light rays coming from the real world are projected in a plane (i.e. projection plane) meaning that the evident consequence is the loss of one dimension. In the case of traditional cameras, the projection plane is the image sensor which converts the natural light rays reflected from the objects to digital information that can be understood and processed by computational systems.

\(\text{Fig. 1. Image projection of a real scene.}\)

\(^1\) https://medium.com/retronator-magazine/pixels-and-voxels-the-long-answer-5889ecc18190
The idea of extracting information about the third dimension from digital images has always captivated the attention of computer scientists, since depth information is required in almost any 3D computer vision problem. So many of the tasks with which artificial vision has to deal with are performed in the three-dimensional world and it is imperative to obtain three-dimensional information such as the depth of the scene. Therefore, the perception of the depth is a fundamental ingredient in low-level computer vision and in the understanding of the natural relationship between the items on the scene. Different applications, require different scene reconstruction levels, so for example when dealing with distance measurements inside a scene, a simple depth description is needed, while in other cases where we are dealing with much more complex purposes for instance Automatic Car Parking, a complete three-dimensional description is needed. These last kinds of problems usually handle algorithms that mix complex computer vision methods.

Accordingly, in the following subdivisions, a general survey explaining the classification of the different contemporary systems able to obtain depth information from real-life scenes will be discussed and related with the main purpose of this project. A clear contrast between different kind of systems in terms of methods to retrieve 3D information is also exposed so that to understand the background that defines this work.

### 1.1. Objectives

Before starting to present the context behind the work, the objectives that define this project are presented:

- Implement a depth estimation method based on energy functional minimization from a focus and aperture bracketing.
- Compute a depth map robust to low-textured regions of the scene by means of a credibility map and inpainting techniques.
- Add a new functionality on a Canon 600D DSLR camera so that to obtain the dual bracketing sequence of images as input to the depth estimation method.
- Present some applications related to depth estimation such as all-in-focus image computing.

### 1.2. Depth Estimation Strategies

In computer vision, the human visual system has been a huge source of inspiration, and many of the strategies to recover depth attempt to achieve some human capabilities. The human visual system uses a huge collection of visual-based passive indications like motion parallax\(^2\), binocular disparity\(^3\), occlusion effects, variations in shadings as well as textures and edges of the viewed scene. During the last decades, so many algorithms has been designed to retrieve such lost information described above directly from the ambient light ray projected 2D image requiring some post-processing after the image acquisition. Such kind of techniques are known as passive methods. On the other hand, the most common and straightforward method is to use active devices which can capture the original image of the scene and simultaneously detect the depth of each of the pixels of the scene. These methods require additional sensors such as the infrared or sonar ones.

---

\(^2\) Motion parallax is the visual effect that describes the fact that near objects seem to move more than the farther ones when we are changing position.

\(^3\) Binocular disparity refers to difference in image location of an object seen by the left and right eyes.
They are usually more expensive than passive systems but also the results are more accurate. Both approaches seek to estimate a depth map, being a gray-scale image with the same size of the original two-dimensional image, that specifies the relative or apparent distance of each pixel from the camera to the scene objects. The following subdivisions eager to present these methods, which have a relative influence on this project.

1.2.1. Active Systems

Active systems involve using special devices that supply controlled energy beams such as incandescent light or ultrasounds. These techniques are known as active range-finding methods and its goal is to detect and analyse the reflected energy from the objects of the scene. A quick survey of the existing strategies can be found below:

**Light-based strategies:**

- **Time-of-Flight:** These methods estimate distances measuring the time that it takes for a pulse of lights to travel from the light emitter back to the light sensor. There exist many kinds of light for this purpose, but the most used is the infrared light. These sources are able to provide a short enough pulse to estimate valuable measurements. The main advantages of this technique are the high accuracy and the high processing rates. However, they use to present high power needs.

- **Incandescent light:** This method uses the electromagnetic waves produced by a conventional incandescent light bulb. The depth information of this method is very sparse, and has some disadvantages such as the dependence on the object colour and the small areas that can be analysed.

- **Pattern projection:** This technique was presented as an improvement of the incandescent light method since it projects a known pattern light to the scene. A camera captures the geometrical distortion of the pattern. This information together with image processing give accurate results.

**Ultrasound-based strategies:** Ultrasounds are sound waves with frequencies higher than the upper threshold of human hearing (from 20 kHz up to GHz). These strategies use the same time-of-flight principle. A common application is Medical Image, as the foetus examination.

1.2.2. Passive Systems

Passive systems are highly related with the computer vision field, since they attempt to understand the scene only using images as input, no other information is required in this kind of techniques. These image-based methods have a wider range of applicability with respect to active systems since they do not require any additional source of energy, only the natural light. Since human visual system is able to perceive depth from many different ways, an extensive effort to study new depth estimation algorithms has been made, resulting in many useful mathematical models. Examples of them are texture gradient analysis, occlusion cues or focusing/defocusing-based ranging. For the project purpose, it is convenient to classify the passive systems in two different groups: Monocular image-based techniques and stereoscopic image-based techniques.
**Multiview image-based techniques:** The human visual system is able to perceive the illusion of depth and three-dimensional structures by means of binocular vision. Both two-dimensional images perceived by each eye are combined in the brain so that to obtain the three-dimensional perception of depth.

- **Structure from motion:** This method is based on the natural phenomenon by which human beings are able to perceive depth information from the two-dimensional motion field of a moving object. When movement is perceived, either by the observer or the scene objects displacement, depth information is obtained from the acquired images over time using the relative motion between the object and the camera. The positional changes in the projection of a scene into the image plane can be analyzed as patterns of change over time, and thus get 3D information of the scene.

- **Depth from stereo:** This technique is a particular case of the structure from motion one. In both techniques, the correspondence between images needs to be found. The only difference between them is that in this second method, the relative motion is known a priori. The idea behind this model is that from a pair of two-dimensional images captured from two different viewpoints, the related information is combined so that to establish the depth of the surrounding. Stereo disparities (binocular disparity) resulting from the spatial shift between both points of views give the necessary information so that to view an object from different angles, thus obtaining three-dimensional information. Well known mathematical models such as Scale-Invariant Feature Transform (SIFT) [1] or Random Sample Consensus (RANSAC) [2] are used to extract spatial features from the scenery and find true correspondences between images. However, this strategy extends to multiple views of the scene, resulting in a full 3D model. Generally, the more views you have, the greater the range of angles that the 3D model can represent.

**Monocular image-based techniques:** Unlike the first mentioned kind of passive strategies, these algorithms only use one single point of view of the scene. As very sparse perspective information can be extracted from one single view, these methods usually present less descriptive depth estimations of the overall images, due to the image projection process limitations. However, they usually present low-level computation problems. Although there are other models than the ones exposed here such as *shape from shading* [3] or *optical flow* [4], we will focus on just two of them since they are the ones that relate the most with this work.

The following methods are based on the human visual perception of depth by focusing. Visual system has the capacity to adjust the focus distance of the eye in order to focus in a wide range of distances. This information is then processed by the brain to compute the distance of the focused plane. Similarly, photography cameras use the same principle so as to focus at different distances by changing the focal length of the camera lens. When a concrete region of the scene is in focus, and the internal parameters of the camera are known, information of the distance between this exact region of the image and the camera, can be estimated. Both of the following methods follow the “Thin Lens Law”[^4] to estimate absolute distances, but they yield different approaches.

The Thin Lens Law describes the image formation process for an aberration-free convex lens. Figure 2 shows the basic image formation geometry. The light rays coming from the scene that are radiated from a concrete object point \( p \) intercept on the lens converging into a point \( p' \) on the image plane (camera sensor). The relationship between the focus length of the camera \( f \), the distance \( u \) from the object to the lens and the distance \( v \) from the lens to the object image is given by the Gaussian (Thin) Lens Law:

\[
\frac{1}{f} = \frac{1}{u} + \frac{1}{v}
\]

(Fig. 2. Image formation in a simple camera system.

- **Depth from defocus**: A non-focused region of an image is understood as the focused region convolved by a blur function. This function represents the degree of defocus and it can be estimated as a function of the lens settings and the depth of the object. Hence, if the amount of blur can be retrieved and the camera settings are known, the depth information can be estimated. The method was first proposed by Pentland [5] and then justified by Subbarao [6]. The main advantage of this method is that only two images with different camera settings (thus, with different defocus degrees) are required to estimate depth. The result is comparable with the ones obtained with stereo vision or motion parallax. Moreover, the method can obtain the complete depth map, regardless whether any part of the image is focused or not.

- **Depth from focus**: The input of this method is a sequence of images where the distance between the lens and the image detector is changed between them. The result of this variation is an exact focus function (Fig. 3) defining the blur evolution for each pixel of the image along time axis (along frames), since a point in the scene is blurred to different degrees in each frame. Therefore, if we take a sequence of \( n \) photos, hence varying \( n \) times the focusing settings, we will get a focus function for each pixel of \( n \) different blur degrees. Such blur degrees are quantized by a certain focus/sharpness measure for each pixel in a local neighborhood of pixels around the analyzed one. Extensive studies have been made in order to find out the best focus measure, but it will be discussed in chapter 2. These measures usually analyze the high frequency content since defocusing a concrete point of an image is understood as a low pass filter operation. Once computed the focus function for each pixel, the frame number

\[\text{http://sipl.gist.ac.kr/about/SFF1.htm}\]
where the function is maximum is assigned to each pixel. Therefore, we end up having the relative distances between the points of the scene and the camera (Fig. 12). By knowing the camera parameters and by means of the “Thin Lens Law” we can compute the absolute distances. An extensive review of this method is presented on chapter 2.

![Focus function](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

**Fig 3. Focus function. The index of the maximum indicates the image in which the pixel is in-focus.**

A brief survey of the existing image-based methods to depth recovery has been exposed. There is no better strategy than other to depth estimation, since it totally depends on the problem to be solved. Complex applications such as autonomous car government need an accurate depth estimation and complete three-dimensional reconstruction, and they probably use combinations of different strategies, either active or passive, to obtain reliable information of the space. However, other applications such as automatic focus on digital cameras that need very low-level computations so that to be efficient, will be using much more simple algorithms such as depth from defocus. An example of a system that uses both strategies is the *Kinect* (Fig. 4) for the videogame console *Xbox 360*. It has two cameras and an infrared light sensor so that to ensure an accurate estimation. The following image (Fig. 5) shows a depth map computed by *Kinect*.

![Kinect](https://es.wikipedia.org/wiki/Kinect)

**Fig 4. Kinect.**

![Kinect depth map](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

**Fig 5. Kinect depth map.**

### 1.3. Motivation and Goal

This project seeks to implement a new approach to depth estimation using an improved version of the passive technique “Depth from Focus”. As introduced before, passive methods are image-based method since they study mathematical models so that to solve certain image processing problems and extract certain information from the scene. Such techniques group together under the computer vision field. We present our mathematical model based on a minimization of an energy functional by means of variational calculus. Exact specifications of the energy functional minimization will be provided in the following subdivisions. Before entering in detail, we should define the outline of the whole project. On the figure 6 the different steps that conform the work are specified: Camera modification, improved depth from focus approach, inpainting model for homogeneous regions, applications.

---

6 [https://es.wikipedia.org/wiki/Kinect](https://es.wikipedia.org/wiki/Kinect)
7 [http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)
1.3.1. Camera modification

Our algorithm takes as input a sequence of images from a combined focus and aperture bracketing. A bracketing consists on capturing sequences of images from the exact point of view (i.e. no camera shifting) by varying some camera parameters between each image. In this case, we vary the focus by sweeping the focus plane through the scene and the diaphragm aperture of the camera lens. Both camera parameters change the degree of defocus through the image. Concretely, a modification of the Canon 600D camera has been implemented so that to achieve this bracketing. We have called it “Dual Bracketing” and works in the following way: for each focus step of the camera, we perform N aperture steps, going from greater aperture (smaller depth of field) to smaller aperture (higher depth of field). The depth of field specifies the distance between the nearest and farthest objects on the scene that appear focused (sharp), in such a way that a small depth of field means that less range of objects are focused at the same time. A complete focused image can be obtained by closing the aperture as much as we can. Therefore, if we take a total of twenty focus steps and ten of aperture, we will end up with a sequence of two hundred images. Although it seems a lot of images, the algorithm cost computation is quite acceptable.

To implement this modification of the camera, the add-on firmware for Canon cameras called Magic Lantern\(^8\) has been modified. Magic Lantern is a portable free software that runs from an external memory such as an SD card and adds new features to Canon EOS cameras that weren’t included from the factory by Canon. Is an open-source code developed by a huge community of hackers. All the code is available on the web meaning that anyone can download it and try to implement new functionalities. Moreover, a developer’s forum is open to everyone and is the current larger source of information so that to search or solve coding doubts referring to Magic Lantern development. We have implemented a new feature that performs the dual bracketing explained before.

\(^8\) http://www.magiclantern.fm/
1.3.2. Improved approach

With the dual bracketing approach, we achieve as the algorithm input a sequence of images from a combined focus and aperture bracketing. As mentioned above, the Depth From Focus theory establish that the evolution of each pixel along the bracketed frames of the sequence, form a one-dimensional focus function (Fig. 3) where the vertical axis is represented for some focus measure that quantizes the sharpness of a concrete pixel with respect to a local neighbourhood of pixels. These measures usually analyze the high frequency content since defocusing a concrete point of an image is understood as a low pass filter operation. An example of these measures based on the image statistics is the gray level variance inside a specific analysis window. A comparison of different focus measure operators for autofocus can be found on [7]. Now, the problem has extended to a two-dimensional function or surface due to the X new focus functions added to the problem, where X corresponds the number of apertures used for the bracketing. A representation of the new problem is showed in figure 8.

![Fig 8. Dual bracketing focus function of a pixel. There is a focus function for each aperture.](image)

Now the algorithm has to deal with a two-dimensional problem. As we can see, in this case the frame where the pixel has the maximum focus measure is approximately the tenth one. The standard deviation of the following focus functions gets larger as the aperture decrease, because the smaller the aperture, the more pixels in the image are in focus. We compute the exact focused frame by means of an energy minimization using variational calculus. The energy functional has a data fidelity term that takes into account the focus measures from different apertures and a regularization term based on the Total Variation, meaning that smoothness between the pixels of the depth map is required but allowing discontinuities on the obtained depth map. Before minimizing the energy functional, a Gaussian estimation of each focus function of every pixel in the scene is computed.

1.3.3. Inpainting model for homogeneous regions

Because of the fact that focus measures rely on local contrast, reliable results are only ensured when the pixels of the neighbourhood determined by the window contain high-frequency information, i.e. when the object has texture or edges. For those areas where a homogeneous value dominate, estimated depth is not consistent and noise appears to the depth map. In order to solve this problem, we use a credibility map that gives a reliability measure for each pixel. This credibility measure quantizes how sure is the algorithm that a distance is well approximated. When we achieve this mentioned map, an inpainting method is applied through the depth map so that to fill the not reliable regions with the edges content. The inpainting model used follows a Laplacian diffusion.
1.3.4. Application of the depth map

Estimating distances has many applications, from the most unimaginable to the simpler ones. In our case, since the distance where a concrete pixel is focused is known, we finally compute an all-in-focus image, where all the regions on the image are in focus.

1.4. Outline

This report is organized in five different parts. The first one has been yet presented and it is the introduction, in which basic notations and definitions has been explained in order to get familiarized with the field. Moreover, the general scheme of this work has been introduced so that to get the first idea. The next three parts compose the bulk of the work, in which the work details are explained.

Chapter 2 introduces the previous work made on the subject. Concretely the chapter is divided into the previous work referring to the Magic Lantern firmware and the state-of-the-art methods for depth estimation using focus information. In the first part a detailed explanation on Magic Lantern programming environment as well as an introduction to its capabilities, are held. In the second part, the main Depth From Focus methods that have inspired and completed this work, are presented. A detailed introduction to variational calculus as well as inpainting techniques are also held.

In chapter 3, the concrete methodology of this work is presented, and is also divided into the Magic Lantern firmware implementation and the depth estimation method. In the first part, a precise explanation of the dual bracketing module implementation is presented, from the programming environment to the specific libraries used. This part also holds a brief introduction to the main camera parameters, so that to understand the goal of the code. Secondly, the depth map computation algorithm is accurately described, from the focus measurement of the images, to the inpainting process. Moreover, the mathematical demonstration of the energy functional formulation defining this approach, is also presented.

In chapter 4, an extensive evaluation on both parts of the project is performed. The dual bracketing module is analyzed into its possible configurations, so that to get a detailed knowledge of its capabilities. Secondly, the depth estimation method is evaluated on different scenes so that to know its limits and understand how each part of the algorithm works. An extensive visual comprehension of the method results is achieved. Finally, the all-in-focus images corresponding to the analyze scenes, are presented.

Finally, in chapter 5, the final conclusions achieved from this work are presented as well as the further work to be done so that to improve the project.
CHAPTER 2

2. STATE OF THE ART

In the first chapter, a review of the parts in which the project is divided, has been introduced. So that to make things clear, we would treat this work as a set of clearly differentiate parts: Camera modification using Magic Lantern, depth map computation and applications. In each of these fields, so many studies and projects with totally different purposes, have been made. However, from each of them we can find specific approaches and literature from where we can take advantage of and the required knowledge so that to settle this complete image-based system. Therefore, from each of the aforementioned parts, the state of the art approaches are introduced.

2.1. Bracketing Using Magic Lantern

Magic Lantern is an open framework under the GNU General Public License (GPL)\(^9\) for developing new features as an extension to the official Canon software. Since 2009, Magic Lantern has been enhancing the possibilities of the Canon DSLR cameras by adding new features to the original firmware that are not included from factory by Canon. Useful features such as “Audio monitoring”, “Focus and exposure assistants”, “RAW video”\(^10\), “HDR video” or “Intervalometer” are examples of how this firmware is able to go beyond the limits of the embedded software of DSLR Canon cameras. In this section, additional information and detailed steps to develop in Magic Lantern are provided. Note that Magic Lantern has much more capabilities than the ones described here, but we only describe the steps we needed to implement our dual bracketing module.

Magic Lantern does not modify the original software but runs from the card as an add-on over the official software, meaning that all the Canon functionalities are still accessible once Magic Lantern is installed and running. All the code developed is possible because a large reverse engineering\(^11\) effort so that to know the complex structure of the software and hardware of Canon cameras. Many doubts have originated related to its legacy, but since it is an interoperable product that does not use any Canon code, it can be labeled as legal. The project was started by Trammell Hudson\(^12\) in 2009 and was originally implemented on the Canon EOS 5D Mark II. Nowadays, is available for many Canon DSLR models\(^13\) and every day the community strives to reach as much models as they can. Magic Lantern is possible because another Free and Open-Source Software (FOSS) called Canon Hack Development Kit (CHDK)\(^14\). It is another open framework that aims to integrate new functionalities to stock Canon software, but in this case, for point-and-shoot cameras. Although the Magic Lantern code is completely build from zero, they have used some of the CHDK tools so that to learn about Canon firmware files. CHDK were the first to understand the Canon real-time operating system DryOs as well as specific Canon firmware files and the boot process. Magic Lantern took advantage of it.

---

\(^9\) [https://www.gnu.org/licenses/gpl-3.0.en.html](https://www.gnu.org/licenses/gpl-3.0.en.html)
\(^10\) [https://www.magiclantern.fm/forum/index.php?board=49.0](https://www.magiclantern.fm/forum/index.php?board=49.0)
\(^11\) [https://www.lumendatabase.org/topics/15](https://www.lumendatabase.org/topics/15)
\(^12\) [https://trmm.net/Magic_Lantern_firmware](https://trmm.net/Magic_Lantern_firmware)
\(^13\) [http://builds.magiclantern.fm/](http://builds.magiclantern.fm/)
Magic Lantern is installed\textsuperscript{15} in the camera as if it was firmware update from Canon. When updating the firmware from the Canon menu, the camera check for new Canon updates and if compatible, it installs the package. The file “ML-SETUP.fir” inside the Magic Lantern folder are false firmware updates that when are loaded, instead of installing new firmware, it sets the \textit{boot flag}\textsuperscript{16} and makes the card bootable. Magic Lantern loads from an external card where the executable and the rest of the code is stored. When the camera is turned on, the \textit{boot loader}\textsuperscript{17} checks the boot settings of the system in its re-writable memory. During the installation of Magic Lantern, a boot flag is set to the camera so that when the system is loading, the boot loader knows which partition to load after checking the boot settings. In the next step, the boot loader check if the external card (SD, SDHC, SDXC, CF) is bootable. A bootable card is a kind of hardware that contains and can read the files so that to initialize a computer system. Once the boot loader has found the bootable card, the Canon firmware loads along with Magic Lantern additional routines. The first Magic Lantern file loaded is “autoexec.bin” which is the main executable of the firmware.

Documentation about how Magic Lantern should be installed/uninstalled is provided on their Wiki. They provide some suggestions and hints to avoid possible problems even permanent damages. These are the steps to follow in order to success both installing and uninstalling the software:

**Installation steps**

1. Make sure SD card only has Canon original files (Advise: Format the SD card from the camera).
2. Copy the files from the folder where are the binary files, to the SD card.
3. Insert the SD card on the camera.
4. Turn on the camera and access to menu → 3rd settings tab → Firmware Ver. → OK (this will install ML).
5. Reboot camera and make sure that ML has been installed.

**Uninstall steps**

1. Turn on the camera → 3rd settings tab → Firmware Ver. → OK (an ML message pops-up).
2. To uninstall we must wait 1 minute and it will automatically proceed. Then reboot the camera (Advise: Format the SD card from the camera).

\textsuperscript{15} http://wiki.magiclantern.fm/install  
\textsuperscript{16} https://en.wikipedia.org/wiki/Boot_flag  
\textsuperscript{17} https://en.wikipedia.org/wiki/Booting
The firmware is coded in the programming language *C* and it uses some routines to handle the boot strapping explained above. The build tools required to create the program executable are written in *Perl*. The build tools are responsible to compile and package the source code into an executable form. It also uses *Makefiles* that specify how to compile and link the different programs so that to be able to track the different dependencies of the source code. It needs the *arm-linux-elf* toolchain of *GCC* compiler system to be able to compile the code as well as the *GNU Binary Utilities* (*binutils*) programming tools. Furthermore, Magic Lantern provides a pre-built toolchain and even a pre-built virtual machine that directly supplies the programming environment needed to start developing.

The code is full available on a *Bitbucket* repository where anyone is able to access and download or clone the source code as well as report issues and make *pull requests*. By means of pull requests, developers send they code implementations of new features and after an extensive review, Magic Lantern main developers decide if this new functionality should join the official Magic Lantern package. The package is distributed as a *Nightly Build*. Since so many developers work on the code and changes are produced every day, nightly builds ensure the correct performance of the program day by day. An automatic program daily compiles the binaries from the latest source code. It does not guarantee the correct performance, since nightly builds contain untested code, but it allows to test the latest implemented features.

Once downloaded the source code and having the programming environment ready, developers have all the required tools to start hacking. The Magic Lantern main folder is structured as showed in figure 11.

There is a great quantity of subdivisions and each having so many code files, but this work has only considered a few of them. The main “.c” files are in the “src” folder. A new component can be coded inside the existent files or it can be created from scratch. Any change here will be directly added to the default Magic Lantern menu, meaning that the new feature will be fixed in the menu. This is the first way to develop and implement new functionalities. The second way is by *LUA* scripting in the “script” folder. For those programming tasks that do not require many resources and are quite easy to

---

18 https://en.wikipedia.org/wiki/Makefile
19 http://magiclantern.wikia.com/wiki/Build_instructions/Unified
20 https://gcc.gnu.org/
21 https://www.gnu.org/software/binutils/
23 https://bitbucket.org/hudson/magic-lantern/
implement, a good option is to implement it by scripting using LUA. LUA is a cross-platform, multi-paradigm, lightweight programming language designed for embedded systems and based on C. Since it does not need so many means and has portability, Magic Lantern was able to implement it on the camera microcontroller. LUA is implemented as a module, and once loaded on the camera, Magic Lantern offers a specific menu for Scripts, in which one can choose to run one script or other and even code from the camera using a text editor. This leads to the third way to code, modules. Modules are independent programs written in C that run just when the user load them. A specific modules menu is on the main menu of Magic Lantern. Inside this menu there are all the modules “.mo” implemented and are only added on the relevant menu if loaded.

When implementing a new functionality, there are a huge quantity of methods and functions yet coded, that can be used. For example, the header “menu.h” from the source file “menu.c” contains all the functions and structures related with creating menus on the system, or the header “lens.h” from “lens.c” contains so many functions related with the camera lens, from getting data from it to mechanically control the lens. As introduced before, the most efficient way to get some documentation of the code is to search and read on the forum. However, in the Magic Lantern main folder, there is attached a Doxyfile as seen on the figure 11. This file is needed to automatically generate code documentation using the software Doxygen. Unfortunately, the code is not labeled and documented as it should be, giving a poor documentation.

Two of the Magic Lantern features that relates the most with this project are the ones called “Advanced Bracket” and “Focus Stacking”. The first one performs separately three types of bracketing: flash, Depth of Field (aperture) and exposure. The user can choose between one of them. On the other hand, the “Focus Stacking” feature performs a focus bracketing. This last functionality is useful for those cases in which the user wants to take an all-in-focus picture. Therefore, taking an image sequence in which the focus plane is swept through it and by using an external software, an all-in-focus can be obtained. A common example where this technique is used is in macro photography.

2.2. Depth Estimation Methods Using Focus Cues

A large number of methods in the literature to compute depths maps using focus cues has been studied during the last two decades. Most of them aim to determine the depth of an object by measuring its sharpness (focus degree) along the images of a focal stack and maximizing a corresponding focus function (Fig.3). These methods usually also seek to model the point spread function that tries to define how a pixel is blurred as it moves away from the focus plane. Here, some of these methods are briefly reviewed and related to this work.

First efforts to estimate the camera-to-object distances for components in the scene with sharp focus has been done in the past [8], and many autofocusing systems based on contrast detection have used them extensively. Depth from focus technique was first proposed by P. Grossman [9] and later detailed by A. Pentland [10]. They suggested estimating the depth of image points by evaluating the image blur due to the defocusing.
\[ D = \frac{F \cdot v_0}{v_0 - F - \sigma \cdot k \cdot f} \]  

where \( D \) is the absolute distance of a pixel, \( F \) is the focal length, \( v_0 \) is the distance between the lens and the image plane, \( f \) the lens aperture, \( k \) a constant of proportionality and \( \sigma \) a spatial constant of the point spread function. As we can see, these methods imply knowing the internal parameters of the camera such as the focal length, the aperture of the diaphragm and the distance between the image sensor and the lens, so that to accurately approximate the point spread function that defines the defocusing. Moreover, they proved that the point spread function can be described by a two-dimensional Gaussian \( G(r,a) \) that depends on a spatial constant \( \sigma \) and a radial distance \( r \). The accuracy of such method is greatly dependent on the blurring model used and since they are approximations from the physical-optics models, they do not ensure high quality results.

S. K. Nayar and Y. Nakagawa in *Shape from Focus* [11] were the first to use different focus levels to obtain a sequence of images differing in sharpness (focus stack). This method aimed to develop a model to describe the variation of the focus measure values because of defocusing. Then this model is used by a depth estimation algorithm to interpolate focus measure values. By studying the behavior of the focus function, this study proved that it is reasonable to model it as a Gaussian function. They relied on the fact that although a lens can precisely focus to a concrete distance, the sharpness decreases gradually on each side of the focused distance, meaning that the limits of the depth of field are not hard boundaries between sharp and unsharp (Fig. 12). Other investigations have proposed different fittings of the focus function such as quadratic or polynomial ([12]), as well as other fits that consider the intensity distribution of the focused image along the optical axis ([13]). Tsai and Chen [14] proposed a focus profile model based on the horizontal gradient of the image. Moreover, it proposes a new focus measure based on the sum-modified-Laplacian (SML). This measure of sharpness has desirable properties such as simplicity, retention of necessary information and elimination of unnecessary information, but it requires some computation time.

![Focus Stack](image.png)

*Fig. 12. Depth From Focus problem. The red pixel is within a closer object than the green one.*

The problem with the focus-based methods is that they rely on the presence of high frequency brightness/intensity variation in the scene, meaning that for those regions on the image that are not rough such as flat surfaces, the methods do not perform well as seen on figure 13. [11] explains what is the concept of “Visibly Rough Surfaces”. Therefore, since these formulations has problems in homogeneous regions, extensive efforts dealing with focus measures has been made in order to find the best focus measure ([7], [15], [16], [17], [18]). The common measures in the literature are based on image statistics (gray level variance [19], image gradients (gradient energy [15] and Tenengrad algorithm [7]), Modified Laplacian [11], Histogram Entropy of the image as well as methods based on image pyramids or wavelet decomposition.
Other popular focus measures are within the field of spatial domain focus operators. To apply this kind of operators throughout the image, the domain is divided into block of fixed size, as proposed by Li, Kwok, and Wang in [20]. A block should be large enough to capture an accurate measurement of the focus level and small enough to only include regions with a similar focus degree. Apart from the block-wise methods, methods taking pixel-wise has also been proposed ([21]). In this case, the operator is computed for every pixel in the image by taking into account a small region of pixels around the desired pixel. Then a focus measure matrix $F_k$ is computed for each image $I_k$ of the focus stack. The values of $F_k$ in a concrete pixel $(x,y)$ along all the focus stack is the focus function explained before (Fig. 3).

During the last years, many modern approaches have been presented in order to overcome some of the problems described above. Another frequent problem in these kind of methods is the magnification problem. Most of the aforementioned strategies, usually obtain the images by means of laboratory instruments having perfect optical lenses. This implies that their methods do not work for general scenes, where a large amount of imperfections have to be taken into account. Therefore, they assumed known calibration parameters of the camera, constant illumination of the scene, parallax free scenes and constant-magnification input sequences such as the ones taken by a telecentric camera lens [22]. Previous work attempted to correct magnification changes through similarity transform [23] or by scaling and translating [24]. Methods such as the one proposed on [25] try to align the input sequence of images so that to solve the magnification error. They propose a new approach that is able to compute a depth map by taking the input images with a moving mobile phone, introducing an optical flow concatenation approach so that to align the images and achieve constant magnification. Moreover, the method is robust to an interesting topic to keep in mind: the brightness constancy assumption. The defocusing process violates this assumption, since it effects on the brightness of the obtained image and creates sharp circular expansions known as bokeh. Other methods also attempt to solve the problems described above [26]. This one obtains all-in-focus RGB-D video of dynamic scenes with a commodity digital camera. An interesting method [27] proposes a selectivity measure to determine if the focus measure is reliable or not. This information can be used to made an adaptively method that is able to differentiate between good and wrong estimations. However, they use it to compute an all-in-focus image by means of image fusion.

---

27 Telecentric lens does not exhibit varying magnification for objects at different distances from the lens. The variation of magnification with distance causes several problems for machine vision and other applications.
Other authors such as Surya and Subbarao in [28] has researched in other defocusing parameters as for example the aperture of the camera diaphragm instead of sweeping the focal plane across the scene ([10] [29]). Although this approach avoids magnification problems, the defocus effects are less pronounced when varying aperture size compared to varying focal depths [22]. Another approach studied is to fix the camera and the focus and translate the object [11]. However, they have to deal with other problems such as the deformation introduced by non-perfect camera lens.

Very recent studies [30] have presented innovative methods that run over the limits of this problem by using techniques such as deep learning (machine learning) following the theory of Depth from Focus. They call this new approach DDFF (Deep Depth From Focus) and seek to overcome typical problems such as the sharpness level in low-textured areas.

2.3. Variational Calculus (Energy Functional)

Variational calculus is the branch of mathematical analysis that aims to maximize or minimize functionals, which are mappings from a set of functions to the real numbers. The functional formulation depends on the choice of functions subject to different constraints such as integral, differential or phase. In other words, by means of variational calculus an extrema function that makes the functional achieve a maximum or minimum value is computed. A simple example could be to find the shorter curve that joins two points. It is obviously a straight line, but if it is subject to particular constraints (e.g. the curve has to lie in a concrete surface), the problem becomes much more complex. In the field of computer vision, so many problems have been solved by its variational formulation since the solution is the minimizer of an energy functional. Problems such as denoising [31], inpainting [32] [33] [34] or optical flow [35] are examples of problems solved by means of energy functional minimization. Formally, in image processing, a functional $J$ has the following form:

$$J: \mathcal{V} \rightarrow \mathbb{R}$$

$$u \mapsto J(u) = \int_{D} F(x, u(X), \nabla u(x)) \, dx ,$$

where $\mathcal{V}$ is a specific space of functions, $D \subset \mathbb{R}^N$ is an unbounded domain of $\mathbb{R}^N$, $u \in \mathcal{V}$: $D \rightarrow \mathbb{R}$ is a function defined on $D$, $\nabla u(x)$ is the gradient of $u$ and $x$ is the independent variable. The problem consists on finding the function $u$ that minimizes the functional $J(u)$. The expression $F$ refers to the operations applied through the input data and its formulation depends on the problem to be solved. Its typical structure is usually wrote as

$$F(u) = \text{fidelity}(u) + \text{regularity}(u)$$

The first term refers to the fidelity of the solution to the input data. This part imposes the solution to be similar to the input values, for example in a denoising problem in which is desired to remove the noise of a given image, the final denoised image is required to be similar to the original one. Therefore, this is the main term in which the simplified problem is presented. However, with only this constraint there would be infinite solutions for the problem, or it may be the trivial one, that is why a second term is added to the data fidelity term. In this case, the regularity term imposes smoothness on the solution $u$
is introduced to reduce the set of possible solutions. Depending on the application, one of these terms is desired to have more presence than the other. That is why in order to weight the contribution of each one, a trade-off parameter $\lambda$ is added to the formulation:

$$F(u) = \text{fidelity}(u) + \lambda \ast \text{regularity}(u)$$  \hspace{1cm} (4)

In this case, smaller values of $\lambda$ will minimize the contribution of the regularity term, meaning that the solution will be closer to the initial data. This is usually a case-sensitive parameter that needs to be fit for the proposed application.

Once the functional formulation is specified, the following step is to minimize it. The process of minimizing a functional composes by computing its derivate and equaling it to zero. In order to derivate the functional (3), it is used the so-called Gâteaux derivate, which is a generalization of the concept of directional derivate in differential calculus. This type of derivate computes the derivate in the direction of a given vector with respect to a function with finite variables. Therefore, the Gâteaux derivate of the functional $J(u)$ with respect to the function $u$ in the direction of function $h \in V$ is defined as:

$$\frac{dJ}{du} \bigg|_h \lim_{\varepsilon \to 0} \frac{J(u + h\varepsilon) - J(u)}{\varepsilon}$$ \hspace{1cm} (5)

Then, the necessary condition that must be satisfied by the minimum of $J(u)$ is:

$$\frac{dJ}{du} \bigg|_h = 0$$

By developing the Gâteaux derivate definition subject to the last condition, the so-called Euler-Lagrange equations arise. The Euler-Lagrange equation for $J(u)$ is given by:

$$\frac{\partial F}{\partial u} - \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left( \frac{\partial F}{\partial u_{x_i}} \right) = 0 ,$$ \hspace{1cm} (6)

where $x_i$ are the independent variables of $u$. The Euler-Lagrange equation usually is subject to boundary conditions that infer in the computation of the solution on the domain edges. The so-called Dirichlet condition specifies the value that the solution has to keep in the domain borderline. Another typical condition is the so-called Von Neumann which specifies the values in which the derivate of a solution is applied within the boundary of the domain. Other conditions such as Cauchy boundary, Mixed boundary or Robin boundary are types of combinations of both Dirichlet and Von Neumann conditions.

A common functional in the field of denoising is the following one:

$$J(u) = \frac{1}{2} \int_{D} |\nabla u|^{2} \, dx + \frac{1}{2\lambda} \int_{D} |u - f|^{2} \, dx \, dy ,$$ \hspace{1cm} (7)

where $u$ is the solution, $\lambda$ is the trade-off parameter described before and $f$ is the input image. The first part of the functional corresponds to the regularization term and imposes the solution to be smooth inside the domain. The second term refers to the fidelity part of the functional. It is regularized by a trade-off parameter and explains the fact that the
solution and the original image have to be similar. This functional is solved by minimizing it using the Euler-Lagrange equation introduced in (6). In this case, the minimized function ends up in the following partial differential equation:

\[
\frac{1}{\lambda} (u - f) - (u_{xx} + u_{yy}) = 0,
\]

where the term \((u_{xx} + u_{yy})\) corresponds to Laplacian operator of \(u\). Then, the solution is found by solving (8). It can be computationally solved by discretizing the Laplacian operator by means of gradient descent using advanced differences for the gradient and retarded differences for the divergence.

### 2.4. Inpainting

In the field of computer vision, most specifically in the area of image restoration techniques, so many problems related with the loss of information in images have been solved by means of the so-called Inpainting technique. This technique aims to reconstruct lost or deteriorated parts of images and videos where the original information is unknown. It has many applications such as restoring damaged portions of an image or removing unwanted objects from a given scene. Formally, given an image and a region \(\Omega\) inside it, the inpainting process seeks to find the values of the image pixels inside \(\Omega\) so that this region looks similar and consistent to its surroundings. The inpainted region \(\Omega\) is specified by the user or another external process different from the inpainting. Therefore, the location of \(\Omega\) is not part of the inpainting process. Therefore, the input of an inpainting model is the original image and a mask specifying the inpainted domain. In figure 14 we can see an example. Many studies have proved the benefits of using inpainting in so many applications and the effort to improve its performance has come to drive new methods that can be classified into three categories: patch-based, sparse, and PDEs/variational methods.

![Fig. 14. An experiment taken from Bertalmio et al [34]. Left: original image. Middle: a user-defined mask. Right: the result with the algorithm of [34].](image)

#### 2.4.1. Patch-based methods

Patch-based inpainting methods aim to find the pixel values of the inpainted region by means of a large comparative between the regions around pixels to fill and those surrounding the existent pixels. The first method was proposed by Efros and Leung [36] even though the initial approach was though to texture synthesis problem. They proposed
a technique in which the image gap is filled recursively inwards from the hole boundary. Each pixel \( P (P \in \mathcal{D}) \) on the hole boundary is filled by the value of an existent pixel \( T (T \in \mathcal{D}) \) so that the neighborhood around \( T (\psi(T)) \) is most similar to the region around the pixel \( P (\psi(P)) \). The neighborhoods are defined as square patches centered at \( P \) and \( T \) respectively. This can be expressed as an optimization problem where the algorithm seeks to find the \( T \) pixel that minimizes the distance between both neighborhoods. Formally:

\[
Value(P) = Value(T), \quad T = \arg \min d(\psi(P), \psi(T))
\]  

(9)

Efros and Leung proposed the Sum of Squared Distances as the similitude measure between the regions:

\[
d(\psi_P, \psi_T) = \sum_i \sum_j |\psi_P(i,j) - \psi_T(i,j)|^2
\]

(10)

Once the value of \( P \) is filled, the method jumps to the next pixel on the hole boundary. The main disadvantage of this method is the computational cost since it has to compare every region on the image in each step. Moreover, the selection of the neighborhood size is made by the user and is case-sensitive. On the other hand, the filling order also affects the performance of the method.

After this initial approach, other researchers intended to improve its performance. Criminisi et al. in [37] proposed a priority scheme to improve the filling order. The scheme priority imposed that those empty pixels on the image objects boundaries have higher priority than those pixels in flat regions. This way they prevent the problem of disconnected boundaries that was present in the Efros and Leung’s method. Furthermore, the velocity of the method was improved since instead of single pixel values, the entire neighborhood was copied. Obviously, this last improvement lead to bad reconstruction on those parts of the image with little details. Ashikhmin [38] contributed as well to improve the original method. He proposed a new technique to boost the speed of the method by reducing the space in which the algorithm looks for the most similar neighborhood. Instead of searching through all the image, the algorithm only searches between the candidates of the neighbors of \( P \) which have already been inpainted. In figure 16, the search methodology is graphically explained. L-shaped neighborhood around pixel \( P \) contains generation pixels. Each of these L pixels give a shifted pixel value (black) extracted from its original region in the input image. This shifting is computed accordingly to the pixel \( P \) original position. The results show a good visual quality inpainting and the computation speed increases considerably.
The second category of inpainting methods are the ones that are based on sparse image representations. Elad et al. [39] propose using a overcomplete dictionaries adapted to the representation of image geometry and texture and an image decomposition model with sparse coefficient. The aim of these methods is to reconstruct the inpainted region by means of combinations of vectors within specific dictionaries.

2.4.2. PDE and variational methods

The aforementioned methods seek to reconstruct the inpainting domain synthetically by copying valid patches either from the image or external dictionaries, that properly define the neighborhood of the inpainted pixel. However, there is a totally different sort of methods based either in variational principles (minimization problems) or by means of Partial Differential Equations (PDE).

Bertalmío, Sapiro, Caselles and Ballester [34] propose PDE model in the very spirit of real paintings restoration, where the restorer sought to smoothly expand the colors of the gap boundaries. Consider the inpainting problem showed on figure 17 in order to model the problem:

The arrows in figure 17 represent the direction of the vectors $\vec{V}$ in which the information to propagate does not change. This idea is expressed by the following expression:

$$D_{\vec{V}}L = 0,$$

(11)
where $D\vec{v}$ is the directional derivative in the direction determined by $\vec{V}$ and $L$ is the information to propagate inside the empty gap. This formulation can be equivalently expressed as follows:

$$\nabla L \cdot \vec{V} = 0,$$

where $\nabla L$ represents the gradient of the information to propagate. Formally, this concept is referred as isophotes (equal luminance contour). The expressions in (11) and (12) literally mean that the change of $L$ in the direction of $\vec{V}$ is zero. At this point, Bertalmío, Sapiro, Caselles and Ballester propose to solve the aforementioned problem by means of PDE. Alternatively, the direction of propagation $\vec{V}$ can be expressed as level lines, which are perpendicular to the gradient. Moreover, they propose to propagate the Laplacian representing the smoothness of the image. Both concepts are presented on the following expression:

$$u_t = \nabla^+ u \cdot \nabla \Delta u,$$

where $u$ is the solution of a time stepping method for the transport-like equation (13).

On the other hand, the inpainting problem can be solved by modeling it with variational calculus. P. Getreuer in [33] explains the classical solution of the inpainting problem by means of the Laplace equation. Let $f: \Omega \to \mathbb{R}$ be a grayscale image and let $D \subset \Omega$ the region to be inpainted. Then, $f$ is known in $\Omega \setminus D := \{x \in \Omega: x \notin D\}$ and unknown in $D$. The inpainting solution by Laplace satisfies:

$$\begin{cases}
\Delta u = 0 \text{ in } D \\
u = f \text{ in } \partial D
\end{cases},$$

where $\Delta u$ denotes the Laplacian of the solution $u$, and $\partial D$ refers to the boundaries of the inpainted region, therefore, the solution has to be equal to the input image on the boundaries. The functional that defines this problem is expressed as follows:

$$J(u) = \int_D |\nabla u(x)|^2 dx$$

PDE/variational methods have an important drawback: they are not able to reconstruct texture properly and this problem is more evident on large inpainting domains. On the other hand, patch-based methods do not perform well in the case of very sparse image where square patches cannot be found. Many studies have proposed methods that combine patch-based and PDE/variational methods in order to implement an inpainting algorithm that is able to perform well both for texture and geometric structures. An extensive survey explaining in detail the aforementioned techniques can be found on [40].
CHAPTER 3

3. DEPTH ESTIMATION FROM DUAL BRACKETING

In this section, all the details related to the implementation of the method are explained. The first part will provide information about the first step introduced on figure 6, where an overview of the projected is specified. This first step is the implementation of a new functionality on a Canon EOS 600D camera by means of the firmware called Magic Lantern. The second section of this chapter contains all the information related to the depth map computation algorithm from the modelling of the problem to the implementation of it. Finally, in the third part, the all-in-focus application is analysed.

3.1. Dual Bracketing Module Implementation in Magic Lantern

The camera modification has been performed due to the need of a specific input data to the depth estimation algorithm. In fact, the algorithm takes as input a sequence of images of exactly the same scene, but changing some camera parameters between each image. Most of the state of the art methods based on focus cues described in section 2.2, change the degree of defocusing of a concrete image region by moving the focus ring of the camera (modifying the focus distance). This process is known as focus bracketing. Others, instead, use the lens diaphragm aperture (i.e. aperture bracketing) to produce this effect and even other methods choose to fix a concrete focus distance and move an object throughout the scene. In this work, we choose to modify both the focus distance and the diaphragm aperture. In order to get familiarized with these two concepts, it is useful to review the basic camera functionalities.

3.1.1. Basic camera functionalities

In this subsection, we briefly introduce the basic parameters of a camera in order to understand the goal of implementing the proposed modification on the camera. In photography, there are three main parameters that directly affect the final appearance of the picture, specifically the exposure (i.e. the amount of brightness in the picture). These are the shutter speed (exposure time), the diaphragm aperture and the ISO sensitivity. The first two parameters directly modify the camera lens while the last modifies the image sensor of the camera. The shutter speed specifies the amount of time that the lens diaphragm is open letting the light reach the image sensor. This parameter is measured as time, meaning that a shutter speed of 1/500th of a second will let half as much light in as 1/250th. The aperture refers to the diameter of the diaphragm; the more open is the diaphragm, the more amount of light will reach the sensor. This parameter is measured by a value called f-number and it is the ratio of the camera’s focal length to the diameter of the diaphragm. The smaller the f-number, the more open is the entrance pupil (Fig. 1828). By modifying this parameter, consequently the depth of field changes too. The depth of field specifies the nearest and farthest objects that are in focus in a scene. The smaller is the pupil, the further is de distance. Accordingly, the greater the aperture, the less pixels will be focused.

28 http://img08.deviantart.net/3606/i/2011/335/2/6/f_number___aperture_by_danutza88-d4hu2bt.jpg
The last parameter is the ISO and it determines the sensitivity of the image sensor to the light. The values usually go from 50 to 25600. However, as much sensitivity, much brighter is the image, but it appears digital noise on it. The art of photography has a lot to do with the perfect control of these three parameters. The best quality (objectively speaking) is achieved with an acceptable trade-off between these parameters. Figure 19 presents a summary of the ideas exposed.

Finally, the focus distance is another key parameter to keep in mind when taking photos. It is modified by the focus ring of the camera lens and when turned, the focus plane sweeps through the scene accordingly. When moving the focus ring, the distance between the lenses that form the main lens, is modified, and consequently the image is formed in a different part of the optical axis (Fig. 2). However, we will not enter in such detail since the lens mechanism is outside the range of this project.

Some of the explained parameters are not directly modified in order to take the desired input sequence of images, however its value will determine the final result. The directly modified parameters are the focus distance and the aperture, which is the goal of the dual bracketing module. Concretely, for each focus step, \( n \) steps of aperture are taken. At this point, one can thing that this operation can be done manually. The problem is that it represents a heavy task for a human that can easily introduce artifacts between the images of the sequence, from illumination changes to dislocation of the scene due to the possible camera movement by the user, even if a tripod is used.

3.1.2. Programming Environment

In chapter 2 the programing environment has been introduced, and the tools needed are briefly specified. Magic Lantern uses arm-elf-gcc as the compiler toolchain and they recommend the 4.6.2 version. Some build instructions related on how to build the

29 https://www.newmobilelife.com/2015/02/27/take-photo-tutor/
toolchain are published on the Magic Lantern wiki, however, many problems arise due to specific and high-level parameters that must be configured. Specifically, the main problem was related to a communication breakdown between some dependencies specified on the Makefile due to the versions currently installed on the system. The version of the documentation software Texinfo was not compatible with the 4.3.6 gcc version of the compiler. Alternatively, a pre-built toolchain\(^{30}\) is provided so that to directly install it on the computer. The recommended operating system is Ubuntu Linux. After installing an Ubuntu operating system in an external bootable disk and several attempts to build the compiler, we decided that the best option would be to configure a pre-built virtual machine. This option provided us with the required tools and the desired agility needed to start implementing the dual bracketing feature.

The specific documentation to get the compiler working is provided on a Magic Lantern forum thread\(^{31}\) by one of the community developers. He proposes installing an Ubuntu virtual machine using the Virtual Box software 5.1.8 version. Several forum users provide such pre-built virtual machines, but the one we used is based on Ubuntu Mate 14.04 LTS\(^{32}\). Ubuntu Mate is a free and open-source Linux distribution and the main difference between other Ubuntu distributions is that it uses Mate desktop environment as the user interface. The next step, once downloaded Virtual Box and the virtual machine, is to configure the settings of the virtual machine. We need to create a new machine by just adding the downloaded virtual disk image and set the desired settings. Table 1 presents the configuration used in this case:

<table>
<thead>
<tr>
<th>Base memory of the motherboard</th>
<th>3328 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of processors (CPU)</td>
<td>1</td>
</tr>
<tr>
<td>Execution capacity</td>
<td>100%</td>
</tr>
<tr>
<td>Screen video memory</td>
<td>50 MB</td>
</tr>
<tr>
<td>3D acceleration</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

Tab. 1. Virtual Box configuration settings.

This virtual machine has the gcc-arm-none-eabi 4.8.2 compiling toolchain installed ready to build Magic Lantern executables. The next step is to get the source code cloning it from the Magic Lantern Bitbucket repository. Once cloned, we can start working with the latest source code version.

3.1.3. Dual bracketing implementation and camera execution

In chapter 2 it is explained the three different existing ways to develop in Magic Lantern. The first one was by directly modifying the source code and adding the new feature on the correspondent C file. Magic Lantern source code is divided into different C files and the corresponding header files (“.h” files). Each of these files contain features grouped under a common goal. For example, most of the features related with the “Focus” menu are implemented in the same file. However, other files can also contain only one single feature. The most important files are those containing global functions that are used by external files. For example, the file called “Lens” contains the methods related with the camera lens, and are able to perform actions from getting information to changing

30 https://launchpad.net/gcc-arm-embedded
parameters of the camera (i.e. aperture, shutter speed). For this last type of files, its header file works as a library from which one can call specific methods. The difficulty of using this first type of development is that one must figure out where to implement the new code as well as adapt it on the already implemented code. Unfortunately, the Magic Lantern documentation is limited and most of the doubts need to be solved by reverse engineering. This is the main reason why developing by Magic Lantern is a tough challenge that requires a lot of time to figure out how most of the code works. Moreover, since it is not an official software developed by a company or an institution, the literature is so sparse. Most of the doubts during the project has been solved by means of a personal forum thread.

The second method to develop is by means of LUA scripting. LUA programming language is useful to implement in Magic Lantern simple functionalities that do not require high computational cost. The disadvantage of using this programming technique is that most of the already implemented methods of Magic Lantern are not available from LUA. LUA module in Magic Lantern is far of being fully functional and efficient since it is a recent implemented module and its developing is quite slow. Our first approach was to use LUA since the dual bracketing feature seemed a simple coding task. However, after spending a lot of time trying to make it work, few LUA errors made us give up on this purpose. A brief review of the found handicaps is presented so that to argue why we decided to change the programming method.

One of the first approaches was to modify the focus and aperture camera parameters while the camera was recording video. This was thought in order to reduce the computational cost of the program. The idea was to avoid the picture taking process, because it implies a mechanical difficulty to the camera since the reflex mirror has to move up and down to let the light reach the image sensor each time it takes a photo. This process is known as mirror lockup and it occurs every time the shutter is pressed when the camera is not in Live View mode. Live View mode in Canon is the photography option that allows the user to see the scene in the display screen. If this option is enabled, the reflex mirror is up, meaning that when taking a photo, the mirror does not have to move. In this case, the only mechanical action is from the shutter, which also can affect the performance of the program. Since the dual bracketing takes many pictures, the picture taking process could mean a high charge on the camera processor. Another problem deriving from these mechanical actions is the fact that it could also introduce artifacts related to the vibration of the camera. Therefore, since movie recording does not imply mechanical movements of the camera, we thought it would be a good choice. After implementing some simple LUA scripts which considered movie recording, the code was not working; when run, the camera was not recording (Fig. 20). Again, it was so difficult to realize the error since it is tough to implement a code which has to be executed on an external device and poor debug tools are provided. Luckily, during these months we were able to contact one of the main Magic Lantern developers. After several contacts, we realized that it was an internal bug with the movie recording method which had to be addressed by pushing a commit to the Bitbucket repository.

33 https://www.magiclantern.fm/forum/index.php?topic=18838.0
34 https://bitbucket.org/hudson/magic-lantern/commits/all
The following approach was to use the picture taking process instead of the movie recording option. In order to understand the problem exposed, a brief and simple code explanation has to be done. The focus bracketing process has two main steps: taking a picture and moving the focus ring. These two steps are executed inside a loop so that to perform the entire bracketing. The method that moves the focus ring in Magic Lantern only works when the Live View mode is enabled. The problem was that this method was not working; the focus ring was not moving. This time we had more information about the problem. By means of a console implemented on Magic Lantern we could realize the error. It was saying that the focus ring movement method only works when the Live View is enabled. The Live view mode was enabled because the dual bracketing feature works in Live View. For this purpose, we developed a LUA script to print some information on the camera screen in order to realize what was happening (Fig. 21).

By means of two functions that provide information about the Live View state we could solve the problem. They were implemented between the picture taking and focus ring movement methods. Concretely, these functions are Boolean and return false if Live View is not enabled either running and true on the contrary. When Live View is enabled, it can be running or paused and when is running it is enabled and not paused. In figure 21, the printed information corresponds first to enabled and then to running. In this case, it can be seen that both are false, meaning that there was an obvious problem. Again, talking with one of the developers, we realized that the state information of the Live View takes some time to be updated. Since the picture taking method change the Live View state, the focus ring movement method was interpreting it as it was not running. For this purpose, a sleep function solved the error by pausing the loop few seconds until the Live View state information was updated.
This collection of problems made us decide to finally implement the dual bracketing feature by means of module programming. The modules are implemented as separate functionalities that are only used when the user loads them on the camera. Modules are located on a different folder from the source code one (Fig. 22). This fact means an advantage with respect to the first type of development because the new code does not need to be adapted to the old one, meaning that the developer has a relative freedom to think how the code will work. Moreover, all the existent external methods and features can be used simply by calling the respective header file as if it was a library. Modules need three essential files to work: a C file, a Makefile and a README file.

The explanation will start by introducing the execution of the module on the camera and how the user interacts with the dual bracketing functionality. At the same time, the code behind the interface will be explained. In this way, the user interface is presented and a better (visual) comprehension of the code can be achieved. After getting the compiled code (Nightly Builds explained in chapter 2) the next step is to install it on the camera. By copying the obtained folder on the memory card and following the installation steps introduced in chapter 2, one can easily install Magic Lantern on the camera. Initially, the module will not appear in the principal menus, because first, we have to load it. In the menu “modules” one can choose which modules to load as seen on figure 23. The procedure is simple; select the desired modules and reboot the camera.

Once the dual bracketing module is loaded, a brief review about the functionality that it implements as well as information about the interface is provided (Fig. 24). This information is written in the README file mentioned before. There, the module author decides which information provide to the user. At this point, the module is loaded in its respective principal menu and ready to be used. We decided to locate the feature on the “shoot” menu, where all the picture taking features are implemented (Fig. 25). As can be seen on figure 26, the menu has three user tasks: “start bracketing”, “step size” and
“aperture brack. range”. As seen on the attached figures, on the lower part of the screen, short information is provided at each interface step. These comments are implemented thanks to a library called “menu.h” which holds every function related with the menu implementation. In figure 25, we decided to put a warning comment which remembers that the camera AV mode is required. The AV or aperture priority mode automatically sets the shutter speed, so that to maintain a good image exposure when the aperture is changed. By means of the “property.h” library we can check if the AV mode is selected. Moreover, so that the module to work, the lens has to be in autofocus mode, because the program automatically moves the focus ring. Since we wanted a fully automatic process, this particular mode is compulsory.

When pressing start bracketing, the main function of the code, “dualBracketing()” is called. A small piece of pseudocode is attached so that to understand the procedure:

```
dualBracketing Procedure
if Live View is not running do:
    turn on Live View
end if
if focus distance is not in Near Focus do:
    while focus distance is not in NF do:
        move focus ring one step
    end while
end if
if aperture brack. range = complete do:
    apertures vector = complete apertures vector
else if aperture brack. range = half do:
    apertures vector = half apertures vector
end if
while focus distance is not HFD do:
    while apertures vector is not over do:
        set aperture X from apertures vector
        take a picture
    end while
    sleep
move focus ring
end while
```

The first thing to check is if the Live View is running. Remember that we wanted to have the reflex mirror up so that to avoid possible camera vibrations. We can check the Live
View state and turn it on by means of the libraries “lvinfo.h” and “shoot.h”. Then we make sure that the focus distance is set on the Near Focus. The near focus is the nearest distance in which the lens can focus and it depends on the type of lens used. In this project, a Canon EFS 18-55 has been used and the nearest distance in which it can focus is at 0.25 m. The focus distance is sparsely reported by “lens.h” library and the focus ring can be moved with one of its functions. Therefore, by setting the focus distance to the first step, we assure that the full bracketing range is taken. Then the dual bracketing loop is executed but first, we have to check in which mode the module is working. As seen in figure 26, the user can modify a parameter called “aperture brack. range”. This parameter let the user choose between “complete” and “half”. Complete range means that a full aperture bracketing is performed. In other words, the camera takes pictures using all the possible f-numbers. Again, it depends on the lens used. In our case, the range goes from 3.5 (most open) to 22 (most closed). In total, we use ten different apertures between the limits. On the other hand, half range only takes pictures with five different apertures. Both f-numbers are specified on the screen (Fig. 26). Therefore, before entering the dual bracketing loop, the program has to know which mode is selected. The apertures vector used inside differs from a mode to the other. In “lens.h” the possible aperture values as well as the method to change the diaphragm aperture, are provided. The aperture values are given by the exposure value (EV). This value specifies the exposure in function of the aperture and the shutter speed.

\[
EV = \log_2 \left( \frac{N^2}{t} \right),
\]

where \( N \) is the f-number and \( t \) the shutter speed. Again, in order to know which f-numbers corresponded to EV values, reverse engineering had to be done. Then, the dual bracketing is performed. The first loop moves the focus ring until the HyperFocal Distance. The hyperfocal distance is the focus distance where the camera is focusing at infinity, meaning that almost all the image is focused, thus, the depth of field is the greater one. This distance is also reported by “lens.h” library. Before moving the focus ring, the second loop performs the aperture bracketing and in each iteration, it takes a picture by means of a method also in “lens.h”. After completing the aperture bracketing loop a sleep function from “dryos.h” is applied so that to solve the aforementioned Live View problem detected while LUA scripting. Finally, the focus ring is moved one step so that to change the focus distance. This function is also called from “lens.h” and takes as input the menu parameter “step size” introduced before. The step size value specifies to the autofocus motor how far it has to move. The smaller the step size, the more pictures it will take.

During the compiling process we had some problems. Every time we compiled, the module was created fine but when searching it on the modules menu, it did not appear. We managed to solve this problem by publishing it on the forum thread that we opened. The problem was that in the modules Makefile, the dual bracketing module was not considered. This was because a “make clean” had to be performed so that to update the Makefile. Once solved, the module was ready to be used on the camera.

Initially, the depth map computation was thought to be implemented on board the camera, and many efforts were done in order to achieve this purpose. Unfortunately, due to the camera processor limitation it resulted in a heavy and inefficient task.
3.2. Depth Map Computation Method

3.2.1. New approach

This project aims to estimate the depth of a given scene following the rules of the so-called Depth From Focus theory. The relative distances of objects with respect to the camera are computed using the focus differences that occur between images when sweeping the focus plane throughout the scene. As introduced in chapter 1, the input of these methods is a sequence of images in which each of them has a different focus configuration. The focus distance in each image is controlled by the camera lens, being a mechanical process. For each pixel in an image, its focus magnitude can be computed using techniques that operate on the spatial or frequential domain, among others. Using this information, a focus function for each pixel on the scene can be computed as introduced on figure 3. This function specifies for each pixel, its focus measure in each image of the sequence. Therefore, by finding the image index on the sequence in which the focus function is maximum, an index map with the same size of the original image can be computed. This map contains the relative distances from the objects of the scene to the camera, meaning that the absolute distance value (e.g. in meters) is not computed. The real distance can be estimated by having exact information on the camera parameters and making some assumptions related to lens imperfections. The range of this work includes from the picture taking process to the depth map computation and introduces some final applications.

The main drawback of these methods relies on the image content, specifically on images with low-textured regions. The problem is that the focus measures applied through the image, does not operate well in the homogeneous regions, since they measure local frequency variations of image intensity. While regions with textured content present high frequency variations, those with flat content does not. In other words, the defocusing process applied through a flat region does not produce a significant change on the pixel values. As a result, a random focus function with no clear information about the maximum focus, is obtained. Many of the research mentioned in chapter 2 propose new focus measure operators that intent to be robust to low-textured regions. However, although some has proved to perform better, few behave in a stable and robust manner over a large variety of images. One of the main objectives of this work, is to obtain a depth map that is robust to homogenous regions. The improvement is not held to the implementation of a new focus operator, but it is introduced as a pre-processing step after the focus measure computations and before the depth map computation.

The approach that we propose is to estimate the depth by having more focus/defocus information of the scene. As it was shown in the state of the art, previous approaches were based on the variations of focus by only modifying the focus distances of the camera lens. Other methods also obtain the defocus process by varying the depth of field, as introduced in chapter 2. Our method uses both defocusing process, meaning that two different information on the focus variation are held. The dual bracketing module implemented on the camera, provides an image sequence where the images differ from focus distance and aperture. Specifically, between each step of focus, a complete aperture bracketing is performed, meaning that if the camera takes $N$ focus steps and $M$ aperture steps, a sequence of $N \times M$ images will be obtained (Fig. 27). As a result, now we are dealing with $M$ different focus functions, one for each aperture step. Therefore, a focus surface is computed for each pixel, meaning
that now we are handling a four-dimensional problem (the spatial location of pixel plus the surface measuring the focus at a specific pixel). The focus bracketing extends from near focus to far focus, while the aperture bracketing extends from small to large f-numbers, meaning that in each step, the depth of field in the image gets larger (Fig. 19). In the ideal case, the resulting focus surface is formed by focus functions that keep increasing its standard deviation as the aperture gets smaller. This is because the concrete pixel stays in focus during more focus steps. In other words, the first focus function is narrower than the last one. In those ideal cases (i.e. textured regions) where the focus measure can be well computed and a defined focus surface is obtained (Fig. 28), since now we have $M$ different maximums, the final predicted index will be assigned with more confidence and even with a better value since an average between the indices corresponding to different maximum can be made.

The focus surface in the case of a pixel within a low-textured region has not a definable behavior, resulting in very noisy plane (Fig. 28). If these noisy surfaces are detected, the distances corresponding to such pixels can be approximated by means of image processing techniques. By means of a credibility measure, the method is able to detect these cases. A detailed review of the credibility measure will be later exposed.

Accordingly, the method proposed here intends to estimate the distances within low-textured regions of the image and compute a more reliable distance for those cases in which the focus measure operator works well. The computation of the depth map is presented as an energy functional minimization using calculus of variations. As introduced on chapter 2, a functional defining the features of the solution, is proposed. By means of the functional minimization, the algorithm seeks to find the two-dimensional function (i.e. depth map) that best satisfies the introduced constraints. A complete explanation of the functional as well as its detailed development are held in later subsections. All the code has been developed using the programming language Matlab.
3.2.2. Preprocessing

In this work, the complete process from the image capturing to the depth estimation is held, meaning that real images are used as input of the method. Many state of the art methods use databases that contain synthetic sequences of images. These images are usually ideal, meaning that the problems related with camera imperfections are not taken into account. Moreover, the defocus effect in the sequence of images is also synthetically added by a concrete blur function. This is not our case, since the sequences of images are obtained using a camera and its limited lens. An example of a set used during the testing of the dual bracketing module is showed in figure 29.

The first remarkable imperfection is the difference of luminance between the images on the sequence. In section 3.1.3, during the dual bracketing module explanation, we mentioned that the camera must be set on AV mode (aperture priority) so that to assure a good performance. The reasoning is simple: since the aperture is changed in each picture, the camera has to automatically set the shutter speed in order to keep the image in an acceptable exposure. During the image capturing process, the natural scene illumination can change and therefore, the exposure (i.e. luminance) of the image is affected. AV mode should keep the same exposure across the sequence, but perfect matching is almost impossible. This work attempts to solve this problem by introducing a luminance normalization across the sequence of images. A reference luminance is taken to accordingly normalize the rest of images. Let $Y_{ref}$ be the luminance of the reference image (in our experiments will be the image corresponding to first focus distance and first aperture). Therefore, we want the mean luminance $Y_i$ of an image to be equal to the mean reference luminance. Then, a certain factor has to be multiplied to the mean luminance:

$$mean(Y_i) \cdot X = mean(Y_{ref}) \rightarrow X = \frac{mean(Y_{ref})}{mean(Y_i)}$$

Accordingly, the luminance normalization formulation:

$$Y_i(x, y) = Y_i(x, y) \cdot \frac{mean(Y_{ref})}{mean(Y_i)} \quad (17)$$

The image luminance is obtained by converting the RGB image to the YCbCr, where the first channel corresponds to the luminance. This normalization is useful in those cases where the luminance differs between images. However, when the sequence images show similar brightness, the result does not vary so much. After this normalization, all the images have the same luminance. In figure 30, an example of luminance normalization on one of the images in our database, is shown:
Another problem result of the camera lens is known as object magnification. The magnification is the concept that defines the ratio between the size of the real object and the size of its projection on the camera sensor (i.e. image). It is a function of the focal length and the closest distance at which the camera can focus. Therefore, when varying the focal length in not-fixed lens (i.e. varifocal lens), the magnification also varies and that is known as optical zoom. In conventional lens, when the focus distance is changed, the focal length slightly does accordingly. This is due to the mechanical performance of the lens optics, which indistinctly vary the focal length since it directly affects the point where the light rays converge (i.e. where the sharp/focused image is formed). Professional lens usually maintain the focal length constant along with the focus distance, as for example telecentric lenses in which the magnification is constant. For professional purposes, it is desired to completely separate both the optical zoom and focusing processes, meaning that when changing focus distance, the focal length should keep constant, or the way round. This is the case of parfocal lenses, which keep the focus distance constant along with the focal length. The magnification effect due to the focusing, is present in our images sequence and it is a problem to solve, since comparing images with different magnification, means comparing wrong pixels. In other words, the focus measures forming a focus function would correspond to measures of different parts of the image. The ideal situation is in the case that a concrete pixel along the focus stack contains information of the same exact part of the image.

Many state of the art algorithms solve this problem by means of techniques such as optical flow or image warping methods. The main goal of this kind of techniques is to align the images so that to obtain exactly the same scene in each frame of the stack. This concept is known as image registration, and it is the process of transforming sets of data (i.e images) into a concrete coordinate system. An example of an image registration performed on images of our data base can be seen in figure 31.
The top images are the original ones and the bottom are the result of the image registration. The images on the left correspond to the image taken with the first focus distance and on the right the ones corresponding to the last focus distance. Between both original cases, the magnification problem is perceivable. On the top right image, the blue object seems to be closer to the camera and greater. Moreover, the field of view has been reduced on the top-right image, meaning that the camera would not be able to capture as much content as in the case of a lower focal length. This effect can be seen by comparing both images on the left, in which the registered one has a reduced field of view in order to align the last frame.

The most popular registration methods are the intensity-based ones, that automatically aligns an image according to another image used as the reference. The algorithms usually solve an optimization problem at each step of a multi-resolution image pyramid with N levels. Optimization starts at the coarsest level and does not change to the next until the convergence is achieved. At each level, the geometrical transformation that relates both images is estimated and refined at each upper level until the full resolution one. Intensity-based methods compare intensity patterns in images by means of correlation metrics. Examples of these metrics are the well-known mutual information or means square value. On the other hand, the most popular optimizer is the gradient descent.

As introduced in chapter 2 the defocusing process violates the brightness constancy assumption meaning that the images brightness is not constant along the sequence. In other words, as the objects of the scene are blurred along the bracketing, when comparing images with the reference, the registration can be difficult since a concrete object does not remain exactly equal along the sequence. Most of the registration errors are cause of this fact. An example of the performance of an intensity-based registration method is shown in figure 32. On the left, the purple mask represents the image to be aligned and the green mask represents the reference one. The magnification problem is perceivable. On the right image, the registration has been applied. As it can be seen, both purple and green masks are almost perfect aligned, thus solving the magnification difference introduced by the focus sweeping.

![Fig. 32. Image registration using an intensity-based registration method.](image)

3.2.3. Focus measure operator

In chapter 2 we introduced the focus measure operators used in Depth From Focus/Defocus techniques. We commented on the advantages and inconvenient as well as how they are classified. We talked about spatial and frequency domain operators as well as modern measures based on deep learning. The selected criterion of sharpness should respond to high-frequency content in the image and have maximum response for
those focused regions. Therefore, the focus measure operator should maximize the high frequency variations in the images. The main focus measure used in this project is within the spatial domain operators and follows a pixel-wise measure. In pixel-wise measures the operator is computed for every pixel in the image by taking into account a small region of pixels around the desired pixel. Then a focus measure matrix $F_k$ is computed for each image $I_k$ of the focus stack, being $k$ the frame number. The values of $F_k$ in a concrete pixel $(x,y)$ along all the focus stack is the focus function explained before (Fig. 3). In this work we mainly use the gray level variance computed as the Mean Square Error (eq. 18), where $\mathcal{A}(x,y)$ is the $r \times r$ neighborhood of $(x,y)$ and $\mu$ is the mean gray level of the neighborhood.

A neighborhood of size 3x3 (i.e. $r = 3$) has been considered for this work. However, in chapter 4, a discussion of the neighborhood size is held.

$$F_k(x,y) = \sum_{(i,j) \in \mathcal{A}(x,y)} (I(i,j) - \mu)^2$$

(18)

After $F_k$ computation, a regularization is performed through the focus measure matrix. Concretely, an averaging filter is applied on each $r \times r$ neighborhood of the image. This step, is applied in order to keep the computed focus measures as much uniform as attainable so that to avoid possible wrong measures. Although this process does not largely improve the result, it can locally reduce the focus measures error. An example of focus measure matrix for a concrete pixel computed with our method is shown on figure 33.

As it can be seen on figure 33, the function along the aperture axis is not a straight line. A straight line is the ideal behavior for a given lens, meaning that the pixel remains equally in-focus independently of the aperture used. Although camera lens does not operate ideally, there are lenses that are able to decrease this effect. That is one of the factors that influences the price of a camera lens. In figure 34 three different plots representing three different type of lenses, are shown. These plots represent the focus measure along the aperture bracketing for a Sony Alpha 7 camera. In each graph, both a blue and a red curve are plotted. The red curve represents the behavior of a pixel located in the image center, while the blue line represents a pixel in the image boundaries. Both pixels are always in-focus. The horizontal axis of the plots represents the range of apertures from small f-number to greater (i.e. from small depth of field to larger). On the other hand, the vertical axis represents the focus measure. These plots reveal that the lens focus quality is approximately constant for a pixel in the center of the image. However, at the boundaries, the focus quality depends on the used aperture, existing an aperture with which the image sharpness will be maximum. This last concept is known as sweet point of a lens. As higher and straighter are the lines, the better is the lens quality. However,
other effects such as vignetting and chromatic aberration influence the performance of the lens. An example of the performance of our lens can be seen on figure 35.

![Figure 34](image1.png)

**Fig. 34.** Focus quality of lens depending on the aperture. Image extracted from "Responses Photo magazine, n°301, April 2017".

![Figure 35](image2.png)

**Fig. 35.** Focus quality of lens depending on the aperture. (Result from our lens)

There exist other possible camera limitations that could affect the performance of the algorithm, but luckily in this work its effect is negligible due to the specific image analysis implemented. An example of these limitations is the so-called vignetting. Vignetting is an image phenomenon that appears at the periphery of images as a gradually reduction of brightness with respect to the center of the image. It appears as a radial darkening towards the corners of the frame. This shadowing effect distorts the image itself by adding false light information that can alter the focus measures values, since the gray-level is modified. There are several causes of vignetting such as mechanical, optical, natural and pixel. Although the magnification of the effect differs from each type of vignetting, the pattern is the same. As pointed out before, this effect is negligible and does not represent an important problem for the depth map computation, since the focus measures are computed locally. The focus measure operator computes the sharpness of a pixel inside a neighborhood that is small enough to not keep the vignetting effect. If the neighborhood was larger, it probably will be affecting the value.

Another well-known problem in photography is the so-called chromatic aberration. This image effect is the result of a lens failure to focus all the colors to the same convergence point. Camera lens are composed by many groups of lenses, having different constructions each one. The cause of chromatic aberration is due to the lens having different refractive indices for different wavelengths of light. This problem is also negligible since the focus measure is the gray-level variance and the color information is not processed.
3.2.4. Gaussian fitting

Although a lens can precisely focus to a concrete distance, the sharpness decreases gradually on each side of the focused distance, meaning that the limits of the depth of field are not hard boundaries between sharp and unsharp. That is why the focus function does not look as an impulse function but a parabolic-like function. Although many state of the art methods use parabolic interpolation, in this work we fit the focus function to a gaussian. In figure 33, a focus measure matrix of a pixel within a textured region is shown, and it can be visually stated that it makes sense to interpolate the function to a gaussian. As the aperture decrease, the standard deviation of the gaussian increases, meaning that the pixel is focused during more frames. This is due to a larger depth of field on the image. Therefore, the method interpolates all the focus functions to the respective gaussian to obtain accurate depth estimations. The gaussian fitting used in this work performs a three point gaussian interpolation. An example can be seen on figure 36:

![Gaussian fitting example. Left: Before gaussian fitting. Right: After gaussian fitting.](image)

As it can be seen, the standard deviation of each gaussian is not computed as expected. It should be greater at each aperture step, because the depth of field increases with the decrease of the aperture. To solve any confusion, in this case, a bracketing of ten apertures has been chosen to compute the depth map, that is why the num apertures axis on the focus measure matrices goes from 1 to 10. The first aperture corresponds to the greater one, and the tenth corresponds to the closest one. The problem of the wrong standard deviation estimation does not affect in any way the result. The method does not use the standard deviation in any computation, therefore, the only requirement is to get the correct gaussian means. In chapter 4, this fact is better explained.

3.2.5. Energy functional

Once the gaussian estimation has been performed throughout all the focus measure matrices, the next step is to compute the distances for each pixel, thus the depth map. The problem is solved as an energy functional minimization by means of variational calculus. As explained in chapter 2, by means of variational calculus an extrema function that makes the functional achieve a maximum or minimum value is computed. The desired function is subject to different constrains so that to delimitate the functions space in which the solution has to be found. In this work, the function that we seek to compute is the depth map.
As described in chapter 2 (4), a functional is composed by a data fidelity term and a regularity term. The first one imposes the relationship between the desired solution and the initial data, for example in a denoising problem in which is desired to remove the noise of a given image, the final denoised image is required to be similar to the original one. However, with only this constraint there would be infinite solutions for the problem, or it may be the trivial one. That is why a second term is added to the data fidelity term, the regularity term. Let us introduce our functional:

\[ d \mapsto J(d) = \int_{\Omega} F(x, d(X), \nabla d(x)) \, dx, \]  

where \( J \) is the functional, \( d \) is the solution, thus the depth map, and \( \Omega \) the domain defined as \( \Omega = [1, N_i] \times [1, N_j] \), where \( N_i \) is the number of rows in the image and \( N_j \) the number of columns. Remember, that the expression \( F \) refers to the operations applied through the input data and its formulation depends on the problem to be solved. That is, \( F \) determines how both data fidelity and regularity terms are defined. It considers the solution \( d \) and its gradient \( \nabla d \), corresponding to the regularity term. The gradient imposes smoothness on the solution \( d \) and is introduced to reduce the set of possible solutions. In our case, the functional is defined as follows:

\[ J(d) = \int_{\Omega} |\nabla d| \, dx + \beta \sum_{a=1}^{nA} W_a (\mu_a - d)^2 \, dx, \]

where \( \beta \) is the trade-off parameter that weights the effect of the data fidelity term, \( nA \) the amount of apertures used and \( W_a \) the weights applied through the estimated gaussian means \( \mu_a \) for each aperture. These weights give more or less importance to the focus function mean according to the aperture. The first part of the sum, that has been written with abuse of notation, is the regularity term and it is known as Total Variation. Now, if we substitute the following definition of the gradient module

\[ |\nabla d| = sup < \tilde{p}, \nabla d >; |\tilde{p}| \leq 1 \]

and applying integration by parts, we can rewrite the functional in (20) using the dual formulation of the total variation:

\[ J(d) = TV(d) + \beta \sum_{a=1}^{nA} W_a (\mu_a - d)^2 \, dx, \]

where \( TV(d) \) is

\[ TV(d) = \int_{\Omega} |\nabla d| \, dx = sup \left\{ - \int_{\Omega} < d, \text{div} \tilde{p} > \, dx \right\}; \tilde{p} \in P, \]
with \( P \) being the set of vectors defined as

\[
P := \{ \bar{p} \in C^1_c(\Omega; \mathbb{R}^2); |\bar{p}| \leq 1 \}
\]  \hspace{1cm} (24)

\( C^1_c \) is the space of continuous functions having first derivate. The suitable space in which \( d \) is defined is the Bounded Variation (BV) described as:

\[
BV(\Omega) = \{ d \in L^1(\Omega; \mathbb{R}^2); TV(d) < \infty \},
\]  \hspace{1cm} (25)

being \( L^1 \) the space of integrable functions. This definition of the solution space, expands the range of functions by also keeping depths maps with discontinuities. In other words, the solution computed will respect the edges between objects at different distances from the camera. At this point, let us develop the data fidelity term of (22):

\[
\beta \int_\Omega \sum_{a=1}^{nA} W_a (\mu_a - d)^2 \, dx =
\]

\[
= \beta \int_\Omega \sum_{a=1}^{nA} W_a (\mu_a^2 - 2\mu_a d + d^2) \, dx =
\]

\[
= \beta \int_\Omega \sum_{a=1}^{nA} W_a \mu_a^2 - 2d \sum_{a=1}^{nA} W_a \mu_a + d^2 \sum_{a=1}^{nA} W_a \, dx =
\]

The first term of the integral can be omitted since the minimization is done with respect to \( d \) and this term does not depend on \( d \). Therefore, by keeping the common factor:

\[
= \beta \int_\Omega \sum_{a=1}^{nA} W_a \left( -2d \frac{\sum_{a=1}^{nA} W_a \mu_a}{\sum_{a=1}^{nA} W_a} + d^2 \right) \, dx =
\]

The term inside the parenthesis can be written as the squared difference, by adding the following component:

\[
= \beta \int_\Omega \sum_{a=1}^{nA} W_a \left( \frac{\sum_{a=1}^{nA} W_a \mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \, dx =
\]

As before, last term can be omitted as it does not depend on \( d \), resulting in:

\[
= \beta \int_\Omega \sum_{a=1}^{nA} W_a \left( \frac{\sum_{a=1}^{nA} W_a \mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \, dx
\]  \hspace{1cm} (26)
The problem can be seen in two different ways, depending if the weights $W_a$ are a function defined on $\Omega$ domain (each pixel has a different weight) or it is the same value for each pixel.

**$W_a$ not depends on $\Omega$ (it is the same value for each pixel)**

In this case, as weights does not depend on the pixels defined in $\Omega$, the first term of the integral in (26) can be taken out:

$$J(d) = TV(d) + \beta \sum_{a=1}^{nA} W_a \left( \frac{\sum_{a=1}^{nA} W_a\mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \, dx$$

By finally putting together the whole functional

$$J(d) = TV(d) + \beta \sum_{a=1}^{nA} W_a \left( \frac{\sum_{a=1}^{nA} W_a\mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \, dx,$$

we figure out that it corresponds to the Rudin-Osher-Fatemi denoising functional [31]:

$$J(d) = TV(d) + \frac{1}{2\lambda} \int |f - d|^2 \, dx,$$

where $f = \frac{\sum_{a=1}^{nA} W_a\mu_a}{\sum_{a=1}^{nA} W_a}$ and $\lambda = \frac{1}{2\beta \sum_{a=1}^{nA} W_a}$.

**$W_a$ as a function defined on $\Omega$**

In this case, as weights depend on the pixels defined in $\Omega$, the first term of the integral cannot be taken out.

$$J(d) = TV(d) + \beta \sum_{a=1}^{nA} W_a \left( \frac{\sum_{a=1}^{nA} W_a\mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \, dx$$

$$J(d) = TV(d) + \beta \int_{\Omega} \left( \frac{\sum_{a=1}^{nA} W_a\mu_a}{\sum_{a=1}^{nA} W_a} - d \right)^2 \sum_{a=1}^{nA} W_a \, dx$$

That corresponds with the ROF-like energy functional (31) with the following change of variable

$$\beta \sum_{a=1}^{nA} W_a = \frac{1}{2\lambda}$$
\[ J(d) = TV(d) + \int_D \frac{1}{2\lambda} |f - d|^2 \, dx , \]

where \( f = \frac{\sum_{a=1}^{nA} W_a \mu_a}{\sum_{a=1}^{nA} W_a} \) and \( \lambda = \frac{1}{2\beta \sum_{a=1}^{nA} W_a} \).

**Numerical scheme**

It has been proved that when \( W_a \) does not depend on \( \Omega \) the functional proposed in (20) corresponds to the Rudin-Osher-Fatemi denoising functional. Therefore, the depth map \( d \) corresponds to a denoised image, where the reference image is given by the computed expression \( f \). In order to finally compute the depth map, the obtained functional has to be minimized. Chambolle in [41] proposes a minimization scheme based on a dual formulation. He proves that the denoised image \( d \) is given by

\[ d = f - \lambda \text{div} \bar{p} , \]

where \( p \) satisfies the equation

\[ -(\nabla (\lambda \text{div} \bar{p} - f)) + \| (\nabla (\lambda \text{div} \bar{p} - f)) \| \bar{p} = 0 \]

This equation can be solved using a descent gradient algorithm

\[ -(\nabla (\lambda \text{div} \bar{p} - f)) + \| (\nabla (\lambda \text{div} \bar{p} - f)) \| \bar{p} = \frac{\partial p}{\partial t} \]

by means of semi-implicit algorithm, with the following iterative scheme:

\[ p_i^{k+1} = p_i^k + \tau (\text{div} p_i^k - f / \lambda) \]

\[ 1 + \tau \| \text{div} p_i^k - f / \lambda \| , \quad i = 1,2. \]

where as initialization can be used \( p^0 = 0 \) and \( \tau \) is a positive scalar referring to the temporal step. Chambolle proves that for \( \tau \leq 1/8 \) the method converges, but in practice the author observes that the optimum value to assure convergence seems to be \( \tau = 1/4 \).

So many weights \( W_a \) can be used in order to regulate the effect of each aperture, from sets of weights following an exponential function to others such as gaussian (i.e. giving more importance to central apertures). In this work, we have decided to use a linear weight function, giving more importance to the focus functions with greater aperture, since the gaussian is narrower and it gives a good estimated initial distance. The weight function is defined as

\[ W_a = \frac{1}{a} , \]

where \( a = \{1,2,\ldots,nA\} \).
At this point we have to reconsider the order in which the stages introduced in the project outline (Fig. 6). Initially, the inpainting model was thought to be applied after the improved depth map computation approach. In this section, we have proved that our approach is defined as a denoising applied through an image. This image is an initial depth map where the distances per pixel are computed as an averaged mean of the aperture means. Applying the denoising on this first depth map would not solve the low-textured regions problem. The wrong distances would be smoothed along these regions and the depth map would be composed by false information. The inpainting would not definitely be useful after the denoising. However, if the first step after obtaining the initial depth map is the inpainting, the low-textured regions detected by the credibility map are correctly inpainted by filling the regions with true information. Then, the denoising process is applied through the inpainted depth map in order to largely improve the current depth map.

3.2.6. Credibility map and inpainting

The task of the inpainting step, is to fix the wrong distances estimated in the initial depth map. This depth map presents, at points where there is no texture information or edges, false information. This false information is mainly due to the wrong focus measure estimation in not rough regions of the image. However, they can be also cause of other problems such as differences of luminance between images and differences of magnification. The method presented in this work is able to detect these regions where the distance has been wrong estimated. Moreover, it also can detect those parts of the image where the distance has been estimated with not so much confidence. However, the tolerance of the algorithm can be regulated with a certain parameter.

We detect these regions with possible wrong depth map estimation computing a credibility map which points them out. It is important to remark that at this step, the dual bracketing approach shows its potential. After several testings, we discovered that the fact of having more focus information could improve the depth estimation. The advantage is that the focus evolution of each pixel is evaluated for different apertures. The focus function appears to behave distinctly in each aperture. When the focus measure is reliable, the variance of the gaussian means (i.e. the argument of the focus function maximum) for each aperture, is small. On the contrary, where the focus measure is less reliable, the variance of the gaussian means is much higher. Finally, the focus measure in conflictive regions behave totally different from each aperture, thus forming the noisy surface introduced in figure 28. These three cases are shown in the following representations:

![Fig. 37. Focus measure matrices in three different types of pixel. Left: Focus measure matrix of a textured pixel. Center: Focus measure matrix of a less ideal pixel. Right: Focus measure matrix of a conflictive pixel.](image-url)
Although these three different patterns are not always exact, the general behavior is maintained across the pixels of the images. In the first case, the variation between means is small. In the second case is larger but controllable. And in the last case is large. Therefore, the standard deviation can be established as the credibility measure so that if:

$$\sqrt{\frac{1}{N-1} \sum_{i=1}^{N} |A_i - \mu|^2} > T,$$

the actual pixel is labeled as wrong. Where $A$ is the vector containing the mean of each focus function on the focus surface, $N$ is the number of means in $A$ (i.e. the number of apertures), $\mu$ the mean of the vector $A$ and $T$ the credibility threshold. After detecting the conflictive pixels, a mask with the same size of the image is computed in order to tell the inpainting model which parts have to be filled. Corresponding testing with the credibility measure is performed in chapter 4. The inpainting model used in this step is the Laplace equation described in [33] and explained in chapter 2. An example showing the performance of the inpainting method can be seen on the following figure:

![Inpainting example](image.png)

*Fig. 38. Inpainting example. Left: Before inpainting. Middle: Inpainting mask. Right: After inpainting.*

In figure 38, the image on the left is the initial depth map of a real scene. As we can see, there are regions having noise (i.e. wrong estimated distances) due to the low-texture of its content. By means of the credibility map computation, we are able to detect these conflictive regions and thus create the inpainting mask (central image). This mask has value 1 on the pixels that need to be inpainted, and 0 in those pixels where their distance has been well estimated. Once the mask is computed, the inpainting method knows which parts to fill (right image).

### 3.2.7. Application

In this work, an all-in-focus image generation algorithm has been finally implemented in order to prove one of the depth map applications. An all-in-focus image is an image having all the pixels in focus. This is a common application in the photography field, since a camera is not always able to keep all the parts of the image in focus. Photographers usually solve this problem by means of a technique called focus stacking. This technique follows the same principle as this project: find the image in the focus stack where a concrete pixel is in-focus. After taking the sequence of images, photographers obtain the all-in-focus image by means of image edition software such as Adobe Photoshop. Since we know the position in which each pixel is in-focus, an all-in-focus image can be computed. Our method forms the image by using the pixel value of the image in which it has maximum focus measure.
CHAPTER 4

4. RESULTS AND EVALUATION

In this chapter, a comprehensive review on the performance of the method presented in this work, is made. The evaluation is divided regarding both parts of the work: the dual bracketing module and the depth map computation method.

4.1. Dual Bracketing Module Evaluation

The performance of the camera module only depends on the camera hardware, and the code implemented on the dual bracketing module (i.e. software). We do not consider of much importance the performance of the camera, since it is mostly dependent on the Canon software and hardware, and the corresponding evaluation is not held in this work. However, the module code that we have implemented can have a direct effect on the general performance. Anyhow, our module only works if the camera is using the AV module (Aperture Priority), therefore, the exposure of each image will depend on how well the automatic exposure algorithm of the camera works. If the difference of exposure between the images of the sequence is large, the depth map computation method will correct these variations during the preprocessing stage. Furthermore, we choose to take the pictures using the S3 configuration. This configuration takes pictures with a resolution of 720x480 pixels. We have considered this image size in order to reduce the computational cost of the depth map method. The ideal behavior of the module is the one that entirely performs both the focus and the aperture bracketings in the shortest possible time. However, the spent time will not affect the final quality of the depth map. Accordingly, the evaluation of the dual bracketing module will be made by considering the total time spent to obtain the image sequence as well as the number of pictures taken.

The number of pictures on the sequence as well as the amount of time, depend on the dual bracketing module configuration used. As introduced in chapter 3, the module allows to modify two different parameters: the step size of the focus bracketing and the aperture bracketing range. The first parameter allows the user to choose between 1, 2 and 3 as a step size. This size tells the autofocus motor how much has to move the focus ring; if the step size is 3, the sequence of images will be shorter than if the user choose 1 as the step size. The autofocus motor is a so-called stepper that divides a full rotation into a number of equal steps. As explained in chapter 3, the focus bracketing is performed until the hyperfocal distance is reached. In other words, the stepper motor keeps turning until the hyperfocal distance. Since Magic Lantern cannot obtain information about the exact position of the stepper at a concrete time, the focus distance is used to specify the start and end positions of the motor rotation. Furthermore, the focus distance cannot be computed with total precision, since it is a value that depends on variable camera parameters, such as focal length, circle of confusion, the image exposure or the f-number. Therefore, since the module uses the hyperfocal distance as the last focus position, the number of images can vary between different executions of the dual bracketing module. Moreover, the module allows to choose between five (half range) or ten (complete range) aperture steps. If the module, from one execution to another takes for example two more focus steps, the image sequence will end up having ten or twenty images more. On the other hand, the amount of time that takes one execution, also depends on the used
configuration of the module. Obviously, if the user chooses 1 as the step size and the complete range of apertures, the module will need more time to take all the pictures than if a step size of 3 and the half aperture range have been selected. In addition, the amount of time also depends on the scene light. If it is a well-lit scene, the camera will need a greater shutter speed than if the scene has bad illumination. In other words, in those cases where we have a bad illumination, the camera will keep the shutter opened during more time so that to capture the needed light. That is why when the dual bracketing module is running, when it is taking pictures with small apertures, the camera takes much time to capture the photo.

In the following table, an evaluation of the module is performed, based on the number of pictures taken and the amount of time needed. The step size of 1 has not been considered because the resultant sequence contains more images than the ones needed to compute the depth map.

<table>
<thead>
<tr>
<th>Step size</th>
<th>Number of images</th>
<th>Amount of time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;500</td>
<td>&gt;10</td>
</tr>
<tr>
<td>2</td>
<td>330-350</td>
<td>5-7</td>
</tr>
<tr>
<td>3</td>
<td>60-80</td>
<td>1-3</td>
</tr>
</tbody>
</table>

Tab. 2. Dual bracketing module performance. The number of images as well as the execution time with respect to the step size, are specified.

The data in the above table corresponds to the case in which a complete aperture bracketing has been performed. When a half aperture range is carried out, it is approximately half the above data, both the number of images and the amount of time. The number of images is given with intervals, because of the hyperfocal distance problem mentioned above. However, the module has shown to have a variation of one or two focus steps. On the other hand, the focus step is also given with intervals because depending on the scene illumination, the total spent time will be longer or shorter. The ideal performance will be the fastest and the inalterable one, meaning that the same numbers of photos are taken between executions and using the less time possible. However, these disadvantages do not affect the depth map computation.

4.2. Depth Map Computation Method Evaluation

The depth map computation method will be evaluated in terms of how well the relative distances are estimated. As introduced before, this work does not attempt to compute the real (i.e. absolute) distances between the scene objects and the camera. Therefore, the evaluation is mainly based on how well the depth map differentiates between near and far objects in the scene. In addition, this work attempts to solve the distance estimation problem when evaluating low-textured content of the scene. Therefore, the quality of the depth map will be also measured depending on the ability of the algorithm to detect these conflictive regions and later estimate the approximate distance. During the depth estimation method explanation in chapter 3, we proved that the proposed functional that define our approach ends up representing a denoising problem. Accordingly, a final denoising stage is performed to refine the inpainted depth map. Then, the evaluation also considers the performance of the denoising algorithm and how much improves or how much deteriorates the inpainted depth map. Since the input of the algorithm is obtained by means of a conventional camera, there is no limit regarding the scenes to be analyzed. Many of the Depth From Focus state of the art methods only apply their method in
controlled scenes and usually ideal, having textured content. Accordingly, during the evaluation process, the depth map has been computed for different scenes, ranging from large spaces to small and controlled scenes. In this way, we get a wider knowledge of the general performance of the method. The depth map method is composed by different image processing algorithms such as inpainting and denoising and the methods that work with images are usually content-dependent. For this reason, many parameters can be changed, from the size of the focus measure window, to the threshold of the credibility map. Furthermore, other variables can be contrasted such as the number of pictures on the sequence, the alignment of the images and the luminance normalization. In addition, a comparison between the simple version of the depth estimation algorithm and our method is carried out. The simpler version of the method is the one that only uses a focus bracketing image sequence as input. First of all, a general review of the method without entering in so much detail, is made. This first evaluation is made in order to know the limits of the method regarding the type of scenes that it can take as input as well as the range of distances that it can detect.

4.2.1. General review

All the parameters and configurations of the method used in this first evaluation are the ones that show to perform better. They are going to be lately discussed when the corresponding parameter is analyzed in detail. The first testing made is the one corresponding to an ideal scene where all the content is textured (Fig. 38).

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Input Image" /></td>
<td><img src="image2.png" alt="Output Images" /></td>
</tr>
</tbody>
</table>

**Tab. 3. Ideal scene depth map. Top row: Scene setting as well as the camera view. Bottom row: Initial depth map, Inpainted depth map and Denoised depth map, respectively.**

In this case, the method is able to successfully estimate all the distances since the gray level variance operator is able to compute a reliable focus measure throughout the textured image. An example of the resultant focus measure matrix for a textured pixel is
shown on figure 39. Although the obtained initial depth map can be considered as a good solution, it presents some wrong estimated distances. Concretely, in the line that separates the floor from the wall. The method has estimated close distances instead of the far distances that really correspond. Therefore, it is a good first chance to prove the performance of the credibility map and the inpainting method. This low-texture case is particularly easy to solve, since the true distances are the ones corresponding to the surrounding regions. Therefore, the inpainting method will successfully diffuse inwards the correct distances as it can be seen in the second depth map (Tab. 3). Finally, the denoising step hardly smooth the depth map, solving all the possible isolated wrong distances and making the result resemble to a depth map obtained using an active system for depth estimation.

The following analyzed scene is the same ideal scene already commented, but adding a low-textured object. The goal of this test is to expose the algorithm to a low-textured scene component. The algorithm is expected to successfully perform on the textured regions, and badly on the homogeneous parts. If the algorithm is not able to detect the undesired distances, they will be conserved and the denoising method will expand the wrong information across the image domain.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Input Image" /></td>
<td><img src="image2.jpg" alt="Output Images" /></td>
</tr>
</tbody>
</table>

Tab. 4. Ideal scene with low-textured regions depth map. Top row: Scene setting as well as the camera view. Bottom row: Initial depth map, Inpainted depth map and Denoised depth map, respectively.
This is a good example to understand the problem on regions which do not present visibly rough surfaces. This is the case of the introduced box, which its surface homogeneously reflects the light and thus, the image intensity value remains invariable from one pixel to the next. The focus measure operators are based on local contrast, and the homogeneous regions do not have contrast, independently if they are focused or not. At the contrary, when a surface is textured, the intensity values vary drastically and unpredictable between near pixels. This kind of light reflectance is known as specular or diffuse and the image sensor perceives it differently depending on how it is focused. A particularly homogeneous region is the white surface of the box. In the initial depth map, the gray level variance operator has not been able to successfully measure the focus across the bracketing, because the contrast remains invariable across the sequence. This effect can be seen on the box white region in figure 41. An example of the gaussian fitted focus measure matrix of one of the pixels within the white region of the box is shown on figure 40:

![Focus measure matrix of a pixel within the white region of the box. There is no relation between the focus functions maximums.](image)

As it can be seen, the method estimates random distances between each aperture. Due to the large variance between the gaussian mean of each aperture, the credibility measure is able to detect that the estimated distances in the white regions of the box are not reliable. However, it is interesting to point that there are regions of the same box in which the distances have been well computed. This is due to the color variations inside these regions, that is, the tracings in the box. Different colors, produce different reflections since they are characterized by different spectrum frequencies, so the edges between different colors are a good point where measure the focus. An example of one of this region along with its corresponding focus measure matrix is shown on figure 41:

![Defocus effect on a drawing and its corresponding focus measure matrix. The bottom representation proves that the focus measure operator also performs well through the boundaries within low-textured regions.](image)
As proved in the focus measure matrix representation, the gray level variance successfully performs through this kind of content. This is why in the initial depth map on table 4 the distances around this kind of content are well estimated. However, inside the drawing itself, the focus operator fails, because the window is not large enough to capture these variations. On the other hand, the white region of the box is visible wrong estimated and the credibility measure detects the large variance of the means. Later, the inpainting inpaints these regions with the information in the surroundings. Unfortunately, there are few wrong estimated distances within the box, that are not detected because they are under the credibility threshold. The denoising algorithm, then, expands the false information but the method still has been able to compute an acceptable depth map. Another interesting representation so that to evaluate the depth map is the presented in figure 42, in which the depth has been plotted in the three-dimensional space.

![Fig. 42. 3D representation, corresponding to the second analyzed scene.](image)

The next scene introduces a new problem that is due to a camera limitation and it is only solvable by using another type of camera gear. In this case, a less controlled scene, meaning that it has not intentionally set, has been photographed. There are both textured and not textured objects, but distributed in different parts of the scenario. The method is expected to work in the same way that in the second scene commented. The low-textured regions will be detected by means of the credibility measure and the inpainting method will solve for them, while in the textured regions the distances will be correct.

<table>
<thead>
<tr>
<th>Input</th>
<th><img src="image" alt="Input Image" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td><img src="image" alt="Output Images" /></td>
</tr>
</tbody>
</table>

*Tab. 5. Unplanned scene depth map. Top row: Camera view. Bottom row: Initial depth map, Inpainted depth map and Denoised depth map, respectively.*
This particular scene let us realize two facts that initially where not taken into account. The first one and less important, is that when analyzing the scene, the coach of the background was labeled as a low-textured region and thus, we expected the method to fail. Unexpectedly, the method was able to estimate the correct distances. This is a prove that the visual not rough surfaces, sometimes can contain the necessary contrast so that the focus operator to work well. The second detected problem can be visually perceived on the initial depth map. The method fails within the closer distances to the camera. As one can see, the near regions present certain noise (i.e. wrong distances) although containing texture. This is due to a camera limitation: the closer distance at which the lens can focus. This is one of the main characteristics of the lens and usually is directly proportional to the price. The lenses that are able to focus at very close distances are usually desired in macrophotography, in which very small subjects are close-up photographed. This problem was not perceivable on the already mentioned scenes because the camera was a little bit raised from the floor, while in this case, the lens is totally staying on the floor. The Canon EFS 18-55 lens that we have used in this work are able to focus at a minimum distance of 0.25 meters. This distance is not measured from the end of the lens, but from the image sensor location, therefore, the object can be placed in a shorter distance than the specified. The conclusion is that the range of distances that we can measure is lower limited by the lens used. The focus measure matrix that characterizes these close regions is shown in figure 43:

![Fig. 43. Focus measure matrix of a pixel closer than the minimum focus distance. As well as in the case of a low-textured pixel, the focus functions maximums do not present any relation.](image)

The focus measure operator behaves similarly to the low-textured regions, therefore they can be perfectly recovered, since the means are largely separated. Accordingly, there is also an upper limit for the range of distances that can be measured. In this concrete scene, there is no evidence of this fact, since the depth map seems to differentiate between far distances. However, there are scenes where from a certain depth, the distance assigned is the same for all the pixels. This distance is known as hyperfocal distance and has been introduced before. When the camera is focusing at this particular distance, all the pixels from half of the distance to the infinity are in focus. In other words, when this distance is reached, there is no focus difference between far objects. Then, the focus function for these distant pixels will be the same. Therefore, our method will perceive the objects farther from this distance at the same depth. In conclusion, it exists and upper and a lower limit for the focus distances, and is determined by the camera lens. The fact that our method is dependent on the focus distance range, constraints the kind of scenes from we can extract depth information. A fast example is shown on table 6 in which a scene has been intentionally set, to test and understand this problem. In this case, an entirely textured surface containing distant objects is analyzed. The goal of this test is to demonstrate that our method is constrained by the already explained range of distances. For this example, only the denoised depth map is shown.
The objects in this scene have been intentionally located at far distances in order to understand and prove the aforementioned problem. Concretely, they have been placed far from the hyperfocal distance. The hyperfocal distance can be computed by means of the following formulation:

$$H = \frac{(focal \ length)^2}{N \cdot CoC},$$

(38)

where $H$ is the hyperfocal distance, $N$ is the f-number defining the aperture and $CoC$ is the Circle of Confusion. All these parameters are known for each picture taken. The perception of a concrete object being in focus is a subjective measure that varies between humans. When a point is in focus, the human visual system perceives it as a sharp defined spot, but as the camera changes the focus distance, it starts blurring. The circle of confusion is a subjective measure of the largest circle diameter that is indistinguishable from a point. Although it is subjective, in full-frame 35 mm still photography is usually chosen to be about $1/30$ mm. In our case, the sequence of images is taken with fixed focal length, and constant across all the scene commented here. We set it at 18 mm. However, as we perform the dual bracketing, the f-number varies between images, meaning that the hyperfocal distance varies for each of the images in the aperture bracketing. Following the formulation in (38), the minimum hyperfocal distance from which objects will not be distinguished is the one corresponding to the largest f-number, thus the closest aperture. In our case, the largest f-number is 22, thus an hyperfocal distance can be computed. By substituting the data in (38) we obtain that the minimum distance is 44.18 cm. Finally, a focus distance range of [25 to 44.18] cm can be established. This interval specifies the range of distances in which scene objects have to be located in order to be able to compute a successful depth map.

As it can be seen on the scene, both boxes are relatively distant from the camera. This distance consequently affects the final depth map. The denoised depth map successfully estimates the distances of the scene until the hyperfocal distances. The first box was located intentionally exactly at the hyperfocal distance, while the other box was placed farther. As a result, the distance of the first box is sparsely detected, but the other box is completely mixed with the background. The corresponding focus measure matrices should follow a gaussian function until the hyperfocal distance is reached in each of the apertures. The hyperfocal distances on the first apertures should be farther than in the last ones. A focus measure matrix of a pixel within the first box after being gaussian fitted is shown on figure 44:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
</table>

Tab. 6. Depth map of a scene presenting the hyperfocal distance problem. The depth estimation method cannot accurately approximate the distances of the background objects.
The conclusion obtained after exposing the focus distance range constraint is that our method is only useful for close-up scenes in which objects are near the camera. Accordingly, it totally fails when evaluated on large spaces scenes. An example of this concrete evaluation can be seen on table 7:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="input_image.png" alt="Image" /></td>
<td><img src="output_image.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Tab. 7. Large space scene depth map. Since objects are farther from the hyperfocal distance, the depth estimation method is not able to compute reliable distances, and thus, a bad depth map is obtained.

The simpler approach of this work is the one that only takes a single focus bracketing as input of the method. The code works exactly in the same way than ours, but it is not able to detect the low-textured regions and consequently, not able to implement the inpainting method to correct the wrong estimated distances. Once computed the initial depth map, the denoising is applied through it. As mentioned above, the denoising step will expand the false distances, thus providing a wrong depth map. The scene chosen to perform the comparison of both methods is the second that has been described in this subsection (Tab. 4). Since it contains both textured and homogeneous surfaces, an efficient evaluation can be achieved. The depth map corresponding to the simpler approach is shown on table 8:
Although the distance in textured regions is successfully estimated, the wrong distances are expanded along the domain, thus creating a wrong depth map. If we compare the result with the achieved using the dual bracketing one, the initial depth map looks less noisy than ours. The fact that the final distance is a ponderation between the different aperture means, sometimes imply that the distances within a region at a certain depth, slightly vary between near pixels. The expected behavior in this case would be to obtain exactly the same distance in these pixels. This is why the dual bracketing is more prone to noise.

4.2.2. Preprocessing

Once the limits of the method have been exposed and its general performance has been introduced, the next step is to evaluate in detail each stage of the depth computation in order to understand and compare its effect in the final depth map. In this subsection, the preprocessing applied through the image sequence before being analyzed, is evaluated. As explained in chapter 3 our method takes into account two preprocessing steps: luminance normalization and image registration. Therefore, the main goal of this subsection is to understand how useful is the preprocessing stage.

The luminance normalization has been implemented in order to solve those cases where large variations of exposure has happened during the picture taking process. We have to keep into account that the camera is in AV mode, meaning that the shutter speed is automatically computed, and it can significantly vary between pictures, due to scene illuminations changes. Nevertheless, the luminance usually is quite constant through all the sequence and the improvement between a luminance normalized sequence and a not normalized one is small. However, it is worth it to show the difference. In table 9 this difference is shown:
The differences between both results are really small in this case. There are small regions that seem to have less noise as for example the front surface of the bigger box. The explanation to this little difference is the following: the focus measure operator used, measures the local contrast within a small region of pixels. In principle, variations of the luminance of these regions should not change the value of the focus measure since it is locally measured, and the gray-level variance should stay invariable across luminance variations. However, when the exposure change is perceivable, the pixels value can be largely affected and in consequence alter the focus measure.

On the other hand, another important preprocessing step is the image alignment. As explained, during the picture taking process, the camera is completely stationary and any movement between images is expected. However, the magnification problem explained in chapter 3 produces a scale difference between images of the same focus bracketing. In order to first address this problem, we decided to use the image edition software Adobe Photoshop before implementing an image registration code in Matlab. The platform incorporates an option that automatically aligns a stack of images. Thus, an aligned sequence of images was obtained and the magnification problem was almost solved. Unfortunately, when we applied the depth estimation method, the results weren’t better. As introduced before, the defocusing process apply a certain degree of blur at each focus step, so the objects appearance is not identical across the sequence. This is why registering

---

**Tab. 9. Luminance normalization effect in dual bracketing method.** The contrast between the depths maps resulting from luminance normalized sequence with respect to those from a not normalized one, is shown.
is especially difficult in this case. The Photoshop output sequence was quite acceptable, however, it introduced other geometric differences between images, such as rotation or translation. Although the magnification problem was solved, these little differences meant a large error for the depth map computation. In table 10, a comparison between the depth map of an aligned sequence and the one corresponding to a magnified sequence, is shown.

<table>
<thead>
<tr>
<th>Not registered sequence</th>
<th>Registered sequence</th>
</tr>
</thead>
</table>

*Tab. 10. Image registration effect in dual bracketing method. The table shows the difference between a depth map resulting from a registered (bottom) sequence and from a not registered one (top).*

As might be seen, the alignment does not suppose an improvement in the depth map. Obviously, the noise that appears on the bottom of the first depth map due to the minimum focus distance, has disappeared because of the cropping during the registration process. However, other distance errors emerge due to the little rotation and translations that have appeared during the alignment. Since, the magnification effect does not seem to heavily affect the depth map computation, we decided not to register the images sequence. Furthermore, the register process supposed a huge computational cost of about 8-10 minutes more above the total time. The depth maps presented on the general review have been computed from a not registered sequence of images.

4.2.3. Focus measure matrices

In this section, the effect of a concrete parameter used during the computation of the focus measure for each pixel, is explained. On the other hand, the gaussian fitting performed through the focus measure matrices is analyzed in detail.

**Window size**

The focus measure operator used in this work is the gray-level variance and it is applied as pixel-wise measure. Remember, that in pixel-wise measures the operator is computed for every pixel in the image by taking into account a small region of pixels around the desired pixel. Then a focus measure matrix $F_i$ is computed for each image of the focus stack. The size of this small region of pixels is manually determined and it is usually
called window size. The window size should be large enough to capture an accurate measurement of the focus level and small enough to only include regions with a similar focus degree. If for example the operator is measuring the focus amount on an edge using a large window size, the difference between both distances (i.e. the one from the object edge and the one from the background) will be less abrupt. In other words, the edges will be smoother than if a small window size is chosen. However, with a small window size, the operator is taking less pixel values to compute the focus. This can give a bad estimation of the focus measure, because not enough information is kept. In table 11, three different depth maps using three different window sizes, are shown:

<table>
<thead>
<tr>
<th>Window size of 15x15 pixels</th>
<th>Window size of 7x7 pixels</th>
<th>Window size of 3x3 pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Depth Map" /></td>
<td><img src="image2.png" alt="Depth Map" /></td>
<td><img src="image3.png" alt="Depth Map" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Depth Map Inpainted" /></td>
<td><img src="image5.png" alt="Depth Map Inpainted" /></td>
<td><img src="image6.png" alt="Depth Map Inpainted" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Depth Map Denoised" /></td>
<td><img src="image8.png" alt="Depth Map Denoised" /></td>
<td><img src="image9.png" alt="Depth Map Denoised" /></td>
</tr>
</tbody>
</table>

*Tab. 11. Comparison between different window sizes. The window size decreases from top to bottom rows. The larger the window size, the diffuser the depth map.*
The results are largely better when using a window size of 3x3. It is easy to see that the other proposed window sizes give bad depth maps. As mentioned before, when the window size is too large, the operator keeps too many pixel values, meaning that it possibly can take multiple regions with different focus degree under the same window. In consequence, the distance resolution is reduced and edges are not well preserved. By saying that the distance resolution is reduced, we mean that the distance assigned to near pixels is similar although in reality are located on different depths. Moreover, when the window size is large, the magnification problem is more perceivable, also affecting the shape of the scene objects. The direct and bigger consequence of these problems, is that the wrong distances that the credibility measure cannot detect, now have more presence in the depth map. Therefore, the inpainting method expands this abundant false information and a worse depth map is obtained. Nevertheless, when a small window size is used, the edges are preserved, the magnification problem is almost unperceivable and the wrong distances are more concentrated, thus they can be better restored. During the general review, a window size of 3x3 has been used to compute all the depth maps.

Gaussian fitting

Each focus measure matrix is composed by as many focus functions as apertures used during the dual bracketing. In chapter 3, we explained that by studying the behavior of the focus functions that compose the focus measure matrices, it is reasonable to model them as gaussian functions using a gaussian interpolation algorithm. This assumption is often applied in shape-from-focus methods [11]. In this way, we get rid of the noise that can affect to the final estimated distance because we operate with well-defined functions with a differentiated maximum. Many examples of the gaussian interpolation algorithm has been showed during this chapter, and is evident that the standard deviations that describe the gaussians are not well modeled. We do not give much importance to this problem, because the standard deviation value is not used to perform any computation. However, the gaussian mean is expected to be perfectly estimated, because it will determine the entire performance of our method. For example, it is implicitly used to compute the initial depth map and a key component to detect the low-textured regions, as explained in chapter 3. In figure 45 an example of the gaussian fitting effect on a focus measure matrix, is shown. Moreover, in table 12, the effect of using and not using gaussian fitting is contrasted:

![Fig. 45. Gaussian fitting through a focus measure matrix. Left: Focus measure matrix without gaussian fitting. Right: Focus measure matrix with gaussian fitting.](image)
When the gaussian fitting is not performed, the method computes the maximum of each focus function in the focus measure matrix and takes the corresponding index in the focus stack, as the distance for this particular aperture. Once the maximum for each aperture has been computed, the initial depth map is computed by means of a weighted mean between the different maximums, as introduced in chapter 3, during the energy functional explanation. The problem is that due to the not ideal performance of the focus measure operator, the focus functions usually have the following shape:

![Image of focus functions with and without gaussian fitting]

Tab. 12. Illustration of the gaussian fitting effect. Below maps with gaussian fitting are more robust to the noise resulting from depth estimation. On the other hand, the top depths maps without gaussian fitting present some noise.

Fig. 46. Gaussian fitting through a focus function. Left: Gaussian interpolated focus function. Right: No gaussian interpolated focus function.
However, as it can be seen in table 12, the gaussian fitting makes a pretty difference in the result. The key fact is that with the gaussian fitting, the index in which the focus function is maximum can vary. Due to the variation in magnification, the focus function may be multi-modal with one strong peak and one or more weak ones, and in consequence, the gaussian fitting can differ between these. For example, in figure 46, the correct distance is the one corresponding to the second maximum, thus, without gaussian fitting, the index corresponding to the first maximum (num focus 27) will be chosen as the correct distance. However, the gaussian fitting has proved that is able to compute a mean (num focus 24) that better represents the second maximum. Therefore, the improvement added by the gaussian interpolation is the better distance approximation. This can affect to the all-in-focus image, because depending on the distance estimated, the image will be composed by one or another pixel value.

4.2.4. Depth map weights

During the energy functional explanation, the concept of the initial depth map was introduced. The data fidelity term of the functional (39) seeks to preserve the general behavior of the data, in this case the means of the focus functions $\mu_a$. However, our approach was to weight the importance of each mean in the focus measure matrix, because they can provide different information. This is why (39) has a multiplier factor $W_a$.

$$\beta \int_\Omega \sum_{a=1}^{nA} W_a (\mu_a - d)^2 \, dx \quad (39)$$

In chapter 3, we explained that many types of weight can be applied through the data, but after analyzing the behavior of the focus functions at each aperture, we decided to use a linear weight function. This function gives more importance to the focus functions with greater aperture, since the gaussian is narrower and it gives a good estimated initial distance. In table 13, a test is performed in order to realize the effect of the weighting function.
The larger difference is perceived in the initial depth map, where in the first case the noise is more abundant. This is because, the focus measure usually fails in the smaller apertures, and since there is no weighting, the final distance is more influenced by the errors.

4.2.5. Credibility measure

The method presented in this work is able to detect the regions where the distance has been bad estimated. Moreover, it also can detect those parts of the image where the distance has been estimated with not so much confidence. However, the tolerance of the algorithm can be regulated with a certain parameter, the credibility threshold. Since, the focus function appears to behave distinctly in each aperture, in chapter 3 we were able distinguish between three different focus measure matrices. When the focus measure is reliable, the gaussian means (i.e. the argument of the focus function maximum) for each aperture, appear to be almost the same. In those cases where the focus measure is less reliable, the gaussian means are not exactly the same. Finally, the focus measure in conflictive regions behave totally different from each aperture, thus forming a noisy surface. A credibility measure can be stablished in order to detect these three cases. In this work, we use the standard deviation between the gaussian means. In the first case, the variation is small. In the second case is larger but controllable. And in the last case is large.

Many tests were performed in order to find the threshold that was able to keep out as much wrong estimated pixels as possible, but also keeping the acceptable ones. The variance of the means from multiple focus measure matrices was computed until find the better value. Finally, a value of 1 was estimated. If the credibility measure is increased, more pixels are considered as well estimated and more noise is present in the final depth map. On the contrary, if the threshold is decreased, the variance is expected to be smaller and thus, more pixels are considered as wrong estimated. In table 14, an example with two different credibility measures, is shown.

Tab. 13. Illustration of the weighting effect. The contrast between a weighted depth map (bottom row) and a not weighted one (top row), is shown.
The effect of both credibility measures is better seen on the inpainted depth map. For a credibility threshold of 0.2, much pixels are considered as wrong estimated, and thus inpainting method tries to recover more pixels. On the other hand, when the credibility threshold is 5, the inpainting method is applied to few pixels and the low-textured regions are lesser corrected.

4.2.6. Denoising parameter

The denoising method used is work, is the one proposed by Chambolle in [41]. As input argument, the algorithm allows to choose the value of the trade-off parameter $\beta$ in (39). This parameter weights the effect between the regularity term and the data fidelity term of the functional. For example, if the data fidelity term has greater weight, the denoised depth map will be more similar to the input data. On the contrary, if the regularity term has a greater importance, the result will be smoother with respect to the input data, and more different will be. In table 15, two different values of the trade-off parameter have been tested:
As mentioned the denoising trade-off parameter regulates the relation between the regularity and the data fidelity term. In this case, the greater the trade-off parameter, the greater weight has the data fidelity term, and the more similar is the denoised depth map to the inpainted depth map.

4.2.7. Number of images in the sequence

During chapter 3 we explained that the dual bracketing module in Magic Lantern, allows to modify two different parameters. Both change the number of pictures of the final images sequence. The first is used to decide how many focus steps are taken and the other to decide between five or ten aperture steps. In the first section of this chapter, we mentioned that the habitual module configuration (34 focus steps and 10 aperture steps) captures 340 pictures. In this subsection, we evaluate the depth estimation method by reducing the number of pictures. In the first test, an aperture bracketing of five steps has been chosen, while in the second test, we have reduced the number of focus steps to the half. In both cases, one hundred seventy images compose the sequence.
In the first case, the result does not largely vary with respect to a depth map computed with a sequence containing 10 aperture steps. Although this module configuration gives an acceptable depth map, the result is better with more images. On the other hand, this is not applicable to the case in which the focus bracketing is reduced to the half. In this case, the depth map is unacceptable.

4.2.8. All-in-focus image

Once computed the depth map, apart from knowing the relative distances of the scene objects with respect to the camera, we also know, for every pixel, in which image of the stack they have maximum sharpness. This information can be used to compute an artificial image in which its pixel values correspond to the ones that are in focus, therefore obtaining an all-in-focus image. The all-in-focus algorithm choose, for every pixel, the image in the corresponding estimated focus index. From all the possible images in the aperture bracketing corresponding to this focus index, the one with maximum focus measure is selected. The all-in-focus images presented here, are the ones corresponding to all the scenes analyzed during this chapter. The results are shown in the following table:

Tab. 16. Depth map computed using less images in the sequence. The difference between reducing the aperture steps or the focus steps, is illustrated. Top row: Less aperture steps. Bottom row: Less focus steps.
A successful all-in-focus image can be computed for all the presented scenes. As it can be seen all the parts in the images seem to be in focus. However, the closer parts of the image are not possible to recover, because of the minimum focus distance allowed by the camera lenses. Although the result is acceptable, the little details such as the letters or the borders are not very well reconstructed. This is because the magnification problem discussed above.
CHAPTER 5

5. CONCLUSIONS AND FURTHER WORK

5.1. Conclusions

In this work, the depth estimation problem based on Depth From Focus techniques, has been studied from a variational point of view. A new approach to compute the relative distances between objects of a scene from a focus and aperture bracketing by means of an energy functional minimization has been proposed. This depth map computation method is able to solve the main problem of state-of-the-art methods: the weak performance of the focus measure operators in low-textured regions of the scene. The sharpness of image pixels is quantized by means of these operators. This work does not attempt to improve the performance of focus measure operators, but to propose a procedure to detect the wrong estimated distances due to the low-textured regions problem. This procedure uses inpainting techniques to recover the distances specified by a credibility map. Moreover, since our project takes into account the entire process, from the image capturing to the depth estimation, the firmware of a Canon EOS 600D camera has been modified in order to obtain the needed images sequence input to the depth estimation method. By means of the add-on Magic Lantern framework, we have been able to implement a new functionality for the camera, that we called “Dual Bracketing”. This feature allows us to perform the focus and aperture bracketing to obtain the desired sequence of images. Furthermore, an all-in-focus image computation method has been implemented so that to show one of the depth map applications. Finally, the depth estimation method has been tested on several images sequences representing different scenes, obtaining promising results.

In a detailed way, the main theoretical and practical contributions of this degree final project are:

- A new Magic Lantern feature for Canon EOS 600D cameras that allow the user to automatically take a focus and aperture joint bracketing. The procedure of the dual bracketing method is the following one: for each focus step, the camera performs an entire aperture bracketing of length specified by the user. The result of this new functionality is the input sequence of the depth map computation method.

- A new approach to depth estimation from focus techniques robust to low-textured scene content.

  - The definition of a new energy functional to solve the depth estimation problem based on variations of focus and aperture on the input sequence of images.

  - The description of relationship between the energy functional proposed in this work for depth estimation and the Total Variation dual formulation of the Rudin-Osher-Fatemi denoising model. This relationship is used to efficiently minimize the proposed energy functional.
Computation of a credibility map to solve the wrong estimated distances by means of inpainting techniques.

- An extensive evaluation of the depth estimation model in real scenes including low-textured content.

- The implementation of an all-in-focus image computation algorithm that estimates an artificial image composed by the sharpest pixels in the images sequence, thus creating an image where all the pixels are in focus.

5.2. Further Work

There are several lines that can be explored in order to improve both the dual bracketing module and the depth estimation model that we presented in this work:

- The user-interface experience on the dual bracketing module. The code could be improved so that to allow the user to concretely choose the number of pictures to be taken. Moreover, the f-numbers should be specified by the user.

- Optimize the dual bracketing code so that to reduce the execution time.

- Improve the preprocessing stage in the depth estimation method. A better luminance normalization algorithm could be implemented, as well as an image registration method able to align images having different blur degrees. Moreover, the references images in both cases could be automatically selected by the method.

- Implement a better gaussian fitting algorithm for the focus functions. The one used in this project uses only three points to interpolate the function. Although it provides us the desired low computation cost, it fails on estimating the gaussian standard deviation. A more efficient method could be implemented so that to obtain a reliable gaussian function. The standard deviation could be used in so many ways in this work, as for example as a credibility measure.

- All the sequences of images used in this work have been taken with a wide-angle lens. As mentioned during chapter 4, these lenses constraint the scene to be as close as possible to the camera. Further tests could be done using a tele-photo lens. These kind of lens would allow us to evaluate wide space scenes.

- Finally, and the most ambitious, implement the depth map computation method within the dual bracketing module and thus, be able to estimate depth in real time by means of the camera.
BIBLIOGRAPHY


