Università degli Studi di Salerno

Université de Lorraine

FACULTY OF ENGINEERING

Master Thesis

in

COMPUTER ENGINEERING

Disparity map extraction for a low cost 3D sensor

Supervisors:

Prof. Mario Vento

Prof. Salvatore-Antoine Tabbone

Co-supervisor:

Eng. Antonio Greco

Candidate:

Roberto Pisapia

Matr.: 0622700292

Academic year 2015-16

To Arianna

Abstract

The aim of this thesis is the development of a new low cost sensor for the 3D acquisition. The 3D sensor provides several features, like a tool for initial configuration of the sensor, the synchronized acquisition from both the cameras, the rectification of the captured images and the processing of the image to get a range map used in many different applications.

Given the presence of high cost devices, which allow to obtain the three-dimensional representation of the environment taken into consideration, the purpose of this work is to realize a low cost sensor, that makes possible the stereo acquisition, and to produce a depth image from the disparity map. This sensor can be used for counting and classification of people, or for a 3D reconstruction of the environment under consideration.

The 3D sensor consists of two cameras in a PVC container, connected via USB to Raspberry Pi 2, which handles the video stream acquisition and the image processing.

The first step is the assembly of the used components; particular attention was drawn to the layout of the cameras, to avoid misalignment that can negatively influence the result of the disparity map extraction. Although the two aligned cameras, the initial calibration is always necessary, to eliminate the radial distortions related to inherent defects of the camera lenses. For this initial calibration, it is provided a simple tool, which allows, with the aid of a chessboard pattern, obtaining the calibration parameters, which are used to remove the distortion and possible misalignments between the cameras.

For the acquisition and decoding of the video stream from the cameras, we use the library FFMPEG, which gives the opportunity to acquire individual video streams and then decode them in the desired format. One of the features provided by the sensor is the synchronous acquisition from the cameras, transforming two autonomous cameras in a real stereo camera.

An additional available feature is the rectification of stereo images, which, using the parameters obtained from the calibration performed during the configuration of

the cameras, allows to remove the radial distortion caused by lens imperfections of the camera.

The used algorithm, in order to obtain a depth image, is a generative probabilistic model for stereo matching, which allows dense matching with small aggregation windows by reducing ambiguities on the correspondences.

The employed approach requires a prior compute of the disparity space by forming a triangulation on a set of robustly matched correspondences, named “support points”. The implemented algorithm does not suffer in presence of poorly-texture and slanted surfaces; therefore, it suits at more examined environments. Also, to increase the performance, much part of code was parallelized, using the SIMD instruction set NEON, available on Raspberry CPU.

The last step of this study was to calculate the framerate of the stereo camera and the error rate of the stereo matching algorithm. The framerate of the sensor is settled on 3 fps. The average error rate on 16 images taken by Middlebury Stereo Dataset is 11.7%.

The future development of this work is to complete and to make the sensor 3D a plug and play sensor, usable immediately with a rough configuration. Furthermore, updating the using hardware with a new and more powerful CPU, the potentialities of the sensor may be increased and improved.

Index

List of figures

List of tables

Chapter 1

1 Introduction

The basic idea of the thesis is the realization of a smart low cost sensor in order to extract a depth image, which gives the information on the distance of objects from the camera.

Part of this thesis was made with the team QGAR of the LORIA laboratory at the Nancy University (France).

1.1 Computer vision

The computer vision is a field of information technology that, starting from 70s, is having a lot of interest from the scientific community. His endless uses place it among the pillars of the information technology. Computer Vision can be defined along the lines of 'using computers to discover from images what is in a scene, and where it is' [1]. Using a single image from a single camera, it is possible to obtain a very high number of information, which can be used in various ways and with the most variable purposes. Possible uses of the applications that belong to the artificial vision, can be placed in different areas, such as security, where there is the recognition of the people, very popular in this period, in which you want to quickly identify a crowd-confused criminal, or the detection of a fire, in a limited area, or also, in the commercial sector, where it is required the people count into a shopping mall, or the distinction between the types of customers that passes or stops in front of the window of the shop, or again, in industry sector, for quality control of a product, or in the assistance of the movement of a robot in an unknown environment.

Many of these applications have already become part of our everyday life, without someone to indicate part of the computer vision. Not all possible applications can be implemented starting from a single 2D image. Some researchers started to study the stereo vision.

1.2 Stereo vision

Stereo vision is a traditional method for acquiring 3-D information from a stereo image pair, capturing two different perspectives of the same environment. With only one image, it is impossible, without different information from other sensors, to know the tri-dimensional structure of the scene. The reason is the loss of information inherent in the perspective projection, which maps the points of the 3D space in 2D space. Therefore, this technique needs to refer to the ability to infer information on the 3D structure and distance of objects. This method has many advantages in terms of cost, safety, operating range, undetectable characteristics, and reliability. Mobile robots can take advantage of this system as a reliable and effective way to extract range information from the environment: one of process used from robot is simultaneous localization and mapping (SLAM), the best example is the NASA rover robot, it can navigate safely through unknown and potentially hazardous terrain, using autonomous passive stereo vision to detect potential terrain hazards before driving into them [2].

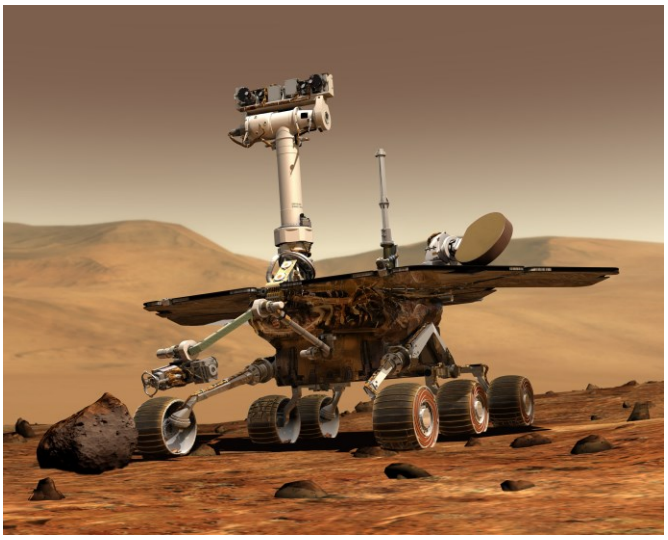

Figure 1.1 - Mars Rover

This technique can also be used for more simple and common applications, like people counting [3] to reconstruct the scene and so to remove the typical problem of people counting, see the inaccurate counting with congested scene. Also mobility aid for visually impaired uses the stereo vision to help blind individuals in their navigation with sound feedback [4].

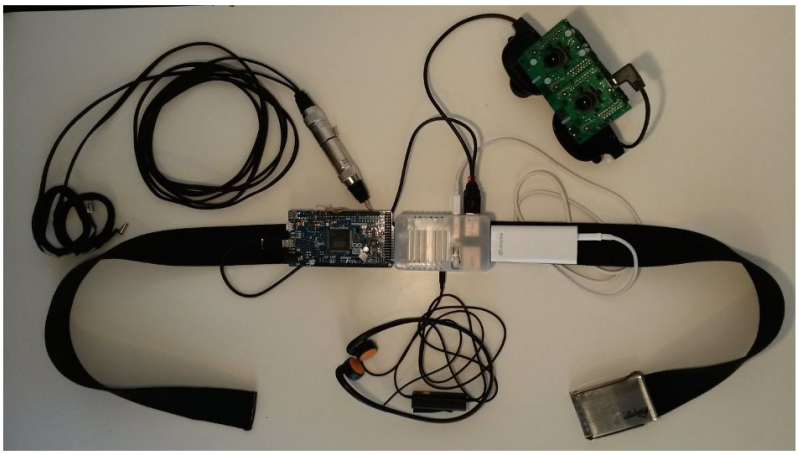

Figure 1.2 - 3D glasses as mobility aid for blind people

1.3 Stereo vision system

The previously listed applications, in order to model the 3D scene follow a series of steps, each one with its aim and difficulty. The steps are:

- camera calibration: if the configuration of cameras remains unchanged, this operation makes once time, to get the parameters of the camera and the position the cameras;
- rectification: using the parameters of previous step, removes the lens distortion and turns the stereo pair in standard form;
- stereo correspondence: aims at finding homologous points in the stereo pair;
- triangulation: computes the position of the correspondence in 3D space with disparity map, baseline and the focal length (not discussed in this thesis).

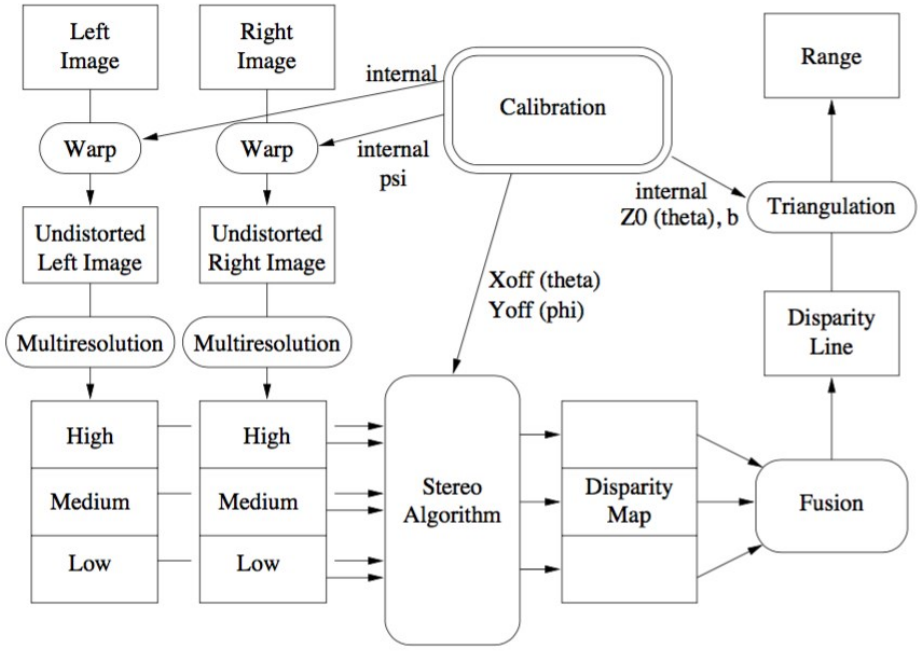

Figure 1.3 - Stereo vision process

1.4 Aim of thesis

The thesis work wants to propose a sensor to obtain a depth image. The use of an additional camera is useful to obtain another important information about the scene and exploit it to realize different and new application.

It is possible to find different devices who develop this task, but many of those use a specific hardware, and, not less important, they have a high cost. For this reason, we propose a house-made sensor, so that all interested person, following this thesis, can make it at home with simple and low cost hardware.

1.5 Raspberry Pi

The first low cost component used to create this sensor, it is **Raspberry Pi**.

This is the sensor core and it performs all the necessary actions to compute the range map and others useful operations at the good working of the device

The Raspberry Pi is a series of credit card-sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and developing countries. The particularity of

this card is that, with a small amount of money, you have a computer used in many different ways. It is used to emulate retro-console, to create an audio system for the house, to build a media centre connectable with the television, to realize a drone what walks, talks and recognize person, and more different projects.

Figure 1.4 - Raspberry Pi 2

The Raspberry Pi used in the sensor is the model 2 B. This is the second generation of Raspberry Pi, and it replaced the original Raspberry Pi 1 Model B+ in February 2015. Like the previous model, it has: 4 USB ports; 40 GPIO pins; Full HDMI port; Ethernet port; Combined 3.5mm audio jack and composite video; Camera interface (CSI); Display interface (DSI); Micro SD card slot; VideoCore IV 3D graphics core

The novelty is the processor, more powerful than the Pi 1, a 900MHz quad-core ARM cortex-A7 CPU and the memory RAM doubled up to 1 GB.

In order to use the raspberry, we installed Raspbian Jessie, a Debian-based computer operating system for Raspberry Pi, developed by a small team of developers.

An additional, the library FFmpeg and OpenCV framework are available on the Raspberry. These libraries are described below, in the section, where they are used, to perform different tasks.

1.6 Outline

The thesis is composed of seven chapters. The following is a brief description of the organization of the document.

Chapter 1 describes the general context in which the work can be collocated, illustrating in general the problems faced by the Computer Vision and the goal of the developed system.

The following chapters, from chapter 2 up to chapter 5, present the modules that make up the sensor.

The chapter 2 talks about the calibration, the action that allows the two single cameras to become a stereo camera.

The chapter 3 describes the acquisition of the images from two cameras, presenting the library to grab and convert the single frames captured from both the cameras.

The chapter 4 talks about the rectification operation, an essential process to facilitate and make less expensive the next module, which makes the horizontal lines aligned between the left and right camera.

The chapter 5 describes the core of the sensor, or rather the stereo matching algorithm. Before to explain the employed algorithm, we explain the difficulties and problems, which make this step approachable with different methods.

In the chapter 6, instead, we show the results in terms of accuracy and computational load, in order to characterize the sensor and the stereo matching algorithm.

In the chapter 7, finally, we draw the conclusions, providing suggestions for the future development of the proposed sensor.

Chapter 2

2 Calibration

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. It has been studied extensively in computer vision and photogrammetry, and even recently new techniques have been proposed. In this chapter, the techniques proposed in the literature are presented, including those using 3D apparatus (two or three planes orthogonal to each other, or a plane undergoing a pure translation, etc.), 2D objects (planar patterns undergoing unknown motions), 1D objects (wand with dots) and unknown scene points in the environment (self-calibration). The focus is on presenting these techniques within a consistent framework.

2.1 Camera model

Before the introduction of the possible methods to calibrate a camera, it is necessary a mathematical formulation of the problem.

The physical space is a 3D Euclidean space (R^3) in which the points, once fixed a reference system, can be represented by three-dimensional vectors. In this space it can say the parallel lines do not intersect, or it intersect at the infinity. The representation of these points at infinity is not possible in this vector space. Thus a new coordinate is added at the tern, the new coordinate is defined in the following way:

$$(kx \quad ky \quad kx \quad k) \quad \forall k \neq 0$$

In this representation, a point in the space is identified as an equivalence class of quadruple, where equivalent quadruples differ only by a multiplicative factor. This is the so-called representation in **homogeneous coordinates** of the point (x, y, z). The space obtained by representing points in homogeneous coordinates is called the **projective space** P^3. Defined the homogeneous coordinates it's possible analytically represent the point at the infinity as point with the last coordinates equal to 0.

In short, every Euclidean space (R^n) can be extended in a new geometric structure, the so-called projective space (P^n), representing the points in homogeneous coordinates. This representation provides for the addition of a new coordinate k to the space vector:

$k \neq 0$ represents real points of (R^n)

$k = 0$ represents the ideal points of (R^n), the so-called points at infinity.

This new structure, therefore, allows to represent and analytically consider homogeneously, without introducing exceptions or special cases, also points at infinity.

So, considering a 3D point in a scene $M = [x, y, z]^T$, with coordinates express in the reference system respect the camera (C), and his projection on the image plane I, indicated with $m = [u, v]^T$.

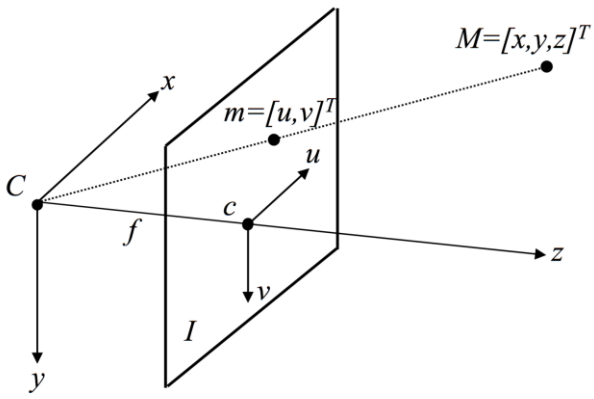

Figure 2.1 - Camera model

Considering the focal length, f, the nonlinear equation that express the image coordinates as a function of 3D coordinates are given by:

$$\begin{cases} u = \frac{f}{z} \cdot x \\ v = \frac{f}{z} \cdot y \end{cases}$$

The coordinates of the previous points can be express in a homogeneous coordinates, showing them with a tilde:

$$\tilde{\boldsymbol{m}} = [u \quad v \quad 1]^T \qquad \tilde{\boldsymbol{M}} = [x \quad y \quad z \quad 1]^T$$

With this new way to express the coordinates, the perspective projection become a linear transformation.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot \frac{x}{z} \\ f \cdot \frac{y}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot x \\ f \cdot y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Indicating, all with the matrix notation:

$$k\tilde{\boldsymbol{m}} = \tilde{\boldsymbol{P}}\tilde{\boldsymbol{M}}$$

Where the $\tilde{P}$ matrix represent the geometric model of the camera, and it's said **perspective matrix**. In the case the distance are measured in units of focal distances, with $f = 1$, the perspective matrix become $\tilde{\boldsymbol{P}} = [\boldsymbol{I}|\boldsymbol{0}]$, with $\boldsymbol{I}$ identity matrix.

In real acquisition system, you have to consider the digitizing the image and the rigid transformation between the camera and the scene (roto-translation).

The digitization is considered inserting in the projection formulas the scaling along the two axes due to the quantization of the image plane and the translation of the piercing point (main point) due to the choice of the reference system of the image.

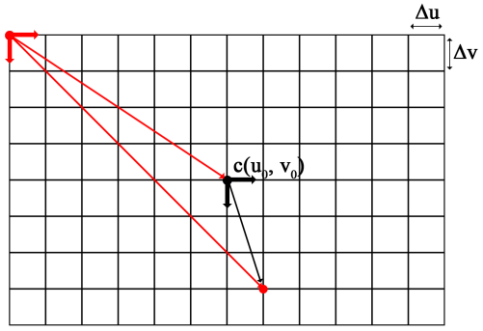

Figure 2.2 - Digitization error

Δu = horizontal size of pixel

Δv = vertical size of pixel

$$u = \frac{f}{z} \cdot x \rightarrow u = \frac{1}{\Delta u} \cdot \frac{f}{z} \cdot x = k_u \cdot \frac{f}{z} \cdot x + u_0$$

$$v = \frac{f}{z} \cdot y \rightarrow v = \frac{1}{\Delta v} \cdot \frac{f}{z} \cdot y = k_v \cdot \frac{f}{z} \cdot y + v_0$$

With the new relations, the perspective matrix can be rewritten in the following way:

$$\widetilde{\boldsymbol{P}} = \begin{bmatrix} f \cdot k_u & 0 & u_0 & 0 \\ 0 & f \cdot k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f \cdot k_u & 0 & u_0 \\ 0 & f \cdot k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \boldsymbol{A}[\boldsymbol{I}|\boldsymbol{0}]$$

The A matrix, that model the characteristics of the sensor, is called **intrinsic parameters matrix**. It's possible minimize the necessary parameters placing $\alpha_u = f \cdot k_u$ and $\alpha_v = f \cdot k_v$, that is the focal length in horizontal pixels and the focal length in vertical pixels. The intrinsic parameters, however, are five and not four; the fifth parameter is the skew, the angle between the axes of the sensor reference system. The cotangent of this angle deals the position $\boldsymbol{A}[1,2]$, but it's always equal to 0, because practically the axes are orthogonal.

Also the rigid transformation between the camera and the scene has to consider to a good mapping between the coordinates of the image plane and the 3D point of the space. Indicating with $\boldsymbol{R}$, the rotation matrix between the camera reference system and the space reference, and with $\boldsymbol{T}$, the translation of the these latter, the link between the point in the image plane and the 3D point become:

$$\widetilde{\boldsymbol{m}} = \boldsymbol{A}[\boldsymbol{I}|\boldsymbol{0}]\boldsymbol{G}\widetilde{\boldsymbol{W}} \quad \rightarrow \quad \widetilde{\boldsymbol{m}} = \boldsymbol{A}[\boldsymbol{I}|\boldsymbol{0}] \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix} \widetilde{\boldsymbol{W}}$$

The general formulation of the perspective matrix is:

$$\widetilde{\boldsymbol{P}} = \boldsymbol{A}[\boldsymbol{I}|\boldsymbol{0}]\boldsymbol{G} \qquad \text{or} \qquad \widetilde{\boldsymbol{P}} = \boldsymbol{A}[\boldsymbol{R}|\boldsymbol{T}]$$

The $\boldsymbol{G}$ matrix, that model the position of the camera respect the scene, is called **extrinsic parameters matrix**.

So, finally the camera model can be reassumed in the following relation:

$$s\widetilde{m} = \boldsymbol{A}[\boldsymbol{R} \quad \boldsymbol{T}]\widetilde{M}$$

where s is an arbitrary scale factor.

All these parameters are related to the single camera, when you want to make a stereo calibration, you need to add others information. Then, the rotation matrix, $\boldsymbol{R}_{stereo}$, between the left camera reference system and the other, and the translation between the two cameras, $\boldsymbol{T}_{stereo}$. In theory, the knowledge of extrinsic compared to the same reference system for both cameras (i.e. the reference system of the first frame), allows to obtain this information, composing transformations $\boldsymbol{G}_i\left(\boldsymbol{G}_j\right)^{-1}$, where i and j are the two cameras.

This method requires hardware synchronized cameras or the use of the static pattern, and however, this approach is little robust respect the noise and it can lead to results very far from the real acquisition system.

So, with the stereo calibration it chooses the standard reference system of one of the two camera, for example the left. The right extrinsic parameters are represented by the addition rotation and translation matrices. The corresponding PPM is then:

$$\widetilde{\boldsymbol{P_L}} = \boldsymbol{A_L}[\boldsymbol{I}|\boldsymbol{0}] \qquad \widetilde{\boldsymbol{P}}_{\boldsymbol{R}} = \boldsymbol{A_R}[\boldsymbol{R}|\boldsymbol{T}]$$

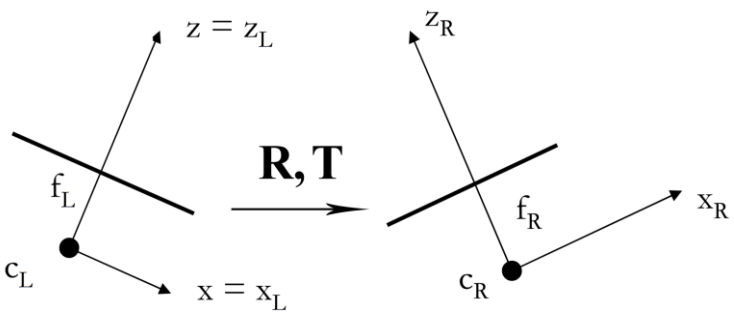

Figure 2.3 - Relation between camera model

2.2 Distortions

In theory, it is possible to define a lens that will introduce no distortions. In practice, however, there are no perfect lenses. This is mainly for reasons of manufacturing; it is much easier to make a “spherical” lens than to make a more mathematically ideal “parabolic” lens. It is also difficult to mechanically align the lens and imager exactly.

The distortion is generally referred to an optical aberration that deforms and bends physically straight lines and makes them appear curvy in images, which is why such distortion is also commonly referred to as “curvilinear”. Optical distortion occurs as a result of optical design, when special lens elements are used to reduce spherical and other aberrations. In short, optical distortion is a lens error, for this reason they are often call lens distortion.

Among the optical distortion you can find the **radial distortion** and the **tangential distortion**.

The radial distortions arise as a result of the shape of lens. They exist three types of radial distortion: barrel, pincushion and moustache, also known as wavy and complex.

When straight lines are curved inwards in a shape of a barrel, this type of aberration is called **barrel distortion**. Commonly seen on wide angle lenses, barrel distortion happens because the field of view of the lens is much wider than the size of the image sensor and hence it needs to be squeezed to fit. As a result, straight lines are visibly curved inwards, especially towards the extreme edges of the frame. Here is an example of strong barrel distortion:

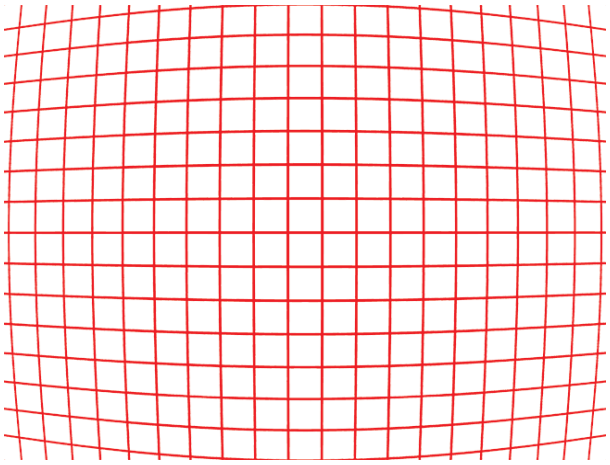

Figure 2.4 - Barrel distortion

Pincushion distortion is also a very common aberration, especially on zoom lenses. In fact, pincushion distortion can be very heavy on consumer-grade lenses. It's important to note that most zoom lenses that go from wide angle to standard or telephoto focal lengths typically suffer from barrel distortion at the shortest focal lengths, which gradually transitions to pincushion distortion towards the longest end. Just like barrel distortion, pincushion distortion can also be easily fixed in post-processing.

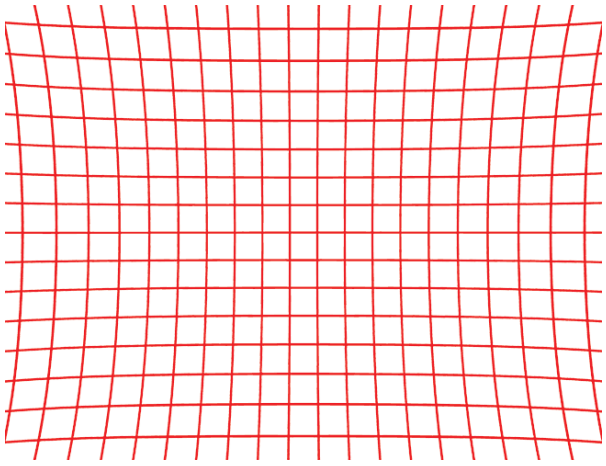

Figure 2.5 - Pincushion distortion

The nastiest of the radial distortion types is **moustache distortion**, which it is sometimes called "wavy" distortion. It is basically a combination of the barrel

distortion and pincushion distortion. Straight lines appear curved inwards towards the center of the frame, then curve outwards at the extreme corners, as shown below.

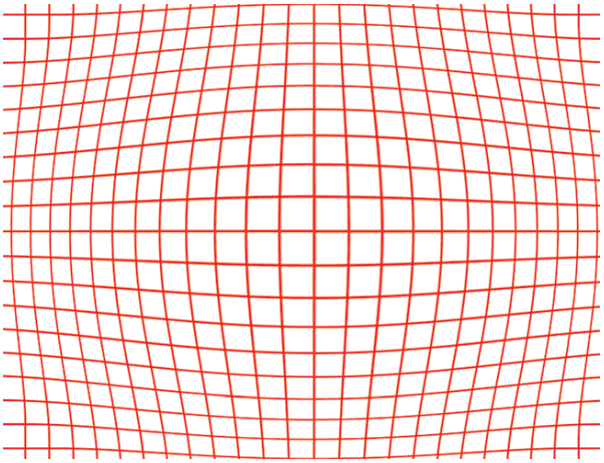

Figure 2.6 - Moustache distortion

With some lenses, rays farther from the center of the lens are bent more than those closer in. The most common distortion is the barrel distortion; it is particularly noticeable in cheap web cameras and less apparent in high-end cameras where a lot of effort is put into fancy lens systems that minimize radial distortion. The Figure 2.7 gives some intuition as to why radial distortion occurs. The other kinds of distortions that occur in imaging systems, but they typically have lesser effects than radial and tangential distortions.

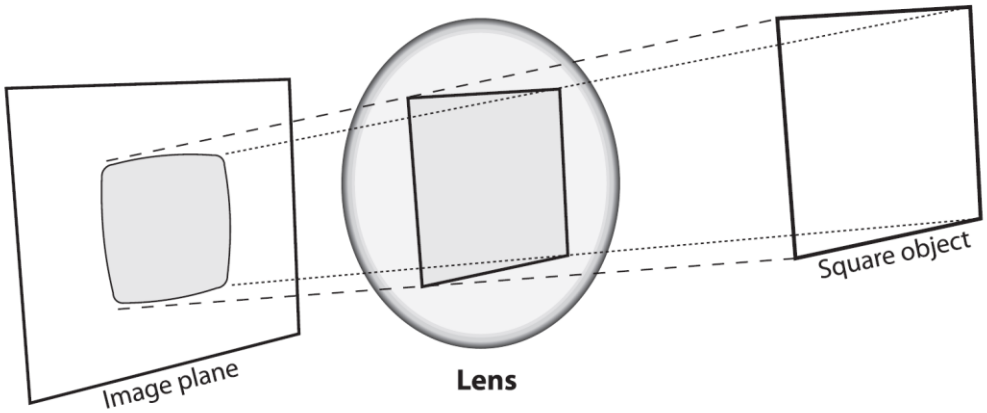

Figure 2.7 - Why radial distortion occurs

Tangential distortions introduces secondary effects, the decentralization of the components of a lens system and manufacturing defects, it arise from the assembly process of the camera as a whole.

All these phenomena are modelled using a nonlinear relationship between the points actually "observed" on the image plane.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix}$$

Here, $(\tilde{x}, \tilde{y})$ is the original location of the distorted point and (x', y') is the new location as a result of the correction. The radial distortion function $L(r)$ is defined only for positive r; it depends on the distance r from the center distortion (x_c, y_c), calculated with the classic formula of the distance between two points:

$$r = \sqrt{(\tilde{x} - \tilde{x}_c)^2 + (\tilde{y} - \tilde{y}_c)^2}$$

This nonlinear function is typically approximated by its Taylor expansion, up to the n-th order depending on the desired precision.

$$L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots$$

For cheap cameras, it is generally used the first two such terms; the first of which is conventionally called k_1 and the second k_2. For highly distorted cameras such as fish-eye lenses a third radial distortion termk_3 is used.

Tangential distortion is minimally characterized by two additional parameters, p_1 and p_2.such that:

$$\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} = \begin{pmatrix} 2p_1\tilde{x}\tilde{y} + p_2(r^2 + 2\tilde{x}^2 \\ p_1(r^2 + 2\tilde{y}^2) + 2p_2\tilde{x}\tilde{y} \end{pmatrix}$$

The coefficients for the correction of radial distortion $k_1, k_2, \ldots k_n$ together with the center of the radial distortion $(\tilde{x}_c, \tilde{y}_c)$ and the two tangential distortion coefficients p_1 and p_2 expand and complement the set of intrinsic parameters of the standard model of a camera. Typically, it is assumed, for simplicity, that the radial distortion center coincides with the centre of the image.

Thus in total there are five distortion coefficients; since all five are, they are typically bundled into one distortion vector; this is just a 5-by-1 matrix containing k_1, k_2, p_1, p_2, and k_3.

2.3 Calibration methods

Much work has been done, starting in the photogrammetry community, and more recently in computer vision. According to the dimension of the calibration objects, those techniques can classify roughly into three categories:

3D reference object based calibration. Camera calibration is performed by observing a calibration object whose geometry in 3-D space is known with very good precision. The calibration object usually consists of two or three planes orthogonal to each other. Sometimes, a plane undergoing a precisely known translation is also used [5], which equivalently provides 3D reference points. Calibration can be done very efficiently, but it requires an expensive calibration apparatus and an elaborate setup.

Figure 2.8 - 3D reference object

2D plane based calibration. Techniques in this category requires to observe a planar pattern shown at a few different orientations. Different from Tsai's technique [5], the knowledge of the plane motion is not necessary. Because almost anyone

can make such a calibration pattern by him/her-self, the setup is easier for camera calibration.

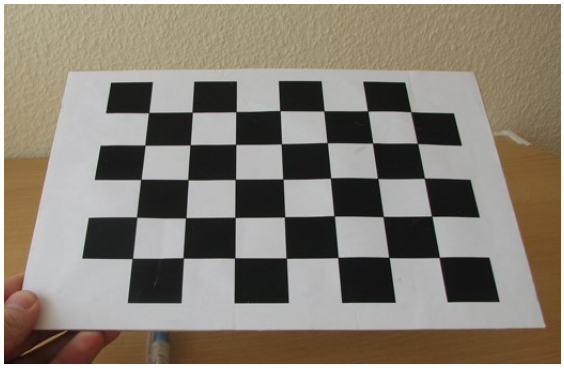

Figure 2.9 - 2D reference object

1D line based calibration. Calibration objects used in this category are composed of a set of collinear points [6]. As will be shown, a camera can be calibrated by observing a moving line around a fixed point, such as a string of balls hanging from the ceiling.

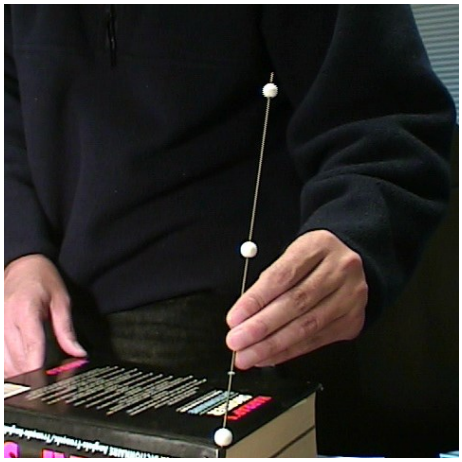

Figure 2.10 - 1D reference object

Self-calibration. Techniques in this category do not use any calibration object, and can be considered as 0D approach because only image point correspondences are required. Just by moving a camera in a static scene, the rigidity of the scene provides in general two constraints [7] on the cameras' internal parameters from one camera displacement by using image information alone. Therefore, if images are taken by the same camera with fixed internal parameters, correspondences between three images are sufficient to recover both the internal and external parameters which

allow us to reconstruct 3-D structure up to a similarity. Although no calibration objects are necessary, a large number of parameters need to be estimated, resulting in a much harder mathematical problem.

Before going further, we would like to point out that no single calibration technique is the best for all. It really depends on the situation a user needs to deal with. Here there are the recommendations to follow, in case you are undecided on the approach to be taken:

Calibration with apparatus vs. self-calibration. Whenever possible, if a calibration apparatus is available, it is preferable pre-calibrate a camera with it. Self-calibration cannot usually achieve accuracy comparable with that of pre-calibration because self-calibration needs to estimate a large number of parameters, resulting in a much harder mathematical problem. When pre- calibration is impossible (e.g., scene reconstruction from an old movie), self- calibration is the only choice.

Partial vs. full self-calibration. Partial self-calibration refers to the case where only a subset of camera intrinsic parameters has to be calibrated. Along the same line as the previous recommendation, whenever possible, partial self- calibration is preferred because the number of parameters to be estimated is smaller. Take an example of 3D reconstruction with a camera with variable focal length. It is preferable to pre-calibrate the pixel aspect ratio and the pixel skewness.

Calibration with 3D vs. 2D apparatus. Using a 3D apparatus can usually be obtained a highest accuracy, so it should be used when accuracy is indispensable and when it is affordable to make and use a 3D apparatus. With the coming feedback from computer vision researchers and practitioners around the world, in the last couple of years, calibration with a 2D apparatus seems to be the best choice in most situations because of its ease of use and good accuracy.

Calibration with 1D apparatus. This technique is relatively new, and it is hard for the moment to predict how popular it will be. It, however, should be useful especially for calibration of a camera network. To calibrate the relative geometry between multiple cameras as well as their intrinsic parameters, it is necessary for all involving cameras to simultaneously observe a number of points. It is hardly

possible to achieve this with 3D or 2D calibration apparatus if one camera is mounted in the front of a room while another in the back. This is not a problem for 1D objects.

2.4 Help from OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV has a modular structure, which means that the package includes several shared or static libraries [8]. Among the available modules, we used calib3d module, which includes basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

The use of the OpenCV library has been dictated by the need to use something standardized and tested countless times to obtain the best estimate of the parameters, which, as you shall see below, is of vital importance for the good realization of the 3D sensor. It is possible find different tools that allow the identification of the parameters described above, stand out the Matlab stereo calibration app. The final decision has reverted on the use the functionality provided by the library OpenCV, because it was decided to make available an on-site the service, so easy to use and immediately available for any of the camera configuration changes supplied sensor.

The calibration process, used by OpenCV, is a 2D plane based calibration. In this routine, the method of calibration is to target the camera on a known structure that has many individual and identifiable points. By viewing this structure from a variety of angles, it is possible to then compute the relative location and orientation of the camera at the time of each image as well as the intrinsic parameters of the camera. In order to provide multiple views, we rotate and translate the object (see Figure 2.13).

Given many joint views of chessboard corners, the process solves for rotation and translation parameters of the chessboard views for each camera separately. It then

plugs these left and right rotation and translation solutions into the equations just displayed to solve for the rotation and translation parameters between the two cameras. Because of image noise and rounding errors, each chessboard pair results in slightly different values for R and T. Starting with the median values for R and T parameters as the initial approximation of the true solution, a robust Levenberg-Marquardt [9] iterative algorithm to find the (local) minimum of the reprojection error of the chessboard corners for both camera views, and the solution for R and T is returned.

To be clear on what stereo calibration gives you: the rotation matrix will put the right camera in the same plane as the left camera; this makes the two image planes coplanar but not row-aligned (in the chapter 0 is seen how row-alignment is accomplished)

2.5 Proposed tool

To obtain all parameters previously described, a simple tool is made available on the sensor. This command line tool gives the possibility to grab the images from the stereo camera and automatically compute the necessary matrices. First of all, to utilize this software, it's necessary to procure a chessboard pattern for which you know the size of the squares. In the Figure 2.11, there is a half-scale chessboard; in the real dimensions the square size is equal 1 inch.

Once prepared the chessboard, the settable parameters, which the user can select are:

- the number of inner corners for the horizontal dimension, in the case of the chessboard previously presented is nine (necessary parameter);
- the number of inner corners for the vertical dimension, in the case of the chessboard previously presented is six (necessary parameter);
- the number of captured frame, it's advisable use a number of the images greater than twenty images, to obtain an acceptable process accuracy (follow it explains the acceptability of the procedure) (necessary parameter);
- the frame size has to be express because the focal length in pixel is linked at the dimension of the captured image;

- squares size, as previously said, the squares dimension of the presented chessboard is 1 inch;
- the output filename for intrinsic and extrinsic parameters;
- indicate the taking zero tangential distortion, in the default case, though the tangential distortion can be neglected, the tangential distortion coefficients are computed;
- indicate if fix the principal point at the center, in the default case, the principal point changed during the global optimization.

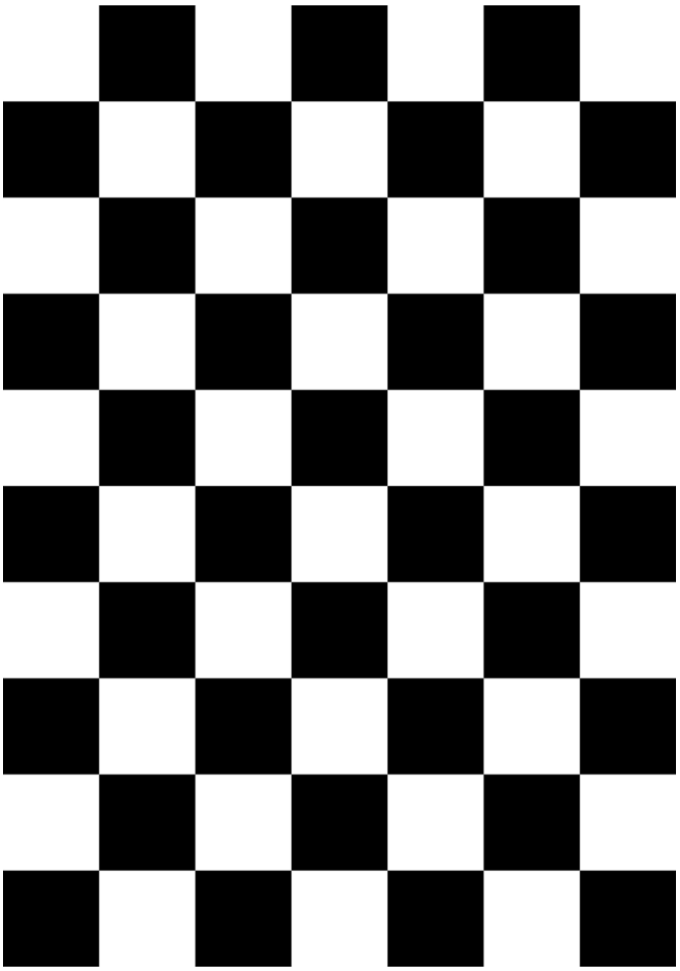

Figure 2.11 - Chessboard pattern (scale 1:2)

After having specified all necessary parameters, starts the capture of the images from both cameras. For each image is sought the chessboard, of size specified with parameters. If in both image, from the two cameras, the chessboard is identified,

the position of all control points on the chessboard are saved in a multi-dimensional array. So, if the used chessboard is formed of 10x7 squares, for a pair images are stored 17 points, or else 34 coordinates.

These steps are repeated as often as the specified number during the tool initialization. At the end of these steps starts the process described above, or else the found of result with a closed-form solution and then optimized by Levenberg-Marquardt [9].

The results produced are stored in a file, with a name specified during tool initialization; in particular, knowing that the intrinsic parameters matrix is composed of:

$$M = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix};$$

the distortion array is composed of:

$$D = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3];$$

the translation vector is composed of:

$$T = [x \quad y \quad z];$$

the rotation matrix is composed of:

$$R = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix}.$$

It is possible to found in saved file the following matrices:

- M_1: the intrinsic matrix of the camera 1;
- M_2: the intrinsic matrix of the camera 2;
- D_1: distortion array of camera 1;
- D_2: distortion array of camera 2;
- R: rotation matrix of camera 2 with camera 1 as reference;
- T: translation camera 2 from camera 1.

2.6 Accuracy

"International vocabulary of metrology" defines measurement accuracy as "closeness of agreement between a measured quantity value and a true quantity value of a measurand" [10]. The document also notes that accuracy is not a quantity expressed as a numerical value but an attribute of a measurement: a measurement is said to be more accurate if it results in a smaller measurement error. Measurement precision is defined as "closeness of agreement between indications or measured quantity values obtained by replicate measurements on the same or similar objects un- der specified conditions" [10]. To ensure more accurate results of the calibration algorithm, a number of requirements has to be met. The view of planar calibration target shall not parallel in two or more calibration images. For better estimation of the camera distortion, the calibration target shall appear in all four corners of the image and cover as much exterior orientations as possible.

To give a quantitative measure to the accuracy of the calibration, the typical measure of camera calibration accuracy is a **reprojection error**, where, given intrinsic parameters and known extrinsic parameter of a view, the known object points are projected onto the screen. Compute the real world model, the reprojection error is the difference between the found points on an image and the same points obtained compute with the real world model.

Then, the root mean square (RMS) projection error is computed between real pixel coordinates (x_i, y_i) and the projected ones $\left(x_i^{proj}, y_i^{proj}\right)$

$$E_{reprojection} = \sqrt{\frac{1}{n_{features}} \sum_{i=1}^{n_{features}} \left[\left(x_i - x_i^{proj}\right)^2 + \left(y_i - y_i^{proj}\right)^2\right]}$$

The reprojection error is used to compute the accuracy of single camera and for the stereo camera. Thus, we obtain three error to characterize the goodness of calibration, two for the single cameras and one for the stereo camera formed by the single cameras.

These results, show below, were reached with the adroitness following presented.

Although the chessboard can easily be procured with a common printer, the cheap ink-jet or laser printers are not designed for rigorous geometrical accuracy. So the used chessboard for the experimental was printed by a professional print shop, which does a much better job than most home printers.

Choose a very flat backing (for the size you mention window glass 5 mm thick or more is excellent, though obviously fragile) and verify its flatness against another edge (or, better, a laser beam).

Attach the pattern very carefully to the backing, using spray-on glue and slowly wiping with soft cloth to avoid bubbles and stretching, and wait for a day or longer for the glue to cure and the glue-paper stress to reach its long-term steady state.

To obtain a sub-millimeter calibration accuracy the environment condition should be considered. The temperature and humidity changes affect on the paper dilation or contraction, because the paper, where the chessboard is printed, absorbs water from the air. And then the lower temperature dilation coefficient of glass compared to common sheet metal is another reason for preferring the former as a backing.

The auto-focus feature of camera must disable, if it has one: focusing physically moves one or more pieces of glass inside your lens, thus changing slightly the field of view and a lot the lens distortion and the principal point.

Place the camera on a stable mount that won't vibrate easily; minimizing the vibrations and associated motion blur when taking photos.

Span the calibration volume when taking pictures. Ideally you want your measurements to be uniformly distributed in the volume of space you will be working with. Most importantly, make sure to angle the target significantly with respect to the focal axis in some of the pictures, to calibrate the focal length you need to frame some real perspective foreshortening.

Use good lighting. Use diffuse ambient lighting, and bounce it off white cards on both sides of the field of view.

And finally you take extra images to verify the accuracy of the solution, using them to verify that the lens distortion is actually removed.

After using both 14x10 chessboard and 10x7 chessboard, the best result was obtained with 10x7 chessboard captured 200 times from each camera. In the Figure 2.12 you can view a set of images captured from the camera in different position and covering all field of view of the camera.

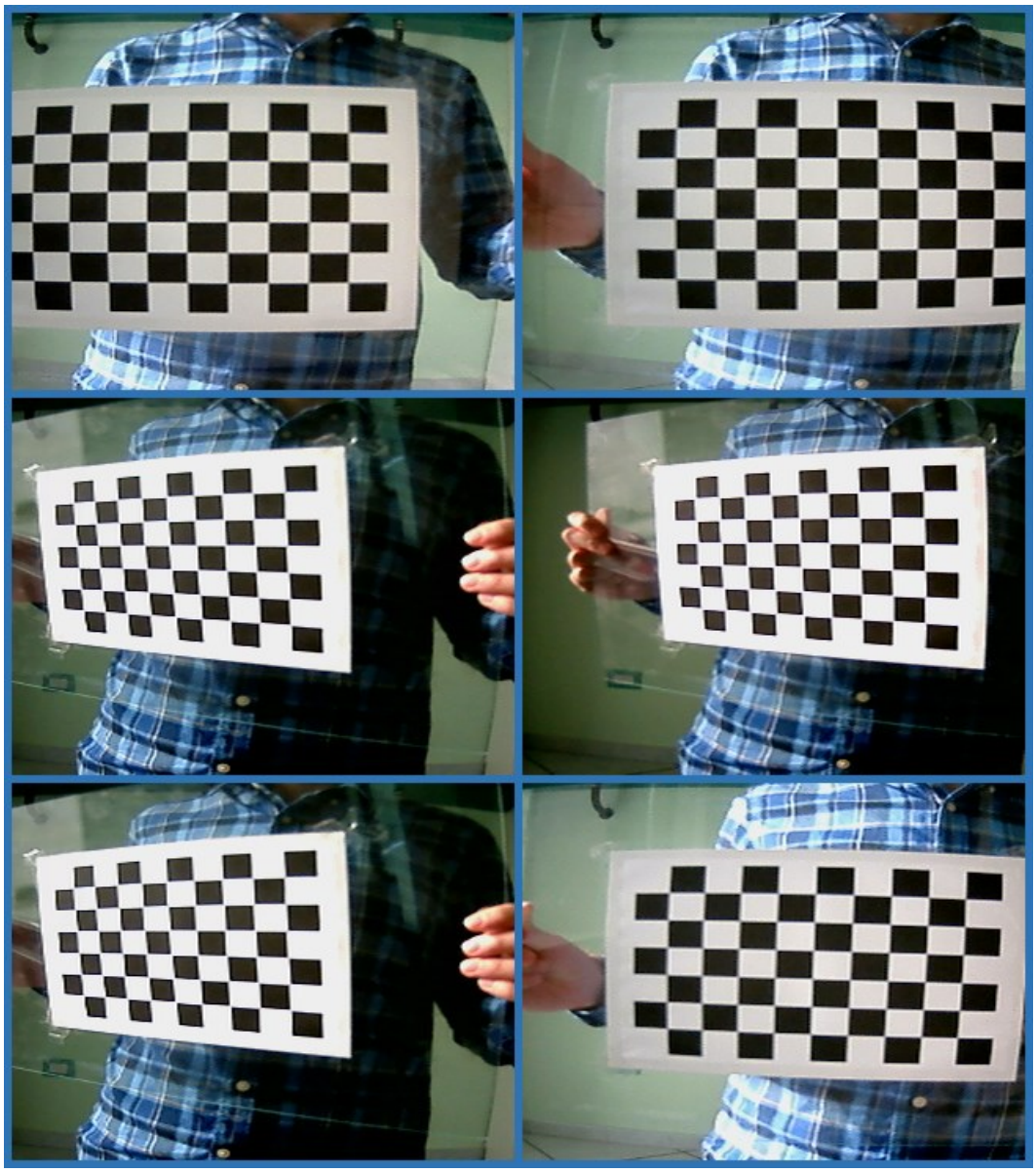

Figure 2.12 - Sub-set image for calibration

You can easily notice the radial distortion introducing from the camera lens, the chessboard rows are very curved on the board of the image.

Rotating and translating the chessboard in different configuration, like Figure 2.13, we provide a discrete number of data to compute the necessary parameters.

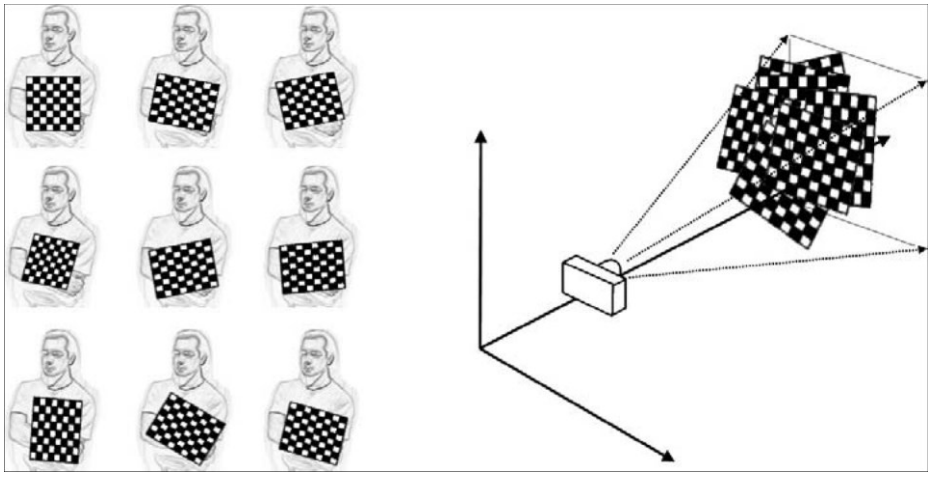

Figure 2.13 - Images of a chessboard being held at various orientations

All captured images from the stereo camera are schematized using Matlab, the following plot, like the right example in Figure 2.13, shows all assumed position to calibrate correctly the stereo camera.

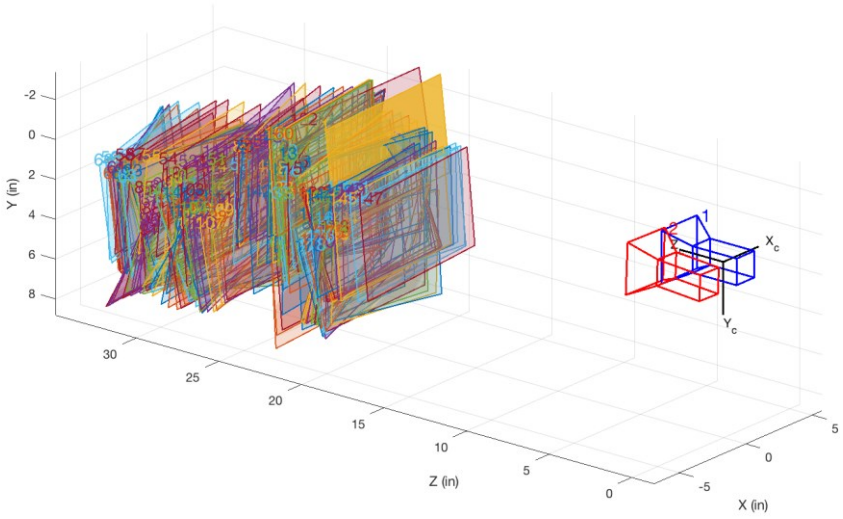

Figure 2.14 - Plot chessboard acquisition

The best results obtain from the best calibration are listed below:

$$M_1 = \begin{bmatrix} 427.89 & 0 & 127.63 \\ 0 & 320.42 & 115.11 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 425.75 & 0 & 152.84 \\ 0 & 318.7 & 99.98 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D_1 = [-0.44 \quad 0.31 \quad -3.85 \times 10^{-6} \quad 1.06 \times 10^{-3} \quad 0]$$

$$D_2 = [-0.44 \quad 0.24 \quad 6.49 \times 10^{-4} \quad 6.21 \times 10^{-3} \quad 0]$$

$$R = \begin{bmatrix} 99.997 \times 10^{-2} & -8.654 \times 10^{-4} & -7.274 \times 10^{-3} \\ 1.165 \times 10^{-3} & 99.914 \times 10^{-2} & 4.129 \times 10^{-2} \\ 7.232 \times 10^{-3} & -4.13 \times 10^{-2} & 99.912 \times 10^{-2} \end{bmatrix}$$

$$T = [2.5934 \quad -2.8744 \times 10^{-2} \quad -1.9285 \times 10^{-1}]$$

With all complete set, the achieve reprojection error is presented in Table 2.1.

Camera	Reprojection error
camera 1	0.00393895
camera 2	0.00416836
stereo camera	0.00467847

Table 2.1 - Reprojection error

There is to remember, since the cameras are fixed on the sensor, and only with the destruction of the sensor can be remove from it, the distance between two cameras and their orientation can't unchanged, and so all parameters indicated in the above list are always usable.

Chapter 3

3 Stereo acquisition

The stereo acquisition is the first module of all chain used by the sensor. “No images, no extraction of disparity map”, so this it is a fundamental part of the device. This module has to join two autonomous cameras in a stereo camera.

First of all, the acquisition by a single camera, is carried out with a library FFmpeg and after to have grabbed the two frames by two cameras, we shape the stereo image, used with the next modules.

3.1 Acquisition by single camera

As say above, the acquisition by a single camera, is carried out with a library FFmpeg.

FFmpeg is a wonderful library for creating video applications or even general purpose utilities. FFmpeg takes care of all the hard work of video processing by doing all, the decoding, encoding, muxing and demuxing. This can make media applications much simpler to write. It's simple, written in C, fast, and can decode almost the most obscure ancient formats up to the cutting edge, as well as encode several other formats. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

FFmpeg contains the homonym application ffmpeg, but also ffserver, ffplay and ffprobe which can be used by end users, the parts that can be used from the application is contained in libavcodec, libavutil, libavformat, libavfilter,

libavdevice, libswscale and libswresample. And just these last libraries are specifically used to acquire the image from the single camera.

Before to show how to acquire the image from camera it is necessary to define some components of a video. First, the movie file is called a container, and in it different information are collected. Among these information, a bunch of streams are presented; for example, an audio stream and a video stream. The stream is just a fancy word for a succession of data elements made available over time. The data elements in a stream are called frames. Each stream is encoded by a different kind of codec. The codec defines how the actual data is COded and DECoded - hence the name CODEC. Packets are then read from the stream. Packets are pieces of data that can contain bits of data that are decoded into raw frames that we can finally manipulate for our application.

A plugged camera is slightly different by a video file, but we have to know that the Raspbian OS, like others system Unix-based (see Debian, on which Raspbian is based), requires a virtual device node to access and control the device in question. This is automatically created, if the driver of the device is available. In case in which the device is a camera and the drivers are available, the system create a virtual device node with the path "/dev/video*". This path is fundamental to access at the camera like a video file, with an infinite stream.

Having said that, a multimedia file can be handled with FFmpeg very simple way, the step to do are: opening of a video stream, reading of a packet from video stream, extraction of the frame from the packet and using of the frame.

More specifically, after to have initialize the libraries of FFmpeg, we have to register all available file formats and codecs of the library so they will be used automatically when a file with the corresponding format/codec is opened.

Now we can actually read the information by the camera and set all appropriate setting, like the resolution of the video, but, because the camera can provide different typologies of stream, like audio or video, it is necessary to find the video stream information. Found the video stream information, we can start the video stream and to use the appropriate codec to interpret the received packet.

After all these initial operations, we can read the packet from the video stream and to extract the frame in a native format, but for the next sections the images format is different, so the frame is also converted to obtain the images in the desired format.

3.2 Stereo capture

The single camera acquisition is very simple with FFmpeg library; after some call at different functions, we can obtain frames from the camera. When we talk about stereo vision, we refer at the acquisition by two cameras. These cameras are positioned in the way to simulate the position of the human eyes, so the distance between two cameras is not excessive. To indicate the two cameras, we can call them left camera and right camera. For the stereo acquisition, the FFmpeg library not provide direct support, with some particular methods. So we the module to capture image from the single camera, for both the cameras. Now the problems is synchronized the two image of the cameras to realize a stereo camera, or else capture the images from both the cameras at the same instance. The used cameras are simple and common cameras, plugged with the Raspberry PI with USB interface, without particular functions, for which it is not possible to synchronize the cameras via hardware.

The road to parallelize with two separate threads is resulted ineffective because, the flow of acquisition was uncontrollable and unpredictable and then the images were acquired randomly.

The big problem is the use of the USB cameras, because the Raspberry Pi also has only one root USB port, then all traffic from all connected devices is funnelled down this bus, which operates at a maximum speed of 480mbps. This mean that the images not can be captured perfectly synchronized, but a small imperfection we can obtain a stereo camera.

Finally, the capture of the image from the cameras is performed in sequential way. The operations of recover of the packets from the cameras is realized one after the other, so the packet of the left camera is read, without extract the frame by the packet, it is immediately read the packet of the right camera. After this, the extraction procedure for both the cameras is carried out to obtain the images to elaborate.

3.3 Performance

Some technical details can help to understand better the problem and the performance reached by stereo camera.

The single camera has the follow characteristics: resolution max 640x480 VGA format; USB2.0 HS / FS; size: 32x32mm; voltage 5V DC; current: 120mA; work temperature -20°~70°. The most interesting characteristic is the reached framerate. This camera can obtain at 30fps (framerate per seconds) an image in YUY2 with resolution 640x480, or 320x240. Also others lower resolution can be obtained by the camera but are not used in our scope.

The used format by the camera is YUYV2, this format use the colours space YUV, It encodes a colour image or video taking human perception into account, allowing reduced bandwidth for chrominance components. The YUV model defines a colour space in terms of one luma (brightness) (Y) and two chrominance (UV) components. The format YUYV2 employs 4:2:2 chroma subsampling. The luma components is mixed with the others components. For our scope the brightness component is useful without chrominance components, whereby a conversion in YUV420 is carried. This necessary conversion add a light delay to acquire from the single camera, and in Table 3.1 we can show that the 30fps from data sheet are not attained, but we are satisfy of follow results.

Resolution image	Time single camera (ms)	Fps
320x240	40	25
640x480	40	25

Table 3.1 - Framerate single camera

The stereo capture, theoretically, doubles the computation time but not in this case, because the second camera, while the first camera send the frame by USB, the second camera has already shutter the image and so only the time to transfer the image from the camera and convert it have to considered. In table below the times of the stereo camera for different resolution.

Resolution image	Time stereo camera (ms)	Fps
320x240	47	~21
640x480	57	~18

Table 3.2 - Framerate stereo camera

The synchronized, as above said, is not realized via hardware and then cannot reach the microseconds precision, but for our scope a precision of the milliseconds is acceptable and excellent. To test the effective synchronization of the stereo camera, not having sophisticated tools to test the real precision, we used a stopwatch of a smartphone. This stopwatch is framed by both the cameras, namely stereo camera and in FIGIGU we show the actual precision measured at milliseconds for the different resolutions 320x240 and 640x480.

Figure 3.1 - Test synchronization with resolution image 320x240

Figure 3.2 - Test synchronization with resolution image 640x480

Chapter 4

4 Rectification

The stereo rectification of an image pair is an important component in many computer vision applications. By estimating the epipolar geometry between two images and performing stereo-rectification, the search domain for registration algorithms is reduced and the comparison simplified, because horizontal lines with the same y component in both images are in one to one correspondence. Stereo-rectification methods simulate rotations of the cameras to generate two coplanar image planes that are in addition parallel to the baseline.

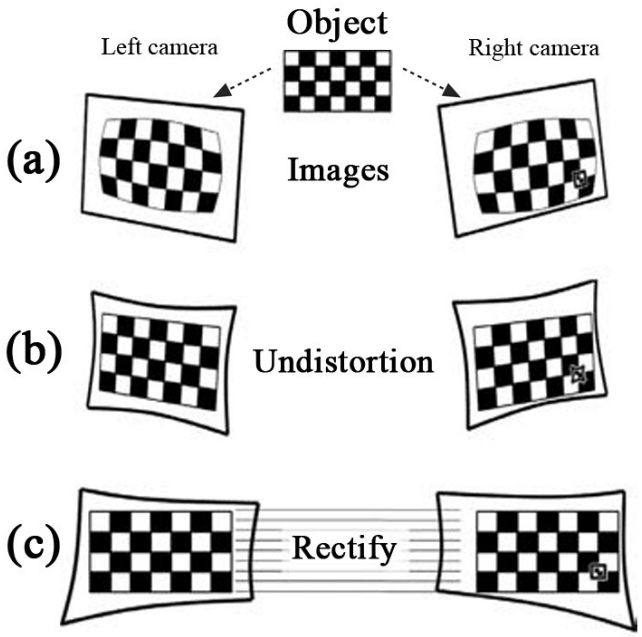

Figure 4.1 - Stereo rectification

4.1 Epipolar geometry

The epipolar geometry of a pair of cameras expresses the fundamental relationship between any two corresponding points in the two image planes, and leads to a key constraint between the coordinates of these points that underlies visual reconstruction.

The world point *P* and the centres of projection of the two cameras identify a plane in space, the epipolar plane of point *P*. The Figure 4.2 shows a triangle of this plane, delimited by the two projection rays and by the baseline of the camera pair, that is, the line segment that connects the two centres of projection. The baseline term is used for the line segment; however, this term is also often used for the entire line through the two centres of projection.

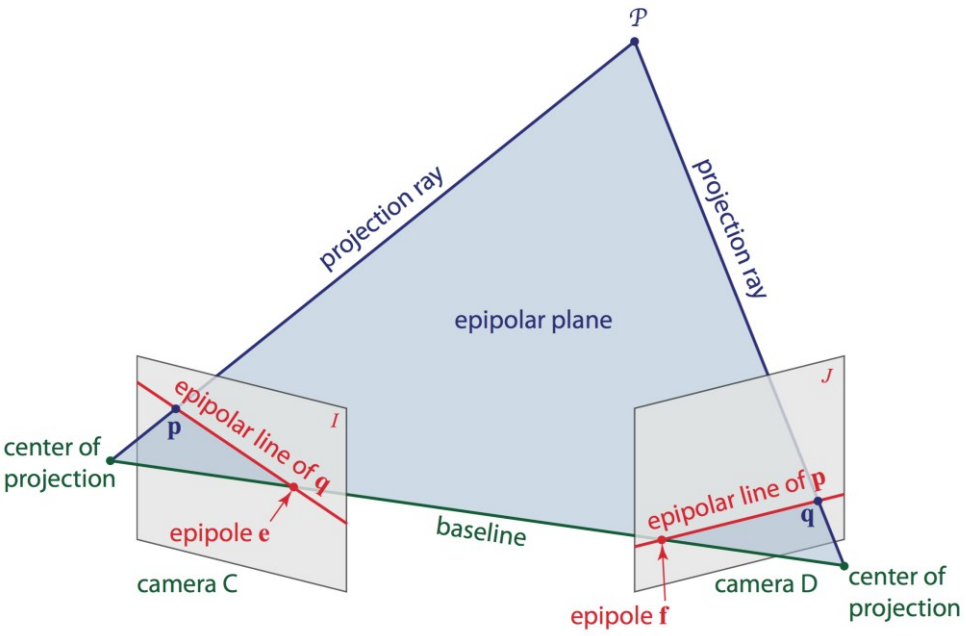

Figure 4.2 - Essential elements of the epipolar geometry of a camera pair

If the image planes are thought of extending indefinitely, the baseline intersects the two image planes at two points called the **epipoles** of the two images. In particular, if the cameras are arranged so that the baseline is parallel to an image plane, then the corresponding epipole is a point at infinity.

The epipoles are fixed points for a given camera pair configuration. With cameras somewhat tilted towards each other, and with a sufficiently wide field of view, the epipoles would be image points. Epipole e in the image I taken by camera C would be literally the image of the center of projection of camera D in I, and vice versa. Even if the two cameras do not physically see each other, this description is maintained in an abstract sense: each epipole is the image of one camera in the other image.

The epipolar plane intersects the two image planes along the two epipolar lines of point P, each of which passes by construction through one of the two projection points $\boldsymbol{p}$ and $\boldsymbol{q}$ and one of the two epipoles. Thus, epipolar lines come in corresponding pairs, and the correspondence is established by the single epipolar plane for the given point P.

For a different world point P, the epipolar plane changes, and with it do the image projections of P and the epipolar lines. However, all epipolar planes contain the baseline. Thus, the set of epipolar planes forms a pencil of planes supported by the line through the baseline, and the epipoles are fixed.

Suppose now that you are given the two images I and J taken by cameras C and D and a point p in I. You do not know where the corresponding point q is in the other image, nor where the world point P is, except that P must be somewhere along the projection ray of p. However, if you know the relative position and orientation of the two cameras, you know where centres of projection are relative to each other. The two centres of projection and point p identify the epipolar plane, and this in turn determines the epipolar line of point p in image J. The point q must be somewhere on this line. This same construction holds for any other point p on the epipolar line in image I.

To understand what the epipolar constraint expresses, consider that the projection rays for two arbitrary points in the two images are generically two skew lines in space. The projection rays of two corresponding points, on the other hand, are coplanar with each other and with the baseline. The epipolar geometry captures this

key constraint, and pairs of point that do not satisfy the constraint cannot possibly correspond to each other.

4.2 Stereo rectification

It is easiest to compute the stereo disparity when the two image planes align exactly. Unfortunately, as discussed previously, a perfectly aligned configuration is rare with a real stereo system, since the two cameras almost never have exactly coplanar, row-aligned imaging planes.

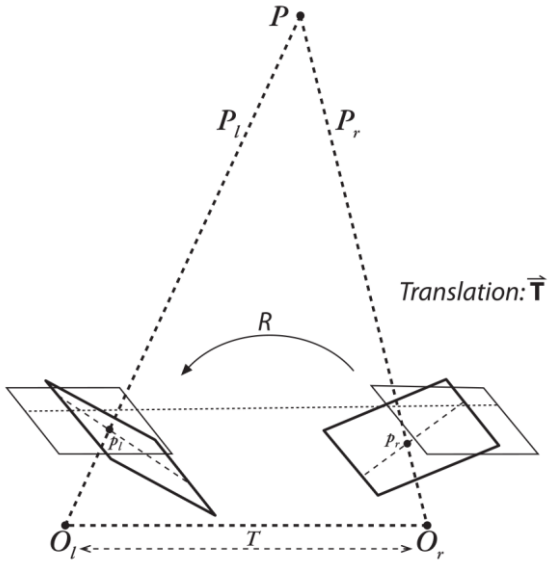

Figure 4.3 - Goal stereo rectification

Figure 4.3 shows the goal of stereo rectification: to reproject the image planes of our two cameras so that they reside in the exact same plane, with image rows perfectly aligned into a frontal parallel configuration.

The image rows between the two cameras have to be aligned after rectification so that stereo correspondence (finding the same point in the two different camera views) will be more reliable and computationally tractable. Note that reliability and computational efficiency are both enhanced by having to search only one row for a match with a point in the other image. The result of aligning horizontal rows within a common image plane containing each image is that the epipoles themselves are then located at infinity. That is, the image of the centre of projection in one image is parallel to the other image plane, but because there are an infinite number of

possible frontal parallel planes to choose from, it is necessary to add more constraints. These include maximizing view overlap and/or minimizing distortion.

The result of the process of aligning the two image planes will be eight terms, four each for the left and the right cameras. Having the intrinsic parameters and rotation and translation matrices for each camera, it is possible make a map, where to interpolate pixels from the original image in order to create a new rectified image.

The used algorithm to rectify the image is the Bouguet's algorithm [11], which uses the rotation and translation parameters from two calibrated cameras.

4.2.1 Bouguet's algorithm

Given the rotation matrix R and translation T between the stereo images, Bouguet's algorithm for stereo rectification simply attempts to minimize the amount of change reprojection produces for each of the two images, and thereby minimize the resulting reprojection distortions, while maximizing common viewing area.

To minimize image reprojection distortion, the rotation matrix R that rotates the right camera's image plane into the left camera's image plane is split in half between the two cameras; calling the two resulting rotation matrices r_l and r_r for the left and right camera, respectively. Each camera rotates half a rotation, so their principal rays each end up parallel to the vector sum of where their original principal rays had been pointing. These rotations put the cameras into coplanar alignment but not into row alignment. To compute the R_{rect} that will take the left camera's epipole to infinity and align the epipolar lines horizontally, a rotation matrix is created by starting with the direction of the epipole e_1 itself. Taking the principal point (c_x, c_y) as the left image's origin, the direction of the epipole is directly along the translation vector between the two cameras' centres of projection.

The next vector, e_2, must be orthogonal to e_1 but is otherwise unconstrained. For e_2, choosing a direction orthogonal to the principal ray (which will tend to be along the image plane) is a good choice. This is accomplished by using the cross product of e_1 with the direction of the principal ray and then normalizing so that we've got another unit vector.

The third vector is just orthogonal to e_1 and e_2, it can be found using the cross product.

The matrix that takes the epipole in the left camera to infinity is then:

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}$$

This matrix rotates the left camera about the centre of projection so that the epipolar lines become horizontal and the epipoles are at infinity. The row alignment of the two cameras is then achieved by setting:

$$R_l = R_{rect} r_l$$

$$R_r = R_{rect} r_r$$

And the projection matrices P_l and P_r:

$$P_l = M_l P_l' = \begin{bmatrix} f_{x\_l} & 0 & c_{x\_l} \\ 0 & f_{y\_l} & c_{y\_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_r = M_r P_r' = \begin{bmatrix} f_{x\_r} & 0 & c_{x\_r} \\ 0 & f_{y\_r} & c_{y\_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Recapitulating in brief, having the calibration information, intrinsic parameter matrix and extrinsic parameter matrix, the proceedings, to align all horizontal lines of left and right camera, is carried out before removing the distortion of camera lens and then rectifying the two obtained images, as you can see in

4.3 Rectification method

The rectification method is divided in two function: initUndistortRectifyMap and remap.

The separation of the rectification method in two separated function is done to minimize the computation time of the rectified images. In fact, considering that the cameras configuration remains unchanged, in that the cameras are the same and their orientation and position are the same, the initUndistortRectifyMap computes the left and right rectification lookup maps for the left and right camera views, one

time at the beginning of the capturing process. While remap function use the map pre-calculated and preserved by previous function taking pixels from one place in the image and mapping them to another place.

As with any image-to-image mapping function, a forward mapping (in which to compute where pixels go from the source image to the destination image) will not, owing to floating-point destination locations, hit all the pixel locations in the destination image, which thus will look like Swiss cheese. So instead, the function works backward: for each integer pixel location in the destination image, look up what floating-point coordinate it came from in the source image and then interpolate from its surrounding source pixels a value to use in that integer destination location.

4.3.1 InitUndistortRectifyMap

The initUndistortRectifyMap computes the undistortion and rectification transformation map.

The function returns lookup maps map_x and map_y as output. These maps indicate from where is needed interpolate source pixels for each pixel of the destination image.

The initUndistortRectifyMap function is called separately for the left and the right cameras so to obtain their distinct map_x and map_y remapping parameters.

The necessary parameters to calculate the lookup map are the intrinsic parameter matrix, the distortion coefficients, rectified rotation matrix and the projection matrix.

To builds the maps for the inverse mapping algorithm that is used by remap, for each pixel (u, v) in the destination corrected and rectified image, the function computes the corresponding coordinates in the source image, or else in the original image from camera.

The following process is applied:

$$x \leftarrow \frac{u - c'_x}{f'_x}$$

$$y \leftarrow \frac{v - c'_y}{f'_y}$$

$$[X \quad Y \quad W]^T \leftarrow R^{-1} \times [x \quad y \quad 1]^T$$

$$x' \leftarrow \frac{X}{W}$$

$$y' \leftarrow \frac{Y}{W}$$

$$x'' \leftarrow x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' \leftarrow y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

$$map_X(u, v) \leftarrow x'' f_x + c_x$$

$$map_y(u, v) \leftarrow y'' f_y + c_y$$

4.3.2 Remap

The function remap applies a generic geometrical transformation to an image, it transforms the source image using the specified map:

$$dst(x, y) = src\left(map_x(x, y), map_y(x, y)\right)$$

The function's arguments are four: the source and destination image, and the two lookup map, which indicate where any particular pixel is to be relocated. Obviously the source and destination image are with same size.

As mentioned previously, to avoid blank pixel it is necessary make an interpolation from pixels. Interpolation is an important issue here. Pixels in the source image sit on an integer grid; for example, referring to a pixel at location $(19, 12)$. When these integer locations are mapped to a new image, there can be gaps, either because the integer source pixel locations are mapped to float locations in the destination image and must be rounded to the nearest integer pixel location or because there are some locations to which no pixels at all are mapped. These problems are generally referred to as forward projection problems. To deal with such rounding problems and destination gaps, backwards solves the problem: for each pixel of the destination image found needed source pixel to fill this destination pixel. These

source pixels will almost always be on fractional pixel locations so with the interpolation among the source pixels is necessary to derive the correct value for destination value.

4.4 Rectified image

After the use of the rectification functions, the left and right image are rectified. An example is showed in below, using an image pair of the KITTI dataset [12] we can see the effect of correction on the lens distortion. The example stereo image is below in Figure 4.4.

Figure 4.4 - Example stereo image of the KITTI dataset

In Figure 4.5, in fact, the border of the images are curved opposite at the distortion effect, this phenomenon is due also the alignment of the horizontal lines.

Figure 4.5 - Example of rectified stereo image

To avoid that the black border cause problems at the next module, the image is cropped. The size of clipping is equal for both the images, and whereas the transformation of image is equal given the calibration, we know where to cut

So, in brief, to obtain the above result, two function are used: the initUndistortRectifyMap and remap. The first function is called one time at the beginning of the operation of grab image, and since perform matrix operations request more time than the remap function. The second function each time an image is captured.

Figure 4.6 - Example of rectified and cropped stereo image

To evaluate the times taken by two functions to compute the rectified image, several test has been effected, with image with different size and on different hardware. The using size of image is 640x480 and 320x240.

In Table 4.1 there are average times of initUndistortRectifyMap function.

Size	**Average time on Raspberry (ms)**
320x240	34.27
640x480	125.8

Table 4.1 - Average time of initUndistortRectifyMap function

The times in the previous table are computed for one call of method, obviously, to rectify a stereo image, or else two images, the time have to be doubled.

In Table 4.2 there are average times of remap function.

Size	**Average time on Raspberry (ms)**
320x240	8.7
640x480	31.47

Table 4.2 - Average time of remap function

Since the 640x480 size is four times greater than 320x240, because the width and the height are the double, also the times of the bigger dimensions are approximately four times greater than the others.

Chapter 5

5 Stereo matching

The camera calibration has an accurate and standardized solution, the rectification is a mathematical operation to apply to images. The greatest difficulty of the stereo vision rises from the stereo correspondence, or stereo matching.

5.1 Stereo matching algorithm

The stereo matching is the stereo vision basis, and all scholars seek the best accurate and fast method to identify the correct matching among all pixels pairs of the couple of images.

Although the algorithms are many and different, they have a common line. The steps of a stereo algorithm generally are the following four [13]:

1. Matching cost computation
2. Cost (support) aggregation
3. Disparity computation / optimization
4. Disparity refinement

All stereo matching algorithms require a cost criterion to measure the extent of matching between two pixels. The **matching cost computation** is the stage in which whether the values of two pixels correspond to the same point in a scene is determined. Therefore, the stereo matching cost computation can be defined as a method of determining the parallax values of each point between the left and right images. The matching cost is computed at each pixel for all pixels under consideration. This matching can be performed via a one-dimensional horizontal search if the stereo pairs are accurately rectified. So the rectification plays a

significant role in stereo matching. This is because the search for correspondences can be limited to a line instead of the entire image space, thereby reducing the required time and search range.

Cost aggregation is the most important stage for determining the general performance of a stereo vision disparity map algorithm, especially for local methods. The purpose of cost aggregation is to minimize matching uncertainties. Cost aggregation is needed because the information obtained for a single pixel upon calculating the matching cost is not sufficient for precise matching. Local methods aggregate the matching cost by summing them over a support region.

Instead the **disparity computation** is different for of the two major optimizations approaches: the local approach or the global approach. The **local approach** applies restrictions on a small number of pixels around the pixel under study. They are usually very efficient but sensitive to local ambiguities of the regions. With this approaches, the disparity for each pixel is essentially selected using a local "winner takes all" (WTA) strategy as define by

$$d_p = \arg\min_{d \in D} C'(p, d)$$

The disparity associated with the minimum aggregated cost d_p at each pixel is chosen. $C'(p, q)$ represents the aggregate cost obtained after the matching cost calculation, and D denotes the set of all allowed discrete disparities.

By contrast, in a **global approach**, certain assumptions are made about the depth of field of the scene, which are usually expressed in an energy minimization framework. The bulk of the effort in a global method is expended during the disparity computation phase, and the aggregation step is often skipped.

In the typical global stereo vision formulation, the objective is to find an optimal energy disparity assignment function $d = d(x, y)$ that minimizes

$$E(d) = E_{data}(d) + \beta E_{smooth}(d)$$

where $E_{data}(d)$ represents the matching costs at the coordinates (x, y); the smoothness energy $E_{smooth}(d)$ encourages neighbouring pixels to have similar disparities based on the previous stated assumptions and β is a weighting factor.

The purpose of the **disparity refinement** stage is to reduce noise and improve the disparity maps. Typically, the refinement step consists of regularization and occlusion filling or interpolation. The regularization process will reduce the overall noise through the filtering of inconsistent pixels and small variations among pixels on disparity map. The occlusion filling or interpolation process is responsible for approximating the disparity values in areas in which the disparity is unclear. Typically, occluded regions are filled with disparities similar to those of the background or textureless areas.

5.2 Difficulty of stereo matching

Several problems create some difficulties in research about a right correspondence between left and right camera. For this reason, large number of proposed method are present in the literature, which have been trying to consider all or a big part of problems presented below.

Following, there are the principal difficulties of stereo matching algorithm, with a short description and a meaningful example, to better understand the dubiousness:

- **Specular surfaces**: shiny surface can radically change the texture surfaces.

Figure 5.1 - Example of specular surfaces problem

- **Foreshortening**: an effect that occurs when a surface is viewed at a sharp angle, because each stereo camera has a slightly different view, the image of the surface is more compressed and occupies a smaller area in one view.

Figure 5.2 - Example foreshortening problem

- **Perspective distortions**: is a warping or transformation of an object and its surrounding area.

Figure 5.3 - Example perspective distortions problem

- **Textureless regions**: a region is characterized by a uniform surface.

Figure 5.4 - Example textureless regions problem

- **Repetitive structures and textures**: a region of the image has a repetitive textures.

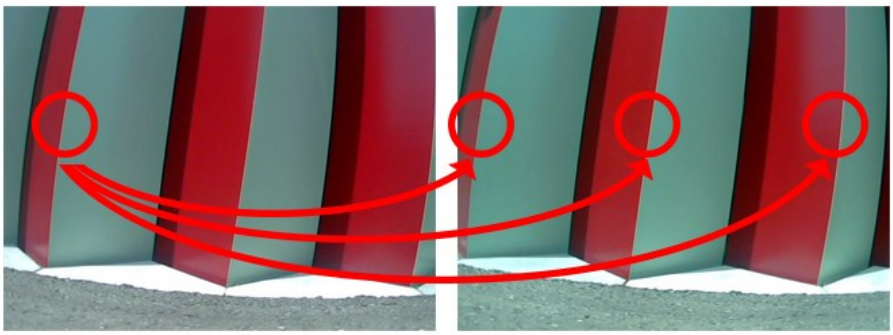

Figure 5.5 - Example repetitive structures example

- **Transparency**: it's possible to see through the transparent object, neglecting it.

Figure 5.6 - Example transparency problem

- **Occlusion**: with different views of the same scene, the objects could hide visible sections from others view.

Figure 5.7 - Example occlusion problem

5.3 State of the art

As above say, the stereo correspondence algorithm are divided in two categories local approaches and global approaches. Local methods can be very efficient, but they are sensitive to locally ambiguous regions in images (e.g., occlusion regions or regions with uniform texture). Global methods can be less sensitive to these problems, since global constraints provide additional support for regions difficult to match locally. However, these methods are more computationally expensive.

The following sections present the techniques of correspondence algorithm to compute the disparity map, separating the local approach and the global approach.

5.3.1 Local approaches

In local methods, the emphasis is on the matching cost computation and on the cost aggregation steps. Computing the final disparities is trivial: simply choose at each pixel the disparity associated with the minimum cost value (WTA strategy). A limitation of this approach is that uniqueness of matches is only enforced for one image, the reference image, while points in the other image might get matched to multiple points. These methods fall into three broad categories: block matching, gradient methods and feature matching.

Block matching methods seek to estimate disparity at a point in one image by comparing a small region about that point with a series of small regions extracted from the other image (the search region). Three classes of metrics are commonly used for block matching: correlation, intensity differences, and rank metrics.

Gradient-based methods, or optical flow, seek to determine small local disparities between two images by formulating a differential equation relating motion and image brightness. Minimize a functional, typically the sum of squared differences, over a small region. In order to do this, the assumption is made that the image brightness of a point in the scene is constant between the two views.

Match metric	Definition
Sum of Absolute Differences (SAD)	$\sum_{u,v} \lvert I_l(u,v) - I_r(u+d,v) \rvert$
Sum of Squared Differences (SSD)	$\sum_{u,v} (I_l(u,v) - I_r(u+d,v))^2$
Normalized Cross-Correlation (NCC)	$\frac{\sum_{u,v}(I_l(u,v) - \bar{I}_l) \cdot (I_r(u+d,v) - \bar{I}_r)}{\sqrt{\sum_{u,v}(I_l(u,v) - \bar{I}_l)^2 \cdot (I_r(u+d,v) - \bar{I}_r)^2}}$
Rank	$\sum_{u,v} (I'_l(u,v) - I'_r(u+d,v))$ $I'_k(u,v) = \sum_{m,n} I_k(m,n) < I_k(u,v)$
Census	$\sum_{u,v} HAMMING(I'_l(u,v), I'_r(u+d,v))$ $I'_k(u,v) = BITSTRING_{m,n}(I_k(m,n) < I_k(u,v))$

Table 5.1 - Common block-matching methods

Block matching and gradient methods are well known to be sensitive to depth discontinuities, since the region of support near a discontinuity contains points from more than one depth. These methods are also sensitive to regions of uniform texture in images. **Feature-based methods** seek to overcome these problems by limiting the regions of support to specific reliable features in the images (e.g., edges, curves, etc.). Among the feature-based method, two approaches are delineated: hierarchical feature matching and segmentation matching.

With the hierarchical technique the matching begins at the highest level of the hierarchy, the surfaces, and proceeds to the lowest, or else the lines. The feature-based hierarchical framework serves much the same purpose as area-based hierarchical frameworks. It allows coarse, reliable features to provide support for matching finer, less reliable features, and it reduces the computational complexity of matching by reducing the search space for finer levels of features. Another feature-based approach is to first segment the images and then match the segmented regions.

5.3.2 Global approaches

The global correspondence methods exploit non-local constraints in order to reduce sensitivity to local regions in the image that fail to match, due to occlusion, uniform texture, and others. The use of these constraints makes the computational complexity of global matching significantly greater than that of local matching.

The global approaches are different: **Dynamic programming** allows resolving optimization problems having an objective function as a sum of monotone non-decreasing functions of resources. In practice, this means that we can infer the optimal solution of a problem using optimal solutions of sub-problems. The dynamic programming applied stereo matching searches for a path of minimal cost through a matrix composed of possible matches. To reduce the complexity, this technique is applied on two sets of points of the same epipolar line. Thus, the stereo correspondence is applied successively to find matchings for all pixels of a line of one image with pixels located on its epipolar line in the other image.

To obtain a global path cost equal to the sum of the partial-paths costs, it is mandatory to use additive costs. We define the local cost for each point in the research zone as the cost of a local stereo matching (SAD, SSD, etc.). Occlusions can be taken into account, making possible to link a set of image pixels with the same pixel in the other image; penalties are considered for these relations (occlusion costs), which will be added to the global cost of any path in the matrix. This formulation presents many inconvenient as the sensibility to the occlusion cost, the difficulty to guarantee inter-lines consistency, and the weak application of constraints on order and continuity, that could be not satisfied.

The dynamic programming can help in finding matchings in poorly textured zones, and in solving some occlusion problems. But this method brings also some weak points, as complexity of calculation, possibility of propagation of a local error through all the research line, and non-consistency of disparity between lines.

This technique not taking into account inter-lines consistency. Hence, they do not use the bi-dimensional nature of the problem. To overcome this drawback, and to take into consideration bidimensionnal continuity constraint, a solution has been

proposed using the graph theory. **Graph cuts,** applied to stereo matching, is formulated like a minimization of an energy function.

The method estimates the optimal disparity map over the entire image. The matching constraints are expressed in a 3D mesh composed of planes, themselves composed of an image of nodes. There is a plane for each level of disparity, and each node represents a matching between two pixels in original images. The 3D mesh is then transformed into a graph of maximal flow by connecting each node to its four neighbours in the same plane by edges called occlusion edge, and with the two nodes in the neighbour planes with edges called disparity edges. Edges are not oriented. The weight of a disparity edge is equal to the mean value of matching costs of the two nodes. For occlusion edges, the weight is multiplied by a constant to control the smoothness of the optimal disparity map. A graph cut will separate the nodes in two sub-sets: the optimal disparity map is constructed by the assignment of each pixel with the bigger value of disparity for which the corresponding node is still connected to the source.

5.4 Stereo correspondence algorithm on sensor

Among the different typologies of stereo matching algorithm, above presented, we choose a suitable algorithm for real environment. Many techniques are good for standard dataset, or simplified scenes. In fact, when these algorithms are used in real world, their accuracy is very low and the computational cost is very high. And, since the sensor has a discrete computing power to elaborate the stereo image, a fast, but accurate algorithm is necessary. The chosen algorithm is “efficient large-scale stereo matching” [14]. This algorithm has good performance with high resolution images, and it is effective in real environment. The used technique is adapted in our case. Although the sensor has a limited computing power, using this method with low resolution images, we can extract quickly a good disparity map.

The method is inspired from the observation that despite the fact that many stereo correspondences are highly ambiguous, some of them can be robustly matched. Assuming piecewise smooth disparities, such reliable “support points” contain valuable prior information for the estimation of the remaining ambiguous disparities. So, the approach begins computing the disparities of a sparse set of

support points, with the use of a full disparity range. The image coordinates of the support points are then used to create a 2D mesh via Delaunay triangulation. A prior is computed to disambiguate the matching problem, making the process efficient by restricting the search to plausible regions. In particular, this prior is formed by computing a piecewise linear function induced by the support point disparities and the triangulated mesh.

5.4.1 Support Points

As support points, we denote pixels which can be robustly matched due to their texture and uniqueness. The method finds that matching support points on a regular grid using the l_1 distance between vectors formed by concatenating the horizontal and vertical Sobel filter responses of 9×9 pixel windows to be both efficient and effective. For robustness it is imposed consistency, or else, the correspondences are retained only if they can be matched from left-to-right and right-to-left. To get rid of ambiguous matches, all points with ratio between the best and the second best match exceeds a fixed threshold, $\tau = 0.9$, are eliminated. Spurious mismatches are removed by deleting all points which exhibit disparity values dissimilar from all surrounding support points. To cover the full image, we add additional support points at the image corners whose disparities are taken to be the ones of their nearest neighbours.

5.4.2 Generative model for stereo matching

With the support points calculated, here is described the probabilistic generative model which.

More formally, let $S = \{s_1, \dots, s_M\}$ be a set of robustly matched support points. Each support point, $s_m = (u_m, v_m, d_m)^T$, is defined as the concatenation of its image coordinates, $(u_m, v_m) \in \mathbb{N}^2$, and its disparity, $d_m \in \mathbb{N}$. Let $O = \{o_1, \dots, o_N\}$ be a set of image observations, with each observation $o_n = (u_n, v_n, f_n)^T$ formed as the concatenation of its image coordinates, $(u_n, v_n) \in \mathbb{N}^2$, and a feature vector, $f_n \in \mathbb{R}^Q$, e.g., the pixel's intensity or a low-dimensional descriptor computed from a small neighbourhood. We denote $o_n^{(l)}$ and $o_n^{(r)}$ as the observations in the left and right image respectively. Without loss of generality, in the following we consider the left image as the reference image.

Assuming that the observations $\left\{o_n^{(l)}, o_n^{(r)}\right\}$ and support points S are conditionally independent given their disparities d_n, the joint distribution factorizes

$$p\left(d_n, o_n^{(l)}, o_n^{(r)}, S\right) \propto p\left(d_n|S, o_n^{(l)}\right) p\left(o_n^{(r)}|o_n^{(l)}, d_n\right)$$

with $p\left(d_n|S, o_n^{(l)}\right)$ the prior and $p\left(o_n^{(r)}|o_n^{(l)}, d_n\right)$ the image likelihood. The graphical model is depicted in Figure 5.8.

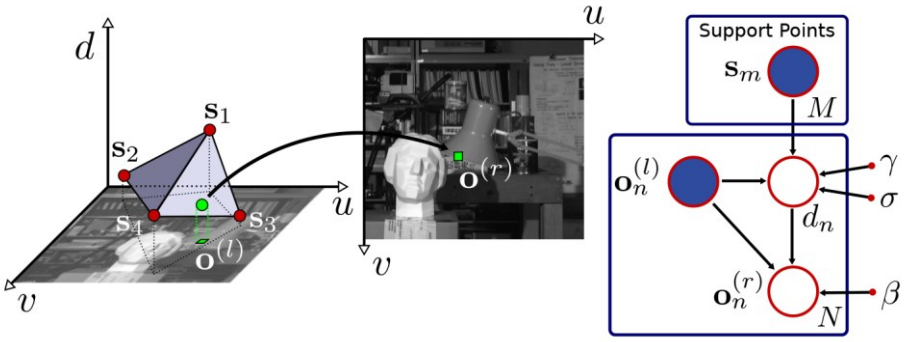

Figure 5.8 - Graphical model and sampling process

In particular, we take the prior to be proportional to a combination of a uniform distribution and a sampled Gaussian

$$p\left(d_n|S, o_n^{(l)}\right) \propto \begin{cases} \gamma + exp\left(-\frac{\left(d_n - \mu\left(S, o_n^{(l)}\right)\right)^2}{2\sigma^2}\right) & if\ |d_n - \mu| < 3\sigma\ \vee d_n \in \mathbb{N}_S \\ 0 & otherwhise \end{cases}$$

with $\mu\left(S, o_n^{(l)}\right)$ a mean function linking the support points and the observations, and N_S the set of all support point disparities in a small $20x20$ pixel neighbourhood around $\left(u_n^{(l)}, v_n^{(l)}\right)$. We gain efficiency by excluding all disparities farther than 3σ from the mean. The condition $d_n \in \mathbb{N}_S$ enables the prior to locally extend its range to better handle disparity discontinuities in places where the linearity assumption might be violated.

We express $\mu\left(S, o_n^{(l)}\right)$ as a piecewise linear function, which interpolates the disparities using the Delaunay triangulation computed on the support points.

For each triangle, we thus obtain a plane defined by

$$\mu_i\left(o_n^{(l)}\right) = a_i u_n + b_i v_n + c_i$$

where i is the index of the triangle the pixel $(u_n; v_n)$ belongs to, and $o_n = (u_n, v_n, f_n)^T$ is an observation. For each triangle, the plane parameters (a_i, b_i, c_i) are easily obtained by solving a linear system. Hence, the mode of the proposed prior, μ, is a linear interpolation between support point disparities, serving as a coarse representation.

We express the image likelihood as a constrained Laplace distribution

$$p\left(o_n^{(r)}|o_n^{(l)}, d_n\right) \propto \begin{cases} exp\left(-\beta\left\|f_n^{(l)} - f_n^{(r)}\right\|\right) & if \begin{pmatrix} u_n^{(l)} \\ v_n^{(l)} \end{pmatrix} = \begin{pmatrix} u_n^{(r)} + d_n \\ v_n^{(r)} \end{pmatrix} \\ 0 & otherwhise \end{cases}$$

where $f_n^{(l)}$ and $f_n^{(r)}$ are feature vectors in the left and right image respectively, and β is a constant. The if-condition ensures that correspondences are located on the same epipolar line and matched via the disparity d_n. In this equation, there is only one observation with non-zero probability for each d_n. The used features f_n are taken as the concatenation of image derivatives in a $5x5$ pixel neighbourhood around (u_n, v_n), computed from Sobel filter responses.

An advantage of having a generative model is that we can use it to draw samples. Given the support points and an observation in the left image, samples from the corresponding observation in the right image can be obtained as follows:

1. Given S and $o_n^{(l)}$ draw a disparity d_n from $p\left(d_n|S, o_n^{(l)}\right)$
2. Given $o_n^{(l)}$ and d_n draw an observation $o_n^{(r)}$ from $p\left(o_n^{(r)}|o_n^{(l)}, d_n\right)$

5.4.3 Disparity Estimation

In order to estimate the disparity map given the left and right images, we rely on maximum a-posteriori (MAP) estimation to compute the disparities

$$d_n^* = argmax\, p\left(d_n|o_n^{(l)}, o_1^{(r)}, \ldots, o_N^{(r)}, S\right)$$

where $o_n^{(l)}, o_1^{(r)}, \dots, o_N^{(r)}$ denotes all observations in the right image which are located on the epipolar line of $o_n^{(l)}$. The posterior can be factorized as

$$p\left(d_n | o_n^{(l)}, o_1^{(r)}, \dots, o_N^{(r)}, S\right) \propto p\left(d_n | S, o_n^{(l)}\right) p\left(o_1^{(r)}, \dots, o_N^{(r)} | o_n^{(l)}, d_n\right)$$

The observations along the epipolar line on the right image are structured, i.e., given a disparity associated with $o_n^{(l)}$, there is a deterministic mapping to which observations have non-zero probability on the line. We capture this property by modelling the distribution over all the observations along the epipolar line as

$$p\left(o_1^{(r)}, \dots, o_N^{(r)} | o_n^{(l)}, d_n\right) \propto \sum_{i=1}^{N} p\left(o_i^{(r)} | o_n^{(l)}, d_n\right)$$

Plugging the two equations in previous section, and taking the negative logarithm yields an energy function that can be easily minimized

$$E(d) = \beta \left\| f^{(l)} - f^{(r)}(d) \right\| - log\left[\gamma + \exp\left(-\frac{\left[d - \mu(S, o^{(l)}\right]^2}{2\sigma^2}\right)\right]$$

with $f^{(r)}(d)$ the feature vector located at pixel $\left(u^{(l)} - d, v^{(l)}\right)$. Note that from the definition of the image likelihood, the energy $E(d)$ is required to be evaluated only if $|d - \mu| < 3\sigma$, or d is an element of the neighboring support point disparities.

A dense disparity map can be obtained by minimizing the energy. Importantly, this can be done in parallel for each pixel as the support points decouple the different observations.

Then, the above approach is applied on both images, and perform a left/right consistency check to eliminate spurious mismatches and disparities in occluded regions. Finally, the small segments with an area smaller than 50 pixels are removed.

5.4.4 Parallelization

The operations above presented require a substantial amount of time to be execute, but we have few operations to perform on a big quantity of data. This situation directs us to the SIMD.

Single Instruction, Multiple Data, namely SIMD, indicates computers with multiple processing elements that perform the same operation on multiple data points simultaneously. Thus, such machines exploit data level parallelism, but not concurrency: there are simultaneous (parallel) computations, but only a single process (instruction) at a given moment. SIMD is particularly applicable to common tasks like adjusting the contrast in a digital image. Most modern CPU designs include SIMD instructions in order to improve the performance of multimedia.

Raspberry PI has an ARM like CPU, more specifically an ARM Cortex-A7. The processors of A series own the NEON technology.

NEON is the implementation of the SIMD used in ARM processors. NEON technology can accelerate multimedia and signal processing algorithms such as video encode/decode, 2D/3D graphics, gaming, audio and speech processing, image processing, telephony, and sound synthesis by at least 3x.

So, the NEON is used with different operations in stereo matching algorithm. One of the operations most used which exploits the power of the NEON technology is the function which individuate the correct disparities of the support points. To find the exact correspondence of a support points, all possible disparities, in the range of the disparity, must be check, at the end the disparity for a single points is calculated. This operation has to be done for all supports points. How we can understand this operation is easily parallelized.

5.5 Method accuracy

To test the method accuracy, the authors of the article [13] has organized all developed works until now in single site [15]. This site provides a list of all stereo matching algorithm, with their accuracy for single tested images, the overall average and the reference at the article describes the algorithm. Beyond this there are some tools to verify the accuracy of a stereo matching algorithm, but to use it we need of a dataset with the ground truth. Also different dataset, with different resolutions, are provided from this site to make the test.

The measured accuracy computes the difference between the disparity value in the ground truth and the disparity value in the produced image of the algorithm. If this difference is lower than a threshold, the disparity value is considered correct otherwise is presented an error. The sum of all errors is divided for the number of pixel of the image, getting, thus, a percentage of error.

In the Table 5.2 are showed the results for a subset of the Middlebury dataset of 2006.

Test image	Error (threshold=1)	Error (threshold=2)
Cones	5.0%	2.7%
Teddy	11.5%	7.3%
Art	13.5%	8.7%
Aloe	5.0%	3.0%
Dolls	11.0%	5.3%
Baby3	10.8%	4.5%
Cloth3	1.4%	0.9%
Lamp2	17.5%	10.4%
Rock2	1.9%	1.0%

Table 5.2 - Result on Middlebury dataset 2006

In the following figures, there are the match between the disparity map computed and the ground truth of the image Teddy.

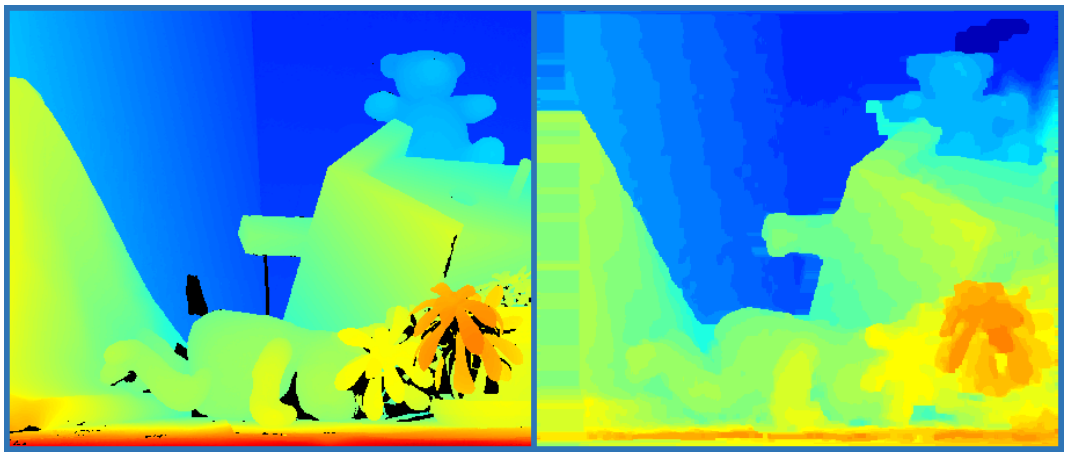

Figure 5.9 - Comparison with ground truth of the Teddy image

With the new Middlebury dataset of 2014 the error percentage is increased, like we can see in Table 5.3.

Test image	Error (threshold=1)	Error (threshold=2)
Australia	33.3%	12.5%
Motorcycle	31.5%	13.9%
Classroom	43.1%	23.7%
Computer	29.5%	20.4%
Djembe	30.5%	11.0%
Newkuba	45.3%	28.6%
Stairs	54.9%	33.3%

Table 5.3 - Result on Middlebury dataset 2014

While in Figure 5.10 there is the comparison from the disparity ground truth, on the left, and the disparity calculated by the algorithm, on the right.

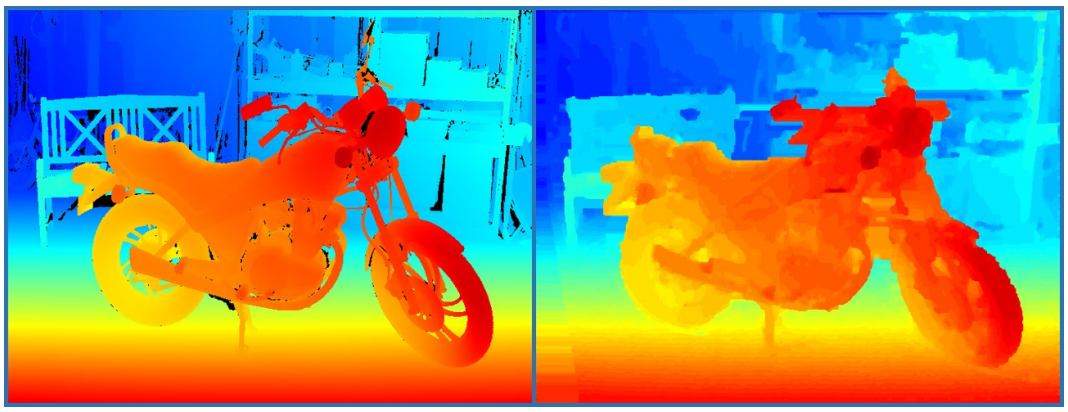

Figure 5.10 - Comparison with ground truth of the Motorcycle image

The average error of the stereo matching algorithm, considering the above tables is 11.7%.

Chapter 6

6 Test and results

In this section we describe the different test with the 3D sensor, showing the all obtained results.

The follow test are carried with different resolutions, 640x480 and 320x240, comparing the execution time of all modules on a PC and on Raspberry PI.

The Raspberry PI was presented in a previous chapter. The used PC has an Intel Core i5 2.6GHz like processors and 8GB of RAM. An additional the SIMD instructions on Intel processors are called SSE2. So a wrapper from two architecture was carried out

6.1 Stereo acquisition

The stereo acquisition module was tested for one hour without have problems. The difference between acquisition by Raspberry and PC are inexistent, and so in Table 6.1 are recur the time of acquisition from a stereo camera in different resolutions.

Resolution image	**Time stereo camera (ms) on Raspberry PI**	**Time stereo camera (ms) on PC**
320x240	47	47
640x480	57	57

Table 6.1 - Time acquisition stereo camera

An example of the acquisition from the stereo camera is showed in Figure 6.1

Figure 6.1 - Example stereo image

How we can see in reference at the skirting on the bottom of the image and the shelf on the top of the image, the introduced distortion from the lens is remarkable and the next operation of rectification is essential for a good success of the 3D sensor

6.2 Rectification

The rectification is essential after the previous acquisition. The Figure 6.2 shows the result of the rectification of the above image.

Figure 6.2 - Example rectified stereo image

The skirting and the shelf, now, are straight and don't exhibit curvatures. We remove the black borders, getting the cropped images.

Figure 6.3 - Example rectified and cropped stereo image

Now the left and right image are overlaid. To distinguish the two figures, the left image has accented blue colours, instead, the right image has accented red colours. In this figure we can better see the aligned horizontal lines. Points at different distance from camera are emphasize and with a yellow line you can notice the good alignment of these points.

Figure 6.4 - Comparison between left and right image

The time to rectify both the images are illustrated above. Now we want compare the request time on the Raspberry PI and the request time on PC to rectified both the images. From the follow tables there is an abyss from the two hardware. The

difference between one and the other is fifteen times in favour of the PC. With this we understand the potential of the used method on a more powerful machine.

Size	Average time on Raspberry (ms)	Average time on PC (ms)
320x240	68.54	4
640x480	251.6	15.7

Figure 6.5 - Comparison time of initUndistortRectifyMap function

Size	Average time on Raspberry (ms)	Average time on PC (ms)
320x240	17.4	1.24
640x480	62.94	4.2

Figure 6.6 - Comparison time of remap function

6.3 Stereo matching

Finally, the stereo matching module use all result from the previous section to elaborate the disparity map. The Figure 6.7 shows, in false colour, the disparity map of the reference image presented in previous part. With the false colour, the red indicate the object near the camera, the orange indicate the object at middle distance from the camera and the blue far object. How we can guess in the stereo image, the right hand of the author (in the figure) is the area of body nearest to the camera, then the left hand is in the middle and total body is in the back. The wall is the background of the image and it is represent with a colour blue.

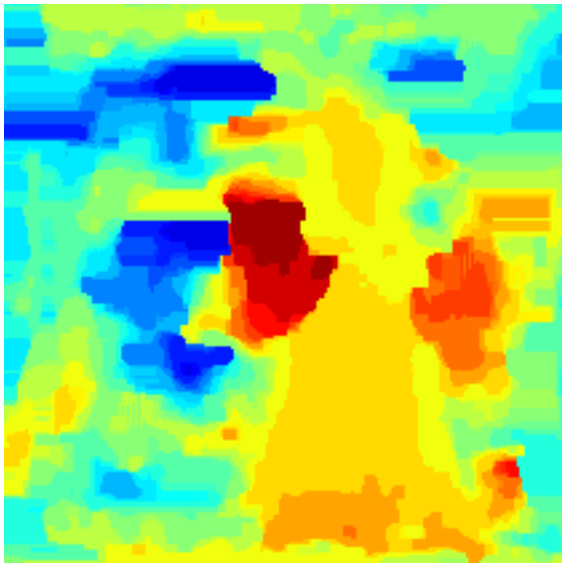

Figure 6.7 - Example disparity map

The computation time of this operations is much bigger the previous functions. In fact in the follow table we can see the long time to extract the disparity map.

Size	Average time on Raspberry (ms)	Average time on PC (ms)
320x240	300	36
640x480	1400	180

Figure 6.8 - Comparison time of the disparity map extraction

6.4 Profiling

Finally, we can compute the total time to execute all the module and then extract the disparity map. The test was carried out with two different resolutions: 320x240 and 640x480.

In the tables, presented to the next page, there are the times for each module and the total.

Operations	Using Raspberry (ms)	Using PC (ms)
Acquisition	57	57
Rectification	62.94	4.2
Disparity calculation	1400	180
Total	1519.94	241.2

Table 6.2 - Execution time for 640x480 resolution

The framerate reached on Raspberry with resolutions 640x480 to extract disparity map is 0.65 fps.

Operations	Using Raspberry (ms)	Using PC (ms)
Acquisition	47	47
Rectification	17.4	1.24
Disparity calculation	300	36
Total	364.4	84.24

Table 6.3 - Execution time for 320x240 resolution

The framerate reached on Raspberry with resolutions 320x240 to extract disparity map is 3 fps.

Since the framerate with the resolutions 640x480 is too high to can use in a real application, we prefer use the resolution 320x240,that, also it has a low framerate, we can think to use this sensor in scene where the environment change slowly.

Chapter 7

7 Conclusion

In this work of thesis, we have presented a low cost 3D sensor, able to extract a disparity map. This sensor has been exposed in all parts. Each part has been previously described, providing a brief context to understand better the technique used. All modules, of which it is provided the sensor, are necessary to a good realization of the sensor.

We started with the calibration, passing from the acquisition, the rectification and, last and more important, the matching operations to extract the disparity map.

The operations performed by the sensor are linked about the idea to minimize the computation load, and to obtain a high accuracy, because the core of the sensor is Raspberry PI, and the limited resources clash with the difficulty of the stereo vision. After all, considering all problems, the stereo matching algorithm has a good accuracy, as say in above of 11.7% of error. The framerate of the stereo camera settled on 3 fps.

The performance may be improved using hardware with a new and more powerful CPU, like the new version of Raspberry PI.

Future developments of this work could concern the completion of the 3D sensor, making it a plug and play sensor, usable immediately with a rough configuration. Or again to develop a library with all functions implemented of the sensor and to transform the sensor in a tool which performs only the operations requested by the users.

Bibliography

[1] J. Knight, “Robot Navigation by Active Stereo Vision,” *Active Vision Laboratory,* 2000.

[2] S. B. Goldberg, M. W. Maimone and L. Matthies, “Stereo Vision and Rover Navigation Software,” *IEEE Aerospace Conference Proceedings,* 2002.

[3] K. Terada, D. Yoshida, S. Oe and J. Yamaguchi, “A method of counting the passing people by using the stereo images,” *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on,* vol. 2, pp. 338-342, 1999.

[4] S. Mattoccia and P. Macri', “3D glasses as mobility aid for visually impaired people,” *European Conference on Computer Vision,* pp. 539-554, 2014.

[5] Tsai and Roger, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses,” *IEEE Journal on Robotics and Automation,* pp. 323-344, 1987.

[6] Z. Zhang, “Camera calibration with one-dimensional objects,” *IEEE transactions on pattern analysis and machine intelligence 26.7,* pp. 892-899, 2004.

[7] S. J. Maybank and O. D. Faugeras, “A theory of self-calibration of a moving camera,” *International Journal of Computer Vision 8.2,* pp. 123-151, 1992.

[8] G. Bradski, “Opencv Library,” in *Dr. Dobb's Journal of Software Tools*, 2000.

[9] J. J. Moré, “The Levenberg-Marquardt algorithm: implementation and theory,” *Numerical analysis,* pp. 105-116, 1978.

[10] BiPM, IEC and IFCC, ILAC and ISO, IUPAC and IUPAP and OIML, “International Vocabulary of Metrology--Basic and General Concepts and

Associated Terms," 2008. [Online]. Available: http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf.

[11] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 2015. [Online]. Available: https://www.vision.caltech.edu/bouguetj/calib_doc/.

[12] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR),* 2013.

[13] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International journal of computer vision,* pp. 7-42, 2002.

[14] A. Geiger, M. Roser and R. Urtasun, "Efficient large-scale stereo matching," *Asian conference on computer vision,* pp. 25-38, 2010.

[15] D. Scharstein, R. S. Szeliski and H. Hirschmüller, "The Middlebury Computer Vision Pages," 2016. [Online]. Available: http://vision.middlebury.edu/stereo/.

Acknowledgements / Ringraziamenti

Finalmente questa odissea è giunta al termine. Chi mi è stato vicino, e chi ha seguito il mio percorso, ha potuto assistere alle vicende protagoniste di questo lavoro di tesi e di tutto il periodo universitario della laurea magistrale. E proprio chi mi è stato vicino, chi mi ha aiutato e creduto in me ho voglia di ringraziare.

Ringrazio il mio relatore, prof. Vento, per l'offerta delle tante possibilità che permettono a noi studenti di crescere come persona.

Grazie al prof. Tabbone per l'accoglienza e la disponibilità offerta durante il tempo trascorso a Nancy.

Un Grazie immenso al mio correlatore Antonio Greco per l'aiuto e il supporto che mi ha dato durante tutto questo lungo lavoro di tesi. Grazie per avermi sempre dato una mano e il sostegno che avevo bisogno.

Ringrazio tutti i ragazzi dell'università, dal primo all'ultimo, per i momenti di divertimento, la gioia e l'aiuto che ci siamo scambiati in questi anni. E, in particolare dico grazie ad Antonio Salvati, Salvatore, Daniele e Gianluigi.

Un particolare ringraziamento va al mio compagno di progetti e amico Gino, per l'aiuto, la condivisione e a volte il supporto morale che mi ha donato in questo mio cammino universitario.

Grazie ai compagni Antonio e Giovanni, per aver condiviso gioie e difficoltà del periodo di Erasmus.

Ringrazio gli amici di una vita, Fabrizio, Marco, Luca e Cristian e mio cugino Fabio per le serate di svago e le cene condivise insieme.

Grazie a nonna Filomena, zia Teresa e zia Titina per tutto l'amore che mi avete mostrato e per essere sempre state orgogliose di me.

Un saluto e un bacio fortissimo a nonna Anna, che inaspettatamente mi ha messo davanti una delle prove più difficili della vita, ma che comunque voglio e vorrò sempre bene.

Ringrazio tutti i miei zii materni e paterni, con particolare attenzione a zio Bruno, per il supporto, l'affetto e il sostegno donatomi.

Concludendo vorrei ringraziare, con tutto il mio cuore, le persone, e non, che sono giorno dopo giorno al mio fianco.

Un immenso grazie alla mia mamma e al mio papà, per aver creduto in me, avermi sempre sostenuto, economicamente e soprattutto moralmente, ed avermi aiutato in tutto il percorso scolastico, che credo che a questo punto sia giunto al termine. Grazie di cuore per l'amore che tutti i giorni sapete incondizionatamente donarmi, grazie per i pochi, ma giusti, rimproveri che mi aiutano a crescere e migliorare. Quindi voglio dirvi semplicemente Grazie.

Ringrazio mio fratello Luca che, con i suoi modi di fare, mi offre distrazioni e momenti di svago. Grazie, ancora, per il tanto amore che mi mostra tutti i giorni.

Un biscottino per ringraziare il nostro cagnolino Alex, per la gioia che sa scodinzolare.

Grazie a Mara e Amedeo per la serenità che sanno donare ai miei genitori e l'affetto che mi danno.

Un Grazie ad Anna Maria, Giuseppe e Gianmarco, per avermi sempre supportato ed accolto nella loro famiglia come un figlio e un fratello.

Ed infine non so come ringraziare la mia dolce metà, Arianna. Ha supportato con gioia la mia volontà di effettuare un periodo in Erasmus e ha sofferto in silenzio per non farmi appesantire questa mia scelta. È stata il mio trampolino per avermi spinto in ogni mia scelta, è stata il mio carburante per avermi rifornito di consigli, è stata il mio paracadute per aver attutito tutte le mie difficoltà. Ha saputo amarmi e perdonarmi, rimproverarmi ed aiutarmi. Ha condiviso con me gioie e dolori. Mi ha alleggerito da alcuni problemi e mi ha ricaricato di energie per andare avanti. Sempre lì, al mio fianco, giorno dopo giorno, ora dopo ora. Un'amica, una fidanzata, una compagna di vita che non cambierei con nulla nell'universo. Sono cresciuto insieme a lei e se sono così oggi è anche per merito, o colpa, sua. Grazie dal più profondo del mio cuore, immensamente Grazie, e ancora Grazie.

Roberto