Scuola Superiore Sant'Anna & Universita' di Pisa

Scuola di Ingegneria
M.Sc. in Embedded Computing Systems

Master's Thesis

Design and Development of a Novel Computer Vision System for Large Plant Inspections

An application in the Railway Industry

Candidate:
Matteo Pampana

Thesis Advisors:
Prof. C.A. Avizzano
Dr. E. Ruffaldi

A.A. 2015-2016

Alla mia Famiglia,
A chi ha creduto in me

To my family,
To those who believed in me

Abstract

Computer Vision is one of the emerging fields in computer science. It presents a lot of challenges, that are not so easy to deal with. In this thesis it is shown the process of design and development of a computer vision system, that is able to reconstruct the three-dimensional surface of a train. This work describes the process of analysis and design of the vision system, the process of hardware's selection in such a way to meet the requirements, and the architectural aspects in order to manage a big quantity of data that comes from the vision system. The core of this work is the development of a 3D Structured Light scanner, that is able to reconstruct the 3D Geometry of the external surface of a train. It shows the importance of the image pre-processing in order to improve the accuracy and the robustness of the reconstruction. Finally, it presents some results of the tests did, at the moment, in the laboratory.

Contents

Chapter 1

Introduction

"As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and shading that play across its surface and effortlessly segment each flower from the background of the scene. Looking at a framed group portrait, you can easily count (and name) all of the people in the picture and even guess at their emotions from their facial appearance.
Despite all of the advances in computer vision, the dream of having a computer interpret an image at the same level as a two-year old (for example, counting all of the animals in a picture) remains elusive. Why is vision so difficult? In part, it is because vision is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify the solution. We must therefore resort to physics-based and probabilistic models to disambiguate between potential solutions. However, modeling the visual world in all of its rich complexity is a very complex problem.
Computer vision tries to describe the world that we see in one or more images and to reconstruct its properties, such as shape, illumination, and color distributions. It is amazing that humans and animals do this so effortlessly, while computer vision algorithms are so error prone. People who have not worked in the field often underestimate the difficulty of the problem."
These are the words of Richard Szeliski, one of the most important researcher in the computer vision area. Computer Vision is hard. The aim of this thesis

is to design and develop a system that tries to push a little up the limits of the structured light reconstruction systems. We propose a solution to perform the inspection of large areas, on a vehicle that is moving and in far than perfect lighting conditions. Some smart image processing solutions are described in this work in order to enhance the quality and the robustness of the 3D Reconstruction.
This thesis shows the process of analysis, design and development with respect to a railway industry application. Actually, this work comes as a part of the project that is described in the next chapters. At the moment, the project is in its early phase and it is in progress.

1.1 State-Of-The-Art Computer Vision Applications

Computer vision is used today in a wide variety of real-world applications, which include:

- Optical character recognition (OCR): reading handwritten postal codes on letters and automatic number plate recognition (ANPR);
- Machine inspection: rapid parts inspection for quality assurance using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts or looking for defects in steel castings using X-ray vision;
- Retail: object recognition for automated checkout lanes;
- 3D model building: fully automated construction of 3D models from aerial photographs used in systems such as Bing Maps;
- Medical imaging: registering pre-operative and intra-operative imagery or performing long-term studies of people's brain morphology as they age;

- Automotive safety: detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar or lidar do not work well
- Match move: merging computer-generated imagery (CGI) with live action footage by tracking feature points in the source video to estimate the 3D camera motion and shape of the environment. Such techniques are widely used in Hollywood;
- Motion Capture: using retro-reflective markers viewed from multiple cameras or other vision-based techniques to capture actors for computer animation;
- Surveillance: monitoring for intruders, analyzing highway traffic, and monitoring pools for drowning victims;
- Fingerprint recognition and biometrics: for automatic access authentication as well as forensic applications.

David Lowe's Web site of industrial vision applications (`http://www.cs.ubc.ca/spider/lowe/vision.html`) lists many other interesting industrial applications of computer vision.

1.2 Work Organization

The thesis is structured as follows.
Chapter 2 defines the characteristics of the railway application useful to define the requirements of the vision system. It shows the characteristics of the carriages and the railway systems that our vision system has to deal with. From this analysis, has been obtained a list of requirements that are used to size the optical devices.
Chapter 3 describes the process of design of the vision system and the choice of the piece of hardware starting from the application's requirements. This includes also the tuning of some parameters necessary to meet the requirements.
Chapter 4 is dedicated to the theoretical background necessary to understand this work. In this section is described the camera pinhole model, some concepts related to the stereo vision and the epipolar geometry, and a brief taxonomy

of the structuring light systems.
Chapter 5 describes the calibration process for the digital camera and for a projector. This part is crucial to obtain the maximum accuracy in the reconstruction and in the measure.
Chapter 6 analyzes in details the core of the thesis: the process of design and development of a 3D Structured Light Scanner for large moving areas. The chapter will analyze the pipeline of pattern detection, in such a way to make it robust to noise and lighting conditions, and the pipeline of three-dimensional reconstruction.
Chapter 7 makes an overview of the software architectural aspects of the work. This includes a description of the software layers architecture of the system, the selection of libraries and languages to develop the system and the software management of the pipelines with ROS, the Robotic Operating System.
Chapter 8 shows some tests and results of the implemented algorithms.
Finally, Chapter 9 conclude the thesis with an analysis of the work done, and a list of challenges and goal for future work.

Chapter 2

Problem Definition

In this thesis it will be shown the design of a Computer Vision System related to a particular application: a maintenance system for the railway industry. The core of the ordinary maintenance in the railway industry is represented by the correct inspection of the lateral train's coachwork and of the bottom part, where trains contain crucial system such as cooling grids, brakes, and so on.
In particular, being able to capture the biggest possible section of the train with a fixed computer vision system maximizes speed and efficiency. In fact, a fixed sensing system neither needs operators moving it, nor stopping the train to perform the inspection.
This is the background in which we will analyze the functional properties and the requirements of the Vision System.

2.1 Analysis of the system

First of all, it is important to define clearly the geometry of the train. Thus, the design of the Computer Vision System heavily depends on the characteristics of the application. In this case, we want to recover the largest possible area with one single frame. To quantify the term *largest possible area* is important to look at the geometrical properties of a train coachwork.

2.1.1 Loading Gauge

The *loading gauge* (in Italian *"Sagoma Limite"*) is a disposition that defines the maximum height and width for railway vehicles and their load to ensure safe passage through bridges, tunnels and other structures. The definition of the loading gauge takes into account, not only the geometric aspect, but also vibrations phenomena, dumpers, tolerances and safety distances. At the present, the international disposition is slightly different respect to the Italian, which is more conservative.

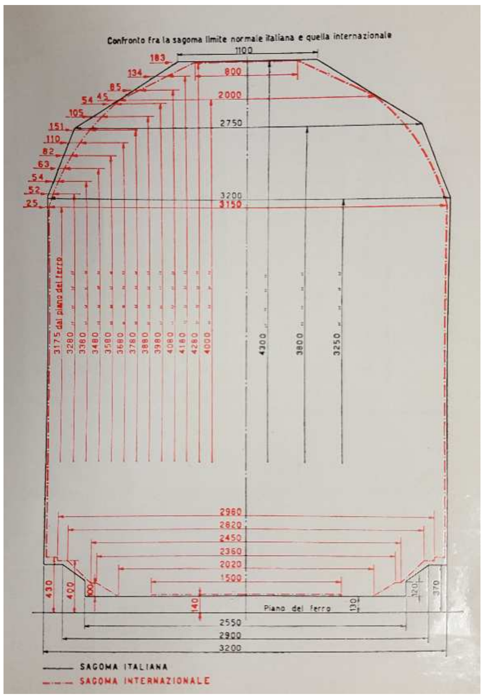

Confronto fra la sagoma limite normale italiana e quella internazionale
Piano del ferro
SAGOMA ITALIANA
SAGOMA INTERNAZIONALE

Figure 2.1: Comparison between the Italian and the International loading gauge

As it can be seen from Figure 2.1, the carriage must not be taller than $4,28m$ from the iron plane and not be larger than $3,15m$. This information is useful because defines the bounding box of carriages, and consequently the minimum distances allowed to place the vision system with respect to the vehicle. Moreover, it gives us an upper bound on the height of the carriage that is important to perform the optical design of the components.

2.1.2 Carriages Dimensions and Characteristics

The fundamental section dimensions - i.e., height and width - of carriages are determined starting from the technical documentation and related to the vehicle called *"Minuetto"*.
These dimensions are the reference for the optical design of the components.

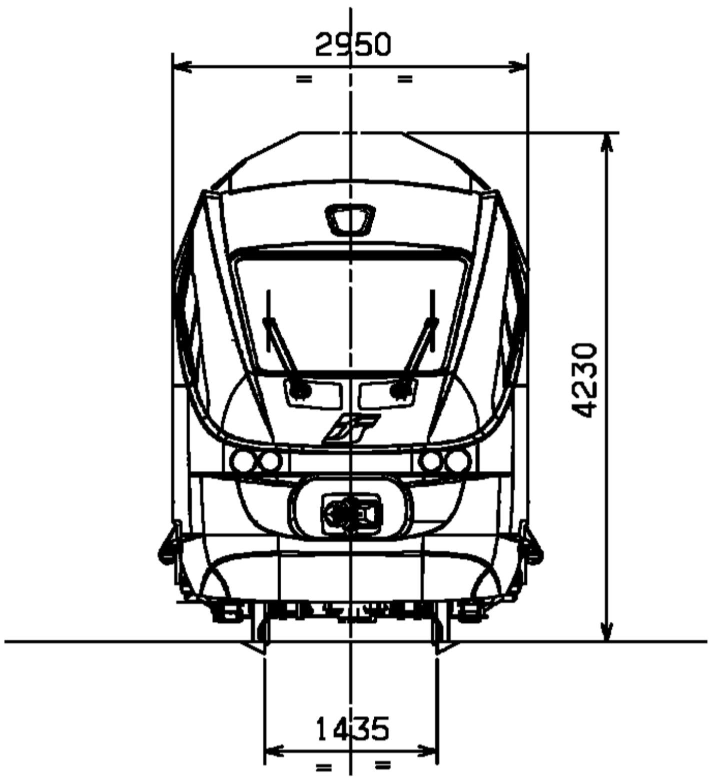


Figure 2.2: Frontal section of the vehicle "Minuetto".

The reference height of the train, from the wheel to the imperial, is equal to 4230 mm, while the width of the coachwork is equal to 2950 mm.

The total length of each "Minuetto" module is equal to 51900mm, and it is composed by two traction units and a carriage in the middle. The imperial of these models is not homogeneous but it is composed by a set of boxes, equipped with doors or grids, containing some electrical power schemes.

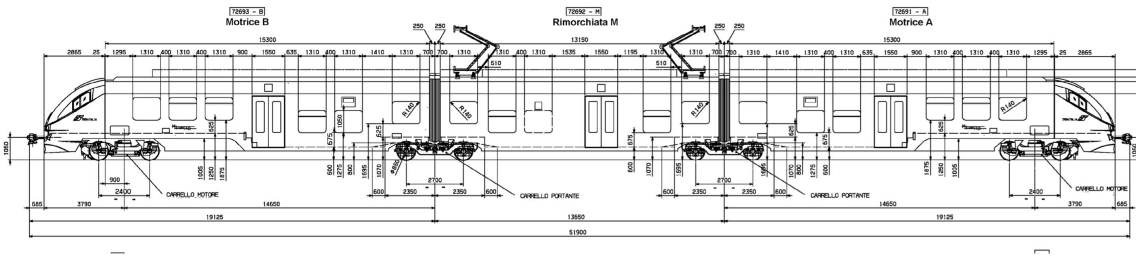


Figure 2.3: Lateral section of a "Minuetto" module.

2.2 Functionalities

The vision system is dedicated to the inspection of the imperial, the lateral surfaces, and the bottom part of a complete train. The main goal of the system is to produce a visual documentation of the train, integrate it with a numerical model, compare it with respect to a CAD model or an equivalent model previously acquired.
The goal of the system is to detect possible damages, failures and critical factors related to the external surface of the carriage.
The final system must provide methods to:

- Detect damages related to relevant surface deformations;
- Detect failures at the cooling grids on the imperial of the traction unit;
- Detect the presence of the cooling grids;
- Detect Graffiti;

Moreover, to make easier the analysis of the train the system has to provide methods to:

- Recognize the serial number of the carriages and the traction unit
- Individuate big movements of the boxes in the bottom part of the train
- Detect the detachment or the absence of flexible parts (cables, tubes)
- Detect glass damages

2.3 Functionality Map

With respect to the functionalities, the system has to provide the following modules:

- A 3D Reconstruction System
- A 3D Matching System
- A system to align different 3D patches
- A system to align different camera frames: a sort of bird's eye view of the entire train
- A 2D Matching System

To be clearer, it is possible to map the functionalities and the modules in a table, that is shown below:

	3D Rec.	3D Match	3D Align	2D Match	2D Align
Surface Damages	x	x	x		
Detachment of flex.parts	x	x	x		
Cooling Grids	x	x	x		
Protection Boxes	x	x	x		
Doors	x	x	x		
Graffiti				x	x
Glass Damages				x	x
Serial Number Scan				x	x

Table 2.1: Functionality Map: this table shows the modules needed to develop the functionalities

This thesis shows how to design and develop a 3D Reconstruction System, since the majority of the functionalities require this module and is the more complex area of the project. It shows an innovative approach for the hardware architecture design of a 3D Structured Light Scanner, in the management of the software's pipeline and in the image processing.

Chapter 3

System and Parameter Design

This chapter explains the choice made for the hardware selection. There a lot of devices with many different attributes and special characteristics over the market, we made the choice that fits best our application case. In order to motivate the process of selection, this chapter lists the requirements of the vision system and provides a brief explanation of the basic functioning principles of the devices taken into account.

3.1 Requirements

The requirements for the computer vision system are:

- One frame must cover an area of at least length 5m, and height 5m.
- The system must capture the train while it is moving at a speed of 5m/s.
- The image processing must be done off-line.
- The accuracy of the depth estimation must be in the order of the centimeters.

The first requirement comes from the analysis of the scenario depicted in the chapter 2, in particular from the characteristics of the "Minuetto" carriage. The second one comes from the fact that trains travel at most at that speed inside the maintenance site, i.e., where we want to place the system. The third one comes from the fact that the quantity of information is huge and it is, at the moment, both unrealistic and unnecessary to perform the processing in real-time. The requirement regarding the accuracy comes from the application: though external surface of a carriage may have some very small defects, actually they do not compromise the correct functioning of the train.

3.2 Hardware Selection

3.2.1 Digital Camera

Basic principles

The term digital means that the camera do not use anymore the film to capture an image. Digital cameras use a photosensor. The most common is the Charge Coupled Device sensor (CCD). Other types of sensors are the CMOS. These sensors cannot give any information about the colors by themselves. For this purpose, color digital cameras use a special filter. The most common is the Bayer Filter.

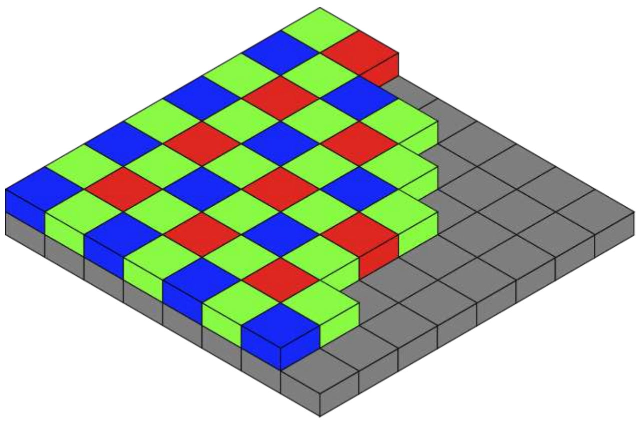


Figure 3.1: Bayer Filter

A Bayer filter is a color filter array for arranging RGB color filters on a square grid of photosensors. The filter pattern is 50% green, 25% red and 25% blue. It uses twice as many green elements as red or blue to mimic the physiology of the human eye. The luminance perception of the human retina uses a combination of cone cells, during daylight vision, which are most sensitive to green light.

The raw output of Bayer-filter cameras is referred to as a Bayer pattern image. Since each pixel is filtered to record only one of three colors, the data from each pixel cannot fully specify each of the red, green, and blue values on its own. To obtain a full-color image, various *demosaicing algorithms* can be used to interpolate a set of complete red, green, and blue values for each pixel. These algorithms make use of the surrounding pixels of the corresponding colors to estimate the values for a particular pixel.

Once the light from a scene reaches the camera, it must still pass through the lens before reaching the sensor. For many applications, it suffices to treat the lens as an ideal pinhole that simply projects all rays through a common center of projection. The optic system of a camera has several features:

- Focal Length
- Field of View
- Aperture

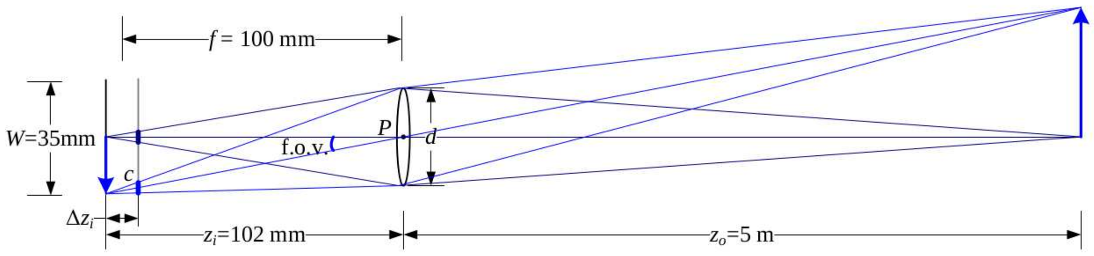


Figure 3.2: A thin lens of focal length f focuses the light from a plane a distance z_o in front of the lens at a distance z_i behind the lens, where $\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f}$. If the focal plane (vertical gray line next to c) is moved forward, the images are no longer in focus and the circle of confusion c (small thick line segments) depends on the distance of the image plane motion Δz_i relative to the lens aperture diameter d. The field of view (f.o.v.) depends on the ratio between the sensor width W and the focal length f (or, more precisely, the focusing distance z_i , which is usually quite close to f).

Focal length is the distance in mm between the convergence point, i.e., the point at which the light rays combine in the lens, and the sensor. The Image

3.2 shows a diagram of the most basic lens model, i.e., the thin lens composed of a single piece of glass with very low, equal curvature on both sides. According to the *lens law* (which can be derived using simple geometric arguments on light ray refraction), the relationship between the distance to an object z_o and the distance behind the lens at which a focused image is formed z_i can be expressed as

$$\frac{1}{z_0} + \frac{1}{z_i} = \frac{1}{f} \tag{3.1}$$

where f is called the *focal length* of the lens.
The field of view can be estimated as

$$\theta \approx tan^{-1}\left(\frac{W}{f}\right) \tag{3.2}$$

The command to capture a frame is given by the shutter of the camera, that is controlled by the sensor itself. Although both technologies record light for the necessary duration, i.e., the exposure time, not every portion of the image starts and stops receiving light at the same time. Digital cameras comes with two main types of shutter: rolling and global shutter. The rolling shutter does not expose all the photosites, i.e. the elements of the sensor, at the same time. This kind of shutter is not adequate for recording moving objects. The global shutter controls incoming light to all photosites simultaneously. At any given point in time, all photosites are either equally closed or equally open. A global shutter is typically considered the most accurate representation of motion, and effectively addresses all of the potential rolling shutter artifacts.

Type of Camera

There are two types of camera in the market: matrix based and line based. As the name suggests, matrix based cameras capture an entire area simultaneously while line based cameras give only a line. A Line Based Camera System is more challenging to design, because needs a precise positioning of the camera and a triggering system to synchronize the speed of the acquisition with the speed of the moving object.
For these reasons we choose to select a matrix based camera.

Resolution

The dimensions of the matrix based frames varies from 2 MegaPixel(MP) to 25MP, and it is reasonable to cover an area from $1mm^2$ to $25mm^2$ per pixel. This implies that a common sensor with a resolution of 1920×1200 may cover an area of $5m \times 5m$ with a resolution of about $2.5mm$ per pixel for the width and $4mm$ for the height. This is near the limits of our requirements, hence we decided to choose a sensor with a better resolution. In fact, a sensor with a frame of 5000×5000 pixels may have a lateral resolution of $1mm$.
We choose a sensor with a frame size of $4096 \times 3072px$ obtaining a width resolution of about $1mm$ and an height resolution of $1.5mm$. Obviously, this choice has a major drawback: the data size. A single RGB frame has a size of $4096 \times 3072 \times 3$ bytes, 36MB.

Frame Rate

From the third requirement, the system must be able to capture, without blurring the image and with some partially overlapped areas, a train that is moving at the speed of 5m/s. Starting from the first requirement, we know that a single frame contains an area of $5m \times 5m$.
Taking into account that the system needs a partially overlapped area between two consecutive frames to stitch the frames in an unique image of the entire train, the minimum allowed frame rate is 2fps. It must be considered that the train does not have a constant velocity and may exceed the speed limit.

Data Transfer Rate

Linking the Frame Rate's result with the data size of the chosen sensor, we need a minimum data transfer rate of about 72 MBps or 576Mbps. This information is taken into account in the analysis of the architectural design.

Exposure Time

The exposure time is the amount of time the shutter remains opened. Bigger the exposure time, bigger the amount of light that strikes the sensor. The fine tuning of this parameter can allow to obtain sharp images of moving objects. In fact, to have a blur of 5mm of the moving object, that travels at the speed

of 5m/s it is needed an exposure time of about 1ms.
To test this behavior we did some experiments at the laboratory, with a free-falling object. We captured different frames with different exposure times: 10ms, 5ms, and 1ms. The pictures shows the different blurs obtained by changing the exposure time of the camera.

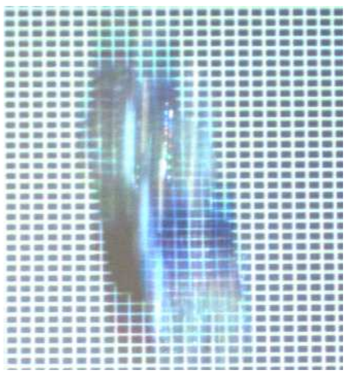


Figure 3.3: 10ms Exposure Time

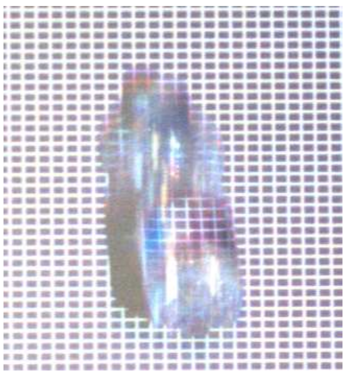


Figure 3.4: 5ms Exposure Time

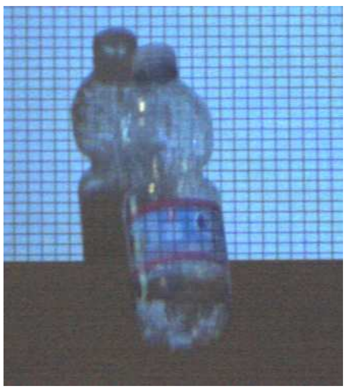


Figure 3.5: 1ms Exposure Time

Shutter Type

From the principles written in the precedent paragraph and the third requirement in the list, the best choice is to use a camera with a global shutter.

Camera Lens

The selection of the best camera lens to use is not so straightforward as the shutter's. Lenses introduce distortions in the image, because of their spherical geometry. The distortions are analyzed in more details in the calibration's section. To reduce this effect, camera lenses are composed by a set of lenses placed in a specific configuration, this explains the huge cost of this devices. At the moment we want to minimize as much as possible the distortions.
To choose a camera lens that meets the first requirement we must impose a condition on the placement of the vision system: it is needed to specify the distance from the train. It has seen that the system, for security reasons, must be placed at a distance of 2.5m away from the train.
Now, it can be computed the required field of view. And by exploiting similar

triangles it can be written that

$$\frac{W}{f} \approx \frac{5}{2.5} = 2 \tag{3.3}$$

where W is the size of the sensor, and f is the focal length of the camera lens. Hence it can be obtained the focal length of the camera lens in function of the sensor's size, as

$$f \approx \frac{W}{2} \tag{3.4}$$

Chosen Camera

Putting together the characteristics that the camera has to have to meet the requirements of the vision system we obtain the following:

- Type of Camera: Matrix Camera
- Resolution: 4096 x 3072
- Frame Rate: at least 2 FPS
- Data Transfer Rate: at least 576 Mbps
- Exposure Time: at most 1ms
- Shutter: Global Shutter
- Camera Lens that introduces very small distortions
- Camera Lens with a focal length half of the sensor's size

Considering these attributes it has been chosen the *Emergent Vision Technologies HT-12000 10 GigE Camera*, that has the following specifications:

- Type of Camera: Matrix Camera
- Resolution: 4096 x 3072
- Frame Rate: max 30 FPS in RGB mode
- Data Transfer Rate: 10 Gbps
- Exposure Times: from 30 μs to 1s

Figure 3.6: The Emergent Vision Technologies HT-12000 10 GigE Camera

- Shutter: Global Shutter
- 28mm CMOS sensor

3.2.2 Projector

Basic principles

The basic principles of the projector are more or less the same of those of the camera. In fact, a projector can be seen as an inverted camera: instead of capturing photons, it emits light rays.

Optics

To meet the requirement of cover an height of about 5m it has been performed an accurate choice of the optics. The characteristics that is interesting is the *throw ratio*. Projectors that can project big areas from a small distance are called *short-throw*, and they have a throw ratio less than 1. The throw ratio is

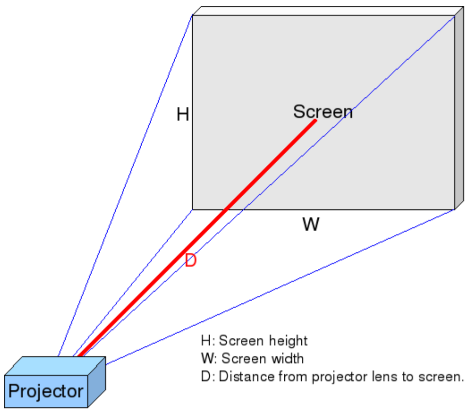


Figure 3.7: Projector Throw Distance

defined as the ratio between the distance of the projector from the plane and the size of the diagonal of the projected area.
From the requirements follows that the system needs a projector with a throw ratio of at least 0.5, or a rack composed by two projectors, one above the other, with throw ratio of about 1.

DLP, Strobo Comparison

In order to obtain the correct color and luminance properties of the surface to be reconstructed, it is needed to alternate the projected pattern. This is what stroboscopic systems do. In the design phase of the project we tried a stroboscopic system and we compared it to a DLP projector. The analysis has been made with an exposure time of 200μs. We found that the optic's power of the stroboscopic system was not enough in order to robustly separate the pattern from the background. The pictures below shows the results of the comparison: From the picture we can see that a normal DLP Projector

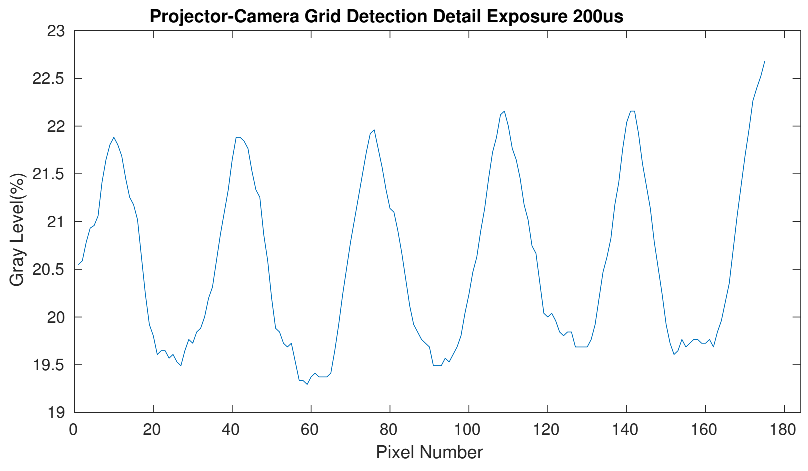


Figure 3.8: The graylevel analysis of the DLP Projector: it has been analyzed a line of a detail of the image orthogonal to the pattern

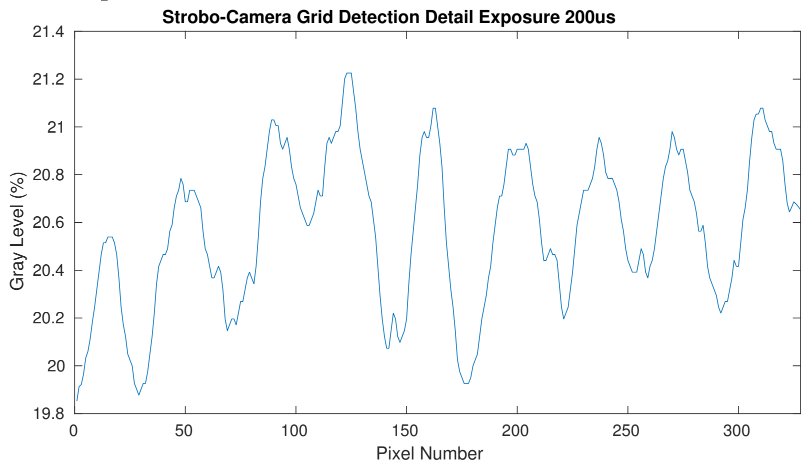


Figure 3.9: The graylevel analysis of the Strobo: it has been analyzed a line of a detail of the image orthogonal to the pattern

outperforms the stroboscopic system: the range of the DLP is about 3% while the mean range of the strobo is about 1%. Starting from these consideration it has been chosen to work with a DLP Projector.

Chapter 4

Computer Vision Background

Among all sensorial activities, vision is widely recognized as the sense with more possibilities. Visual information is the richest one.

Computer Vision is the discipline focused on images analysis with the computers aid. Computer Vision has strong connections with Mathematics and Computer Science and looser connections with physics, psychology of perception and neurosciences.

Computer vision doesn't try to reply the human vision system. There is an intrinsic difference between the two "hardware".

Computer vision could be seen as the inverse problem of Computer Graphics, in which, using:

- scene geometric description,
- scene radiometric description (light sources and optical surface properties),
- complete visual device description (camera),

the computer tries to produce the virtual 2D image seen by the camera.

Computer Graphics systems are not yet perfect, but are able to determine a very good illusion of reality.

In Computer Vision we try to do the inverse: to describe the world that we see in one or more images and to reconstruct and understand its properties, like the shape, the color and the quantity of people and objects present in the scene seen.

Nowadays Computer Vision is yet an open field, especially in terms of algorithm complexity and full scene understanding. So many computer vision algorithms of various levels (from feature extraction through object detection to people skeleton tracking) still remain brittle and can function robustly only under restrictive conditions (during day rather than night, better with diffuse lighting conditions and no shadows). Despite these things, every day a small improvement is done and the list of industrial and consumer applications of Computer Vision has became already long.

4.1 Camera Device

The simplest model of a Camera, widely used in Computer Vision, is certainly the so-called *Pin Hole model* of a camera. Its definition allows to compute the transformations that maps 3D coordinates of real world points into the corresponding 2D image points specified by pixel coordinates.

4.1.1 Pin-hole camera model

Simplified Model Let us begin introducing a particular reference frame $\{C\}$ (Fig. 4.1) with its origin placed in C (Camera Optical Center) and with the z_C axis oriented like the camera optical axis. A reference frame so placed and so oriented is called *standard camera frame*. Initially the world frame $\{0\}$ coincides with the camera frame $\{C\}$. In a more general representation the standard frame and the world frame are not joined, we'll see this in the next paragraph when we will compute the complete pin-hole model equations.

Then it is useful to introduce another reference system (x, y) for the image plane with the origin placed in the image principal point with the axes oriented like x_C and y_C. Consider a point P with world coordinates $\mathbf{P} = [X, Y, Z]^T$ and its image projection point p with image coordinates $\mathbf{p} = [x, y]^T$.

Using similar triangles, it is straightforward to obtain the following relations:

$$\begin{aligned} x &= \frac{X}{Z} f \\ y &= \frac{Y}{Z} f \end{aligned} \tag{4.1}$$

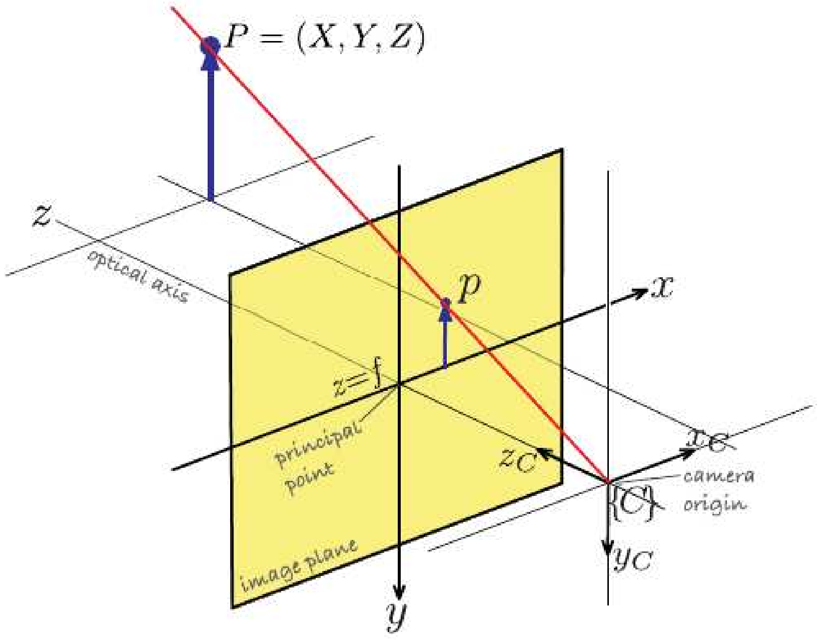


Figure 4.1: Pin-hole camera model. The image plane is f in front of the camera's origin and on which a non-inverted image is formed. The world frame is placed in a particular way, this frame is usually called *standard camera frame*

This is the projection transform. The relation is clearly not linear (caused by the division for Z). Instead, using homogeneous coordinates the relation becomes linear and we can use the usual matrix algebra. So, the simplified model can be rewritten as:

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

In terms of matrices we have:

$$\tilde{p} = C\tilde{P} \tag{4.3}$$

or, better:

$$\tilde{p} \simeq C\tilde{P} \tag{4.4}$$

where $\simeq$ means "equal unless a scaling factor", and C is a 3x4 matrix known as the camera matrix. Note that $\tilde{P}$ is $\tilde{P}^C$, the coordinates of the point with respect to the camera frame. In Computer Vision this matrix is often factored in two matrices:

$$\tilde{p} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{P} \tag{4.5}$$

where the second matrix is the projection matrix, Π.

Complete Model In the previous camera model we have done several simplification. In order to obtain a more realistic and useful camera model, we need to include these following points:

- The *pixellation*, every camera is built up of many small photo sensors, the totality of these compose a CCD matrix. We must take into account its size, shape and position respect to the image plane optical center (Fig. 4.2)
- To remove the hypothesis of world frame placed and oriented as the camera standard frame. We need to add a general homogeneous matrix transformation between camera and world frames (Fig. 4.3).
- To take into account the use of real lenses with distortion effects on the light rays that pass not closer to the lens center (Fig. 4.4).

Intrinsic Parameters In a digital camera the image plane is a WXH grid of light sensitive elements called photosites that correspond directly to the picture elements (or pixels) of the image as shown in Fig.4.2. The pixels coordinates are a 2-vector (u, v) of non-negative integers and by convention the origin is at the top-left corner of the image plane. Therefore, the pixel coordinates are related to the image plane by an affine transformation that takes into account the optical center translation and the independent dimension scaling along u and v axis:

$$\begin{aligned} u &= f\frac{x}{\rho_w} + u_0 \\ v &= f\frac{y}{\rho_h} + v_0 \end{aligned} \tag{4.6}$$

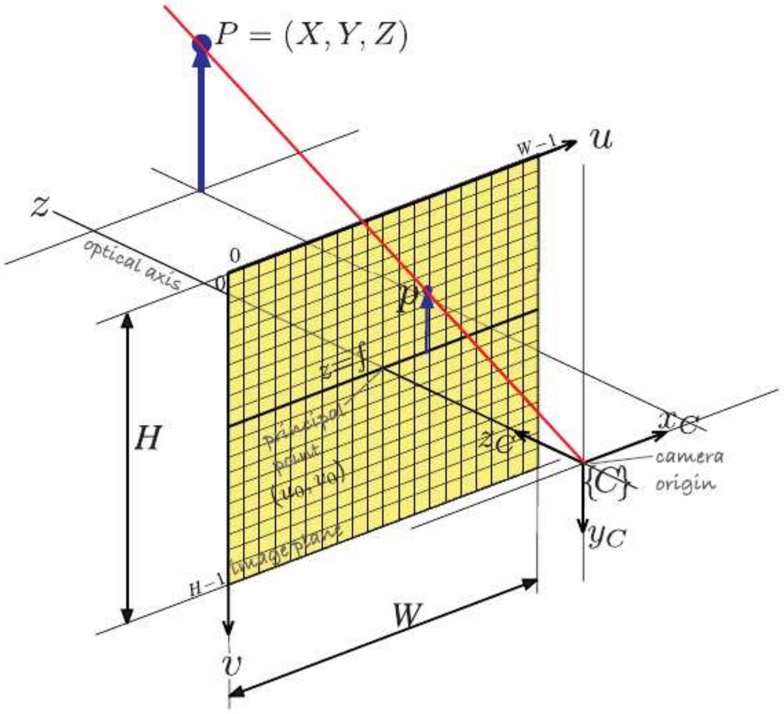


Figure 4.2: Pin-hole camera model, a more realistic model of the image plan contains discrete pixel

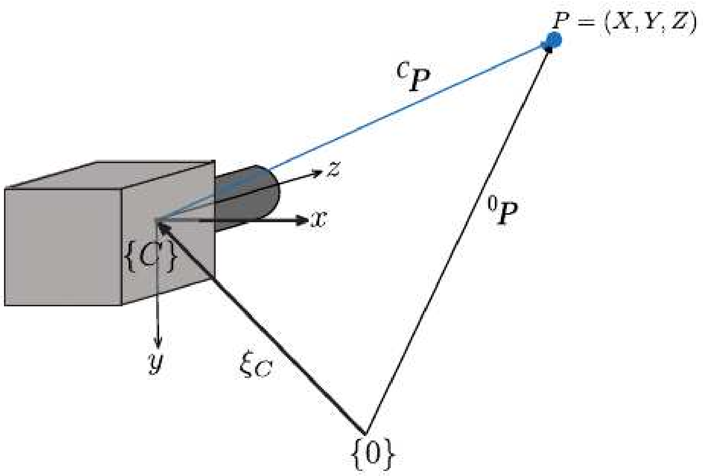


Figure 4.3: In general, world frame doesn't coincide with the camera frame

where ρ_w and ρ_h are the width and height of each pixel respectively, and (u_0, v_0) are the principal point coordinates, the point where the optical axis intersects the image plane.

Using this affine transformation

$$\tilde{p} = \begin{bmatrix} f/\rho_w & 0 & u_0 & 0 \\ 0 & f/\rho_h & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{P} = K[I|0]\tilde{P} = K\Pi\tilde{P} \tag{4.7}$$

where K is the camera parameter matrix or, again, the camera matrix:

$$K = \begin{bmatrix} f/\rho_w & 0 & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

We can choose $\alpha_u = f/\rho_w$ and $\alpha_v = f/\rho_h$, they represented the focal length, respectively expressed in horizontal and vertical pixels. Therefore, $\tilde{p}$ is now expressed with (u, v) coordinates. The intrinsic parameters contained in the new Camera matrix K are the following four: $\alpha_u, \alpha_v, u_0, v_0$.

A more general model considers an additional parameter, θ, the angle between u and v image axis (normally $\theta = \pi/2$). In this case we have another parameter inside the camera matrix K, the shearing factor or skew parameter $s = cot(\theta)$:

$$K = \begin{bmatrix} f/\rho_w & s & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

However this model is not so much used, usually, camera calibration algorithms give an estimation of $\alpha_u, \alpha_v, u_0, v_0$ parameters supposing $\theta = \pi/2$.

Extrinsic Parameters Finally, in order to take into account that world and camera frames don't coincide, we have to introduce the rigid transformation that connects these two frames (Fig. 4.3).

Usually the known homogeneous transformation matrix is : ${}^{w}T_c$ [1] Adding

[1] We are here using one of the more common notation in robotics. This is used in *Robotics, Modellistic, Planning and Control*, Siciliano, Sciavicco. ${}^{w}T_c$ has for columns the c-frame versors expressed in world frame and allows to express a general vector p from camera frame to world frame, finally describes the rigid transformation that brings world frame on camera frame

the frame superscripts, the relation 4.7 becomes:

$$\tilde{p} = K\Pi\tilde{P}^c \tag{4.10}$$

and it follows:

$$\tilde{p} = K\Pi({}^wT_c)^{-1}\tilde{P}^w = K\Pi{}^cT_w\tilde{P}^w \tag{4.11}$$

where ${}^cT_w$ contains the extrinsic camera parameters, in fact the 3x3 rotational matrix and the 3x1 translation vector. These are the extrinsic parameters. Equation 4.11 fully describes the mapping from a 3D real world point to a 2D pixel coordinate point for the ideal pinhole camera model.

Distortion parameters The perspective projection only models the linear part of the image formation process. However, real cameras use lenses and suffer from non-linear distortion. Depending on the design and the quality of the lenses, the resulting image will be more or less distorted (Fig. 4.4).

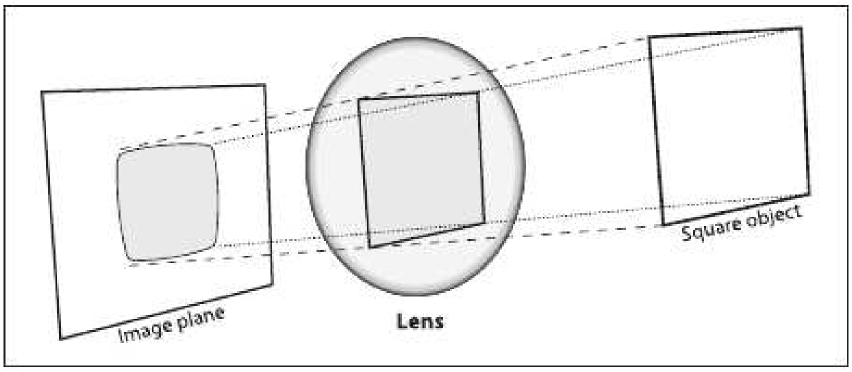


Figure 4.4: Radial Distortion in lenses: rays that pass not closer to the center are bent too much, thus square sides appear rounded

For example, narrow field of view lenses tend to have smaller distortions than wide field of view lenses. In order to use images for 3D reconstruction, it is usually necessary to compensate those distortions. Therefore, the projected image plane coordinates of a real world point have to be transformed before multiplying them with the intrinsic camera matrix. Usually, a distortion model including radial and tangential distortions is used, and the transformation from distorted image plane coordinates (u, v) to undistorted image plane coordinates

(u_d, v_d) is given by:

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u(1 + d_1 r^2 + d_2 r^4 + d_5 r^6) \\ v(1 + d_1 r^2 + d_2 r^4 + d_5 r^6) \end{pmatrix} + \begin{pmatrix} 2d_3 uv + d_4(r^2 + 2u^2) \\ d_3(r^2 + 2v^2) + d_4 uv \end{pmatrix} \tag{4.12}$$

with $r^2 = u^2 + v^2$. The first contribute is the radial distortion, the second one the tangential distortion.

Usually, the distortion parameters are collected in a vector D, that is how a common calibration tool (like the calibration tools based on OpenCV library) provides the distortion parameters. Sometimes d_5 is omitted, depending on how much accuracy is needed. In order to use the model given in 4.12 to undistort an image, non linear equations have to be solved using optimization methods. This cannot be done real time; however, it is possible to precompute a lookup table with a calibration procedure and then to undistort images real time.

Summarizing, using equations 4.11 and 4.12 it is possible to go from pixel coordinates to 3D real world coordinates while compensating for lens distortion.

4.2 Stereo Vision

Stereo vision is motivated from the human vision system, which can perceive depth properties of a scene. The human vision system obtains two different images of a scene from the slightly different views of the eyes and interprets the images for depth perception. Because depth perception is one of the powerful abilities of the human vision system, stereo vision has many application areas, such as three-dimensional (3D) scene reconstruction, object recognition, entertainment, robot navigation, etc. To obtain the depth information of a scene from a pair of images, it needs to solve one of the inherent problems in stereo vision, called the *correspondence problem*.

Because two different image planes are defined in stereo vision, the projections of an object into the planes are represented with respect to different image coordinates. Therefore, the correspondence problem can be defined as determining the coordinate difference between the images of the object. Solving the stereo correspondence problem is called *stereo matching*. The result of stereo matching is commonly represented by a disparity map whose intensity represents the coordinate difference between corresponding image points, called *disparity*.

Finding stereo correspondences needs a huge number of computations. For example, if the resolution of each image is $N \times M$, a brute force approach for fnding all correspondences needs $(N \times M)^2$ computations. To reduce the computation complexity of the stereo matching problem, the 3D geometry of a stereo vision system should be considered so that stereo matching is restricted to certain image areas. To know the stereo geometry, calibration of a stereo vision with respect to a reference coordinate system is needed. A direct calibration method is presented to obtain a geometric relationship between the two cameras, which is represented by 3D Euclidean transformations, rotation, and translation. From the stereo calibration, we know that stereo correspondences are related by a 2D point-to-line projection, called *epipolar geometry*.

4.2.1 Epipolar Geometry

The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the cameras' internal

parameters and relative pose. The fundamental matrix F encapsulates this intrinsic geometry.
It is a 3×3 matrix of rank 2. If a point in 3-space X is imaged as x in the first view, and x' in the second, then the image points satisfy the relation $x'Fx = 0$.
The epipolar geometry between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as axis (the baseline is the line joining the camera centers). Suppose a point X in 3-space is imaged in two views, at x in the first, and x' in the second. What is the relation between the corresponding image points x and x' ? As shown in figure 4.5 the image points x and x' , space point X , and camera centers are coplanar. Denote this plane as π. Clearly, the rays back-projected from x and x' intersect at X , and the rays are coplanar, lying in π. It is this latter property that is of most significance in searching for a correspondence. Supposing now that we know only x, we may ask how the corresponding point

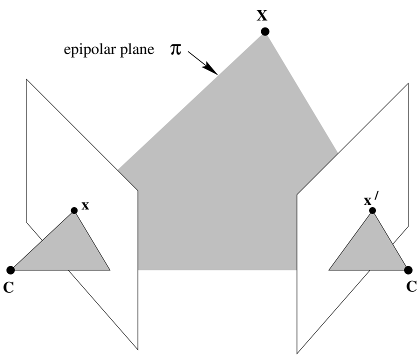


Figure 4.5: The epipolar plane

x' is constrained. The plane π is determined by the baseline and the ray defined by x. From above we know that the ray corresponding to the (unknown) point x' lies in π, hence the point x' lies on the line of intersection l' of π with the second image plane. This line l' is the image in the second view of the ray back-projected from x. It is the epipolar line corresponding to x. In terms of a stereo correspondence algorithm the benefit is that the search for the point corresponding to x need not cover the entire image plane but can be restricted to the line l'.
The terminology is

- The epipole is the point of intersection of the line joining the camera centers (the baseline) with the image plane.

- An epipolar plane is a plane containing the baseline. There is a one-parameter family of epipolar planes.
- An epipolar line is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

4.3 Structured Light

A coded structured light system is based on the projection of a single pattern or a set of patterns onto the measuring scene which is imaged by a single camera or a set of cameras. The patterns are specially designed so that codewords are assigned to a set of pixels. Every coded pixel has its own codeword, so there is a direct mapping from the codewords to the corresponding coordinates of the pixel in the pattern. The codewords are simply numbers, which are mapped in the pattern by using grey levels, color or geometrical representations. The larger the number of points that must be coded, the larger the codewords are and, therefore, the mapping of such codewords to a pattern is more difficult. The aim of this work is to survey the available strategies used to represent such codewords. Pattern projection techniques differ in the way in which every point in the pattern is identified, i.e. what kind of codeword is used, and whether it encodes a single axis or two spatial axis. In fact, it is only necessary to encode a single axis, since a 3D point can be obtained by intersecting two lines (i.e. when both pattern axis are coded) or intersecting one line (the one which contains a pixel of the camera image) with a plane (i.e. when a single pattern axis is coded). In the following sections is described a basic taxonomy of structured lightning strategies. A more in depth report of these systems is given by Salvi et al. in [1].

4.3.1 Time-multiplexing strategy

One of the most commonly exploited strategies is based on temporal coding. In this case, a set of patterns are successively projected onto the measuring surface. The codeword for a given pixel is usually formed by the sequence of illuminance values for that pixel across the projected patterns. Thus, the codification is called temporal because the bits of the codewords are multiplexed in time. This kind of pattern can achieve high accuracy in the measurements. This is due to two factors: first, since multiple patterns are projected, the codeword basis tends to be small (usually binary) and therefore a small set of primitives is used, being easily distinguishable among each other; second, a coarse-to-fine paradigm is followed, since the position of a pixel is being encoded more precisely while the patterns are successively projected.

During the last twenty years several techniques based on time-multiplexing have appeared. Salvi et al. have classified these techniques as follows: a) techniques based on binary codes: a sequence of binary patterns is used in order to generate binary codewords; b) techniques based on n-ary codes: a basis of primitives is used to generate the codewords; c) Gray code combined with phase shifting: the same pattern is projected several times, shifting it in a certain direction in order to increase resolution; d) hybrid techniques: combination of time-multiplexing and neighborhood strategies.
Unfortunately these techniques can only be applied to static objects.

4.3.2 Spatial Neighborhood strategy

The techniques in this group tend to concentrate all the coding scheme in a unique pattern. The codeword that labels a certain point of the pattern is obtained from a neighborhood of the points around it. However, the decoding stage becomes more difficult since the spatial neighborhood cannot always be recovered and 3D errors can arise. Normally, the visual features gathered in a neighborhood are the intensity or color of the pixels or groups of adjacent pixels included on it. These spatial neighborhood techniques can be classified as follows: a) strategies based on non-formal codification: the neighborhoods are generated intuitively; b) strategies based on De Bruijn sequences: the neighborhoods are defined using pseudorandom sequences; c) strategies based on M-arrays: extension of the pseudorandom theory to the 2-D case.

Chapter 5

Calibration

Camera calibration is the procedure to obtain the optical and geometrical device properties (intrinsic parameters) and the position and orientation with respect to an absolute frame (extrinsic parameters). This parameters determine how 3D points are projected into the image plane, so, the fundamental idea of every calibration algorithm is that, using coordinates of known 3D points, it is possible to obtain these parameters values solving the projection equations.

There are different calibration procedures, these can be divided in linear, and non linear methods.

If the number of point used is, from a mathematical point of view, the minimum to solve the system equation, we are talking about a linear or direct method: it is faster but less accurate. Instead, we are using a non linear method when the number of points with known 3D coordinates is larger than the number necessary to solve the equations. In this case, with iterative methods an optimization problem has to be solved: it is slower but more accurate. Non linear method are usually chosen using , as initial values, those provided by a linear method.

First calibration methods used not so practical calibration object like the one represented in Fig. 5.1. The calibrations points are the chessboard vertices, their coordinates are known by construction (the world frame in which extrinsic parameters have to be estimated is placed on the object).

Image points coordinates can be extracted using classical image processing methods. The precision by these points are localized is the key factor to have

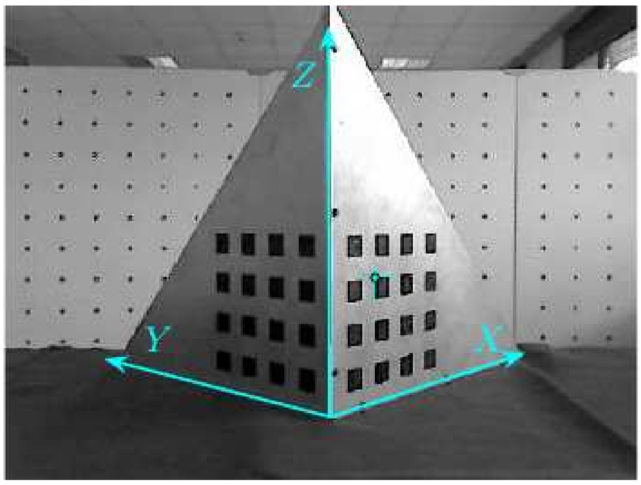


Figure 5.1: First Calibration object with the world reference frame superimposed. In this frame all squares vertex coordinates are measured, one of these is highlighted.

good calibration outcomes.

The previous techniques need an object with two or three perfectly orthogonal planes, an object very difficult to build without the aid of a machine shop. Instead, it is enough easy to get a good planar object.

The majority of commonly used calibration methods is inspired by the approach developed by Zhang in [2], based on many images of a plane instead of one image of many planes.

These are the main steps of Zhang's procedure:

1. Print a pattern and attach it to a planar surface;
2. Take a few images of the model plane under different orientations by moving either the plane or the camera;
3. Detect the feature points in the images;
4. Estimate the five intrinsic parameters and all the extrinsic parameters using the closed-form solution described in [2];
5. Estimate the coefficients of radial distortion by solving by linear least-squares a matrix equation that contains the ideal (u_i, v_i) coordinates and the real coordinates (u_r, v_r) of detected corners (In later works a complete distortion model - radial and tangential - is used);

6. Refine all parameters with a minimization of a functional that bring to a Maximum Likelihood estimation.

The famous Camera Calibration Toolbox for Matlab written by Jean-Yves Bouguet and its C implementation for OpenCV (we have both used in this work) have been deeply inspired by Zhang's work.

From a practical point of view you have to put a chessboard on a good planar rigid object and to take photos with many chessboard poses, trying to cover the entire camera field of view.

How many chessboard images do we need for how many camera parameters? Usually, as in the OpenCV case, we have to solve four intrinsic parameters and five distortion parameters. Intrinsic parameters are tied to 3D geometry (and thus also with extrinsic parameters) of where the chessboard is placed in space; distortion parameters are instead tied to the 2D geometry of how lines get distorted. Three corner points in a known pattern brings six pieces of information, hence, in principle, they are enough (obviously in practice, for robustness, more of three points are usually employed). Thus one view of the chessboard is all we need to compute all the distortion parameters.

We must develop the reasoning for computing intrinsic parameters taking into consideration extrinsic parameters.These are composed by three rotation parameters (χ, ϕ, θ) and three translation parameters (t_x, t_y, t_z) for a total of six for each view of the chessboard, because in each image the chessboard change 3D position. In total (intrinsic plus extrinsic) we have ten parameters to solve for each view.

We can call M the number of images and N the number of corners of chessboard. How many views and corners must we see?

- M images provides $2NxM$ constraints (every image point has u and v coordinates)
- With M images taken, we have to solve always the same 4 intrinsic parameters and $6M$ extrinsic parameters
- Hence we have to solve: $2NM \geq 6M + 4 \rightarrow M(N-3) \geq 2$

Thus it seems that if $N = 5$ then we need only one image. Neglecting robustness need, this could be with a generic disposition of points, but our chessboard points are on a plane. An homography can bring at most eight parameters from four (u,v) pairs. This is because four points are needed to express everything that a planar perspective view can do.

The truth is that for each chessboard (no matter how many corners are detected!)only four corners are really necessary. So if follows that, with Zhang-inspired method, $N = 4$. Therefore, it follows: $(4 - 3)M \geq 1$, which means $M > 1$. This implies that two views of a 3x3 chessboard (taking into consideration only the unique internal corners for image processing requirements) are the minimum to solve the calibration problem.

In practice, robustness considerations lead to collect more images (to estimate better intrinsic parameters) of a larger chessboard (to estimate better the homography): for instance, for high-quality a good choice is to collect about twenty images of 6x9 chessboard (trying to obtain a "rich" set of view covering with chessboard movement every corner of the camera field of view).

5.1 Camera-Projector Calibration

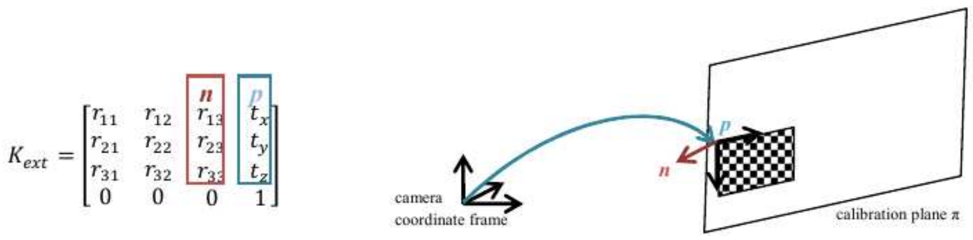
The projector calibration procedure is, from a theoretic point of view, similar to camera calibration. This is due to the fact that a projector can be seen as a dual of a camera: an inverse camera which maps 2D image intensities into 3D rays, thus making the calibration of a projector is mathematically the same of a camera. Having the 2D projected point coordinates, we have to find the 3D coordinates of the intersection between rays and a geometrically known plane surface. Obtained these 3D correspondences, the system can be calibrated using a standard camera calibration method.

This logic has been developed in [3]; the authors have created `ProCamCalib`, a Matlab toolbox for the calibration of a projector-camera system. Coherently what said before, it is based on the well-known "Bouguet Toolbox" for cameras.

Now we can go a bit more deeper in the projector-camera system calibration problem. We have showed the logic that makes, theoretically, the projector calibration the same of a camera. In practice, there are two main differences that make projector calibration more complicated. The first is obvious: projectors cannot see the surface which they illuminate, therefore the correspondence between 2D projected points and 3D cannot be done without a camera. The second one is that it is difficult to determine the 3D coordinates because the calibration pattern is not fixed as physical chessboard and so not strictly attached with a world reference frame. The method allows to obtain the intrinsic and extrinsic parameters for both the camera and the projector.

The method is made up of several steps:

1. Calibrate the Camera using Zhang's method;
2. Compute calibration plane equation in camera coordinate system (Fig. 5.2);
3. Project a chessboard on the calibration plane and detect the projected corners (Fig. 5.3);
4. Apply ray-plane intersection to obtain 3D coordinates of every projected corner (Fig. 5.4 , Fig. 5.5);
5. Calibrate the projector using Zhang's method with the correspondences between the 2D points of the chessboard projected and the 3D illuminated points.



$K_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
n
p
camera coordinate frame
p
n
calibration plane π

Figure 5.2: Use of extrinsic parameters to compute plane equation: p represents the position of a known point in the plane, n a normal vector to the plane.

Details of these main steps are explained in [3].

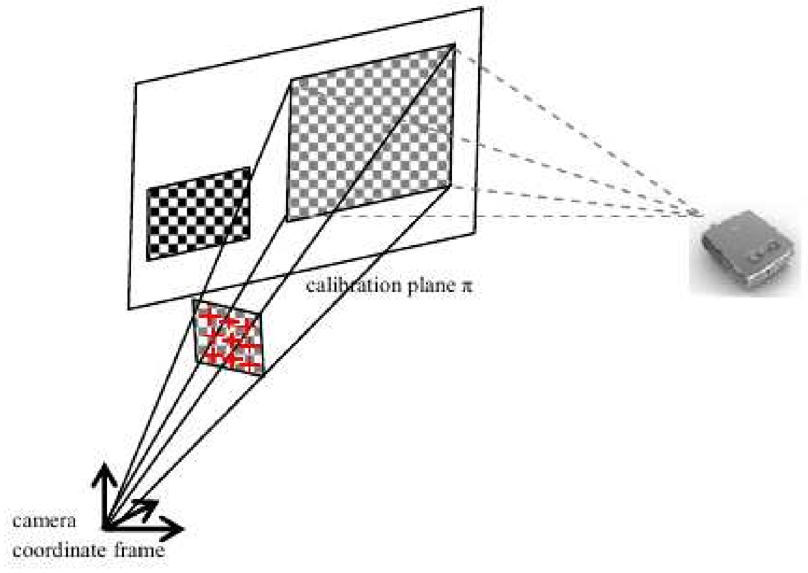


Figure 5.3: Corner Extraction of the projected chessboard.

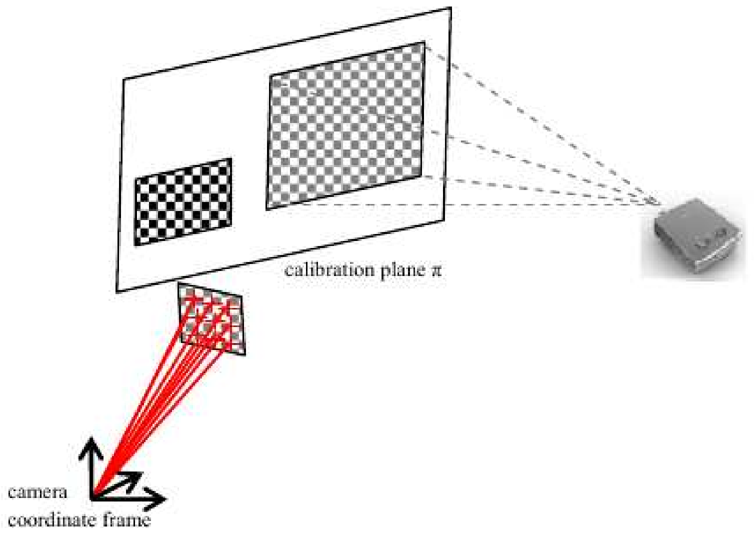


Figure 5.4: Building of the expression of the 3D rays coming from the camera and going through the corners of the projected pattern.

5.2 Implementation Details

The Calibration Procedure has been implemented in three different modules:

- Camera Calibration Module

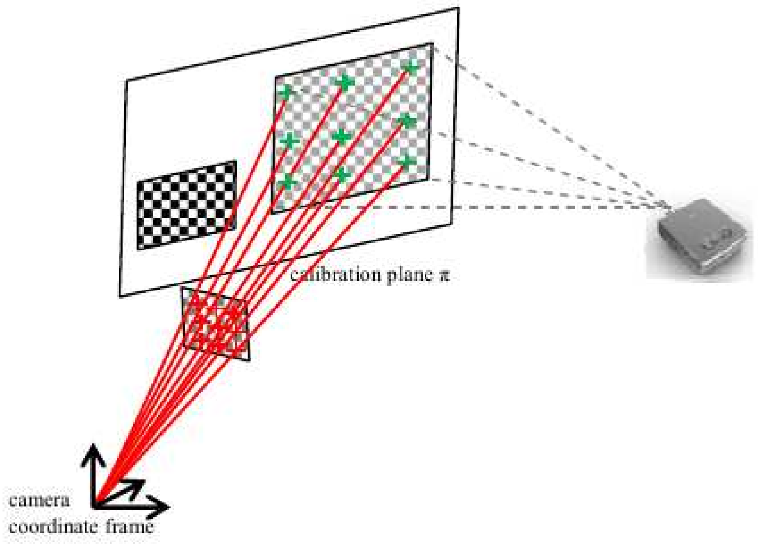


Figure 5.5: Ray-plane intersection gives 3D projected points, now we can apply camera calibration method and recover intrinsic and extrinsic projector parameters with a final Stereo Calibrate.

- Projector Calibration Module

Both the modules have been implemented in Python3 using the version 3.1 of the OpenCV Library. The modules take advantage of the NumPy library to perform linear algebra calculations.

5.2.1 Camera Calibration Module

The camera calibration module takes as input a set of images containing a chessboard in different orientations. The code developed for this module follows the one proposed by the OpenCV Library Examples, it has been customized in order to check each image and decide whether to accept it into the calibration set or not.
The flexibility of the module is guaranteed by a set of command line arguments that are parsed with the argparse Python module, and it is possible to:

- Specify the directory where the images are stored;
- Specify the size of each square of the chessboard;
- Specify the dimensions of the pattern, in terms of inner corners;

- Specify the output directory, where to store the results of the calibration;

Moreover, it is possible to select different calibration modes:

- Calibration Mode that uses an Intrinsic's Guess, i.e., the user can give a starting point for the calibration procedure (a guess). This improves the quality of the calibration if the user has an approximate knowledge of how the intrinsics parameters should be: e.g., by knowing the focal length of the camera lens.

- Calibration Mode with Fixed Principal Point. The principal point of the majority of cameras in the market has a irrelevant offset with respect to the center of the frame. This value can be fixed by knowing the camera's resolution.

The Camera Calibration Module produces as output a JSON File, that contains the results of the calibration: the camera matrix and the distortion parameters. An example of calibration results is given below:

```
{
   "rms": 0.3267281886412525,
   "dist": [[-0.1156807921717977795], [0.0269022363135902 4], [-0.
      0012492869352219 07], [-0.0022755807084936 53], [0.43771652
      64689416]],
   "K": [[3886.843356251176 4, 0.0, 2048.0],
        [0.0, 3888.938773266193 5, 1536.0],
        [0.0, 0.0, 1.0]]
}
```

5.2.2 Virtual Chessboard Tool

The Virtual Chessboard Tool has been developed using OpenGL/GLUT and Python3. This tool allows to control a chessboard pattern in the projector's image plane.
The purpose of this tool is to easily project a virtual chessboard on a plane. In fact, if, as in our case, the projector is fixed, there is the possibility that only a part of the projected chessboard is actually projected onto the calibration plane. This behavior makes the calibration process hard.

The tool generates a chessboard on the projector plane and allows users to, interactively:

- Add and remove rows from the chessboard by pressing I or K
- Add and remove columns from the chessboard by pressing J or L
- Scale the chessboard by pressing Z or X
- Move the chessboard on a specific position using the keyboard arrows

It is capable to work on different resolutions, and it is easy to reconfigure: it takes the configuration parameters parsing the command line arguments. The tool produces a JSON file containing the coordinates of each corner of the chessboard pattern.

5.2.3 Projector Calibration Module

The projector calibration module is an implementation of the Falcao algorithm described in the previous section. It takes as input a set of images containing a pair of chessboard in different orientations: a physical chessboard and a virtual (projected) chessboard. It takes also as input the corner's file generated by the Virtual Chessboard Tool.
The flexibility of the module is guaranteed by a set of command line arguments that are parsed with the argparse Python module, they enable users to:

- Specify the directory where the images are stored;
- Specify the directory where the corner's files are stored;
- Specify the path to the camera calibration output file;
- Specify the projector's resolution;
- Specify the size of each square of the physical chessboard;
- Specify the dimensions of the physical chessboard, in terms of inner corners;
- Specify the output directory, where to store the results of the calibration;

Moreover, it is possible to select different calibration modes:

- Calibration Mode that uses an Intrinsic's Guess, i.e., the user can give a starting point for the calibration procedure (a guess). This improves the quality of the calibration if the user has an approximate knowledge of how the intrinsics parameters should be: e.g., by knowing the focal length of the lens.
- Calibration Mode with Fixed Principal Point. The principal point of the majority of DLPs lies on the middle-bottom point of the image. This value can be fixed by knowing the projector's resolution.

The module produces a JSON Output file which contains the results of the projector's calibration, the results of the camera's calibration, the estimation of the Euclidean Transformation between the projector and the camera.

```
{
   "Kc": [[3886.8433562511764, 0.0, 2048.0], [0.0, 3888.9387732
      661935, 1536.0], [0.0, 0.0, 1.0]],
   "Kp": [[2632.258736035449, 0.0, 700.0], [0.0, 2644.748684412
      726, 1050.0], [0.0, 0.0, 1.0]],
   "extrinsics": [[0.955835340858023, 0.005709240523312683, 0.2
      9273220467330685, -59.94648174593118],
                  [-0.000792138537313434, 0.9998013378986275, -0
                     .016102866800070084, -4.046968369311035],
                  [-0.29273330670060294, 0.015265037250979191, 0
                     .9556789574325704, -8.712028401888363],
                  [-3.4686830229061424e-17, 6.160423589590242e-1
                     7, -1.442472470438644e-17, 1.00000000000000
                     1]],
   "distc": [[-0.11568079217197795], [0.02690223631359024], [-0
      .001249286935221907], [-0.002275580708493653], [0.4377165
      264689416]]
}
```

Chapter 6

3D Reconstruction System

The 3D Reconstruction System aims at recovering the shape of real objects. This module is the core part of the system, because it generates the data that will be used to analyze the depth profile of the train's external surface, retrieve the position of possible defects and send an alarm to the operator.
In this section it is analyzed the design and the development of a 3D Structured Light Scanner that uses a special type of pattern: a grid whose lines are not equally spaced, but follows the sequence of DeBruijn.
The next sections describe the difference between the uniform and the non-uniform approach and give a more detail description of the DeBruijn's sequence. For this type of reconstruction, considered that this system has to deal with moving objects, the chosen pattern it belongs to the *Spatial Multiplexing* type of structured light systems.
Moreover, it has been analyzed the image pre-processing activities in order to give to the reconstruction algorithm a robust and dense set of points, aiming at making it less dependent from the ambient lights, and the consequent difference of luminance among parts of the image. The reconstruction algorithm used for this thesis is the one described by Ulusoy, Calakli and Taubin in [4].
At the moment, the scanner is composed by a digital camera and a DLP Projector. The reconstruction's process can be summarized to the search of the correspondences between pixels lying on the image plane of the two devices.
Once, these correspondences has been found a process of triangulation on the points of the two devices gives the world coordinates of the detected points in the scene, i.e., the keypoints of the pattern.

6.1 Pipeline

The image below describes the pipeline of the reconstruction process.

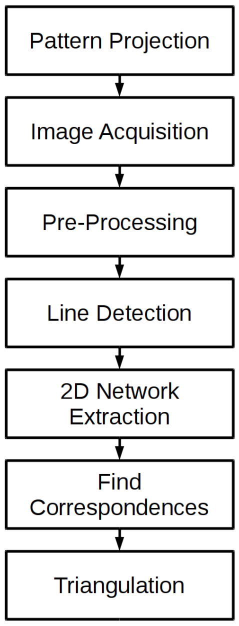


Figure 6.1: The 3D Reconstruction System's Pipeline

As we can see from the image 6.1, the pattern is projected onto the object. Then, a digital camera acquires the image of the object covered by the structuring element. Successively, the image is processed in order to enhance the contrast between the structuring element and the background. At this point the image is ready to be *segmentate*, i.e., is processed in order to decode and isolate the pattern from the rest of the scene.
Then, since it has been used a grid pattern, the system reconstructs a 2D Network to map the spatial relations among the keypoints in the structuring element. This knowledge associated with the geometrical constraints imposed by the vision system, by means of the epipolar geometry, allow a more efficient and robust search of the correspondences between the camera's plane and the projector's plane. Once it has the correspondences it performs a triangulation to retrieve the three-dimensional coordinate of the keypoints. The final output of the system is a *point cloud*, as dense as the pattern is.

6.2 Pattern Choice

In a Structured-Light System the choice of the pattern is relevant in order to achieve robustness and accuracy in the 3D Reconstruction process. The choice of the pattern depends on the characteristics of the application taken into account. In this case, we are dealing with a train that is moving at the speed of 5 m/s. In this scenario, a *Time Multiplexing Approach* will fail, because the pattern's sequence will not be projected onto the same part of the object.
The approach that fits best, for this particular application, is the so-called *one-shot approach*, or Spatial Multiplexing. In this section, it is analyzed the choice among two different patterns: an equally spaced grid, and a grid spaced following a special sequence, called De Bruijn's Sequence.

6.2.1 Uniform Spaced Grid

A grid pattern is a structuring element composed by vertical and horizontal lines. Uniform Spaced Grids have the property that the spacings between the lines of the pattern is constant.

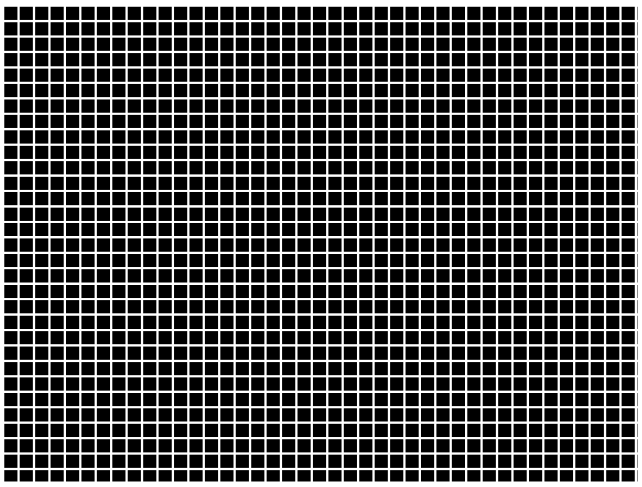


Figure 6.2: An Uniform Grid Pattern

Looking at the state-of-the-art approaches related to this type of pattern we have to cite the work of Ru and Stockman [10] that matched the observed grid in the camera to the projected pattern by exploiting a number of geometric

and topological constraints and global 2D optimization. A major drawback of the approach is that the detected grid points are *numbered with respect to a reference point*, i.e. relative numbering. This necessitates that at least a patch of the pattern be extracted perfectly since in the case of undetected or spurious grid points, the algorithm will fail.
Recently, Kawasaki et al. [5, 4] proposed a colored grid pattern where vertical and horizontal lines are of different colors. Unlike most other grid based approaches, their method does not rely heavily on image processing and is robust against undetected grid points as it does not assume relative ordering. Their solution includes performing singular value decomposition (SVD) on a large and very sparse matrix, which is an expensive operation and may be numerically inaccurate. Moreover, the algorithm may fail to converge to the correct reconstruction in some cases due to instabilities discussed in [5].

6.2.2 De Bruijn Spaced Grid

Ulusoy, Calakli and Taubin in [4] suggest grid spacings that follow a De Bruijn sequence to assure correct convergence. A k-ary De Bruijn sequence of order n is a cyclic sequence containing letters from an alphabet of size k, where each subsequence of length n appears exactly once. There are multiple De Bruijn sequences for a given (k, n), each of length k^n.

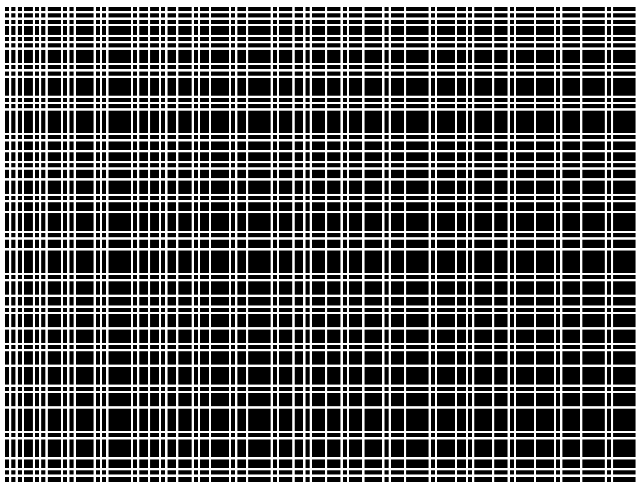


Figure 6.3: A Binary(black and white) De Bruijn Spaced Grid Pattern

The pattern that has been generated has both the vertical and the horizontal spacings between the stripes encoded following a De Bruijn sequence. Thus, a 2D patch consisting of n vertical spacings and n horizontal spacings and containing $(n + 1)^2$ grid crossings is unique in the whole grid pattern. In the next sections, this property is used to guarantee the uniqueness of the correspondence solution between the camera's plane and the projector's plane. Moreover, it has been decided to separate horizontal and vertical stripes giving them two different colors: green and red. The choice of the color comes from the fact that they are two primary colors and they can be easily isolated by taking the green and the red channel from the RGB image.

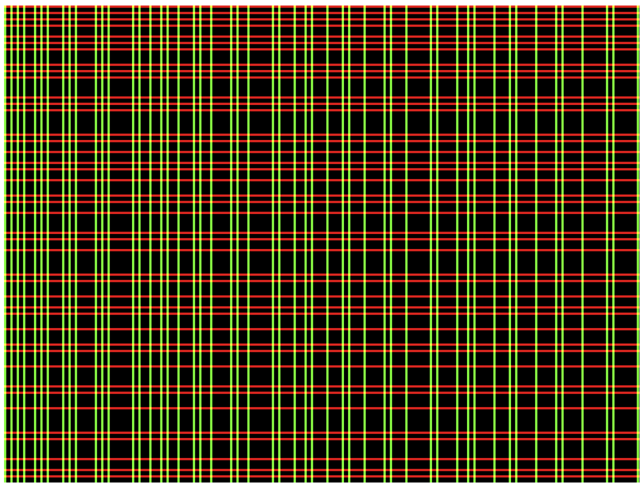


Figure 6.4: The pattern used for the 3D Reconstruction System: A colored De Bruijn Spaced Grid Pattern

6.3 Line Detection

The Line Detection part of the pipeline is what makes this system robust to noise and ambient lights. Since the grid is divided in green and red lines, the detector separates the green and the red channels of the RGB Image, to have only the interested part of the pattern in each frame. The image is processed in order to enhance the contrast between the grid pattern and the background. This section describes the methods used for isolating the grid from the background, for enhancing the contrast, and shows a comparison between two different ways of thresholding the image: with a single threshold and with an *adaptive* threshold. Moreover, this subsystem detects the line with a method called the Center Of Gravity (COG). This method allows a robust detection of the position of the line in the image. The pipeline of line detection is the following:

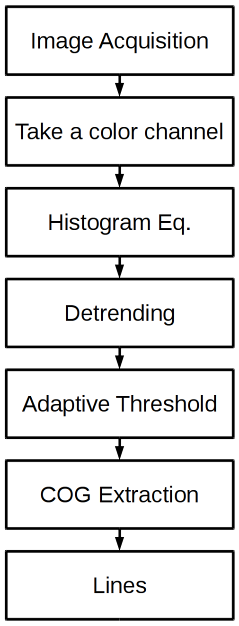


Figure 6.5: The Line Detection's Pipeline

6.3.1 Histogram Equalization

The histogram equalization is a method in image processing of contrast adjustment using the image's histogram. The image histogram is a graphical representation of the intensity distribution of an image. In an 8-bit grayscale image, it is composed by 255 bins and each bin is related to a graylevel value. The number of elements in the bin depend by the number of pixels having that intensity value.

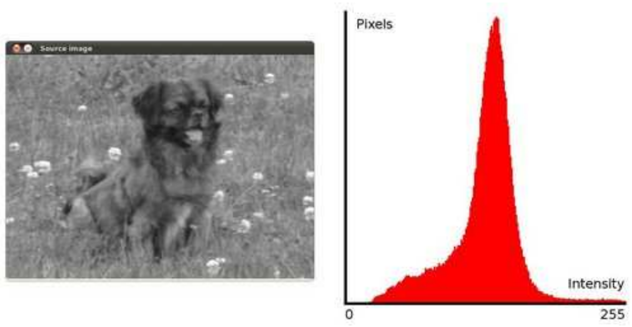


Figure 6.6: A grayscale image with its histogram to the right

As we can see in Figure 6.6, the pixels seem clustered around the middle of the available range of intensities. What Histogram Equalization does, is to stretch out this range.
Anyway, a simple histogram equalization cannot be sufficient or, even worse, can saturate some relevant pixels of the grid, as it is shown in figure 6.7 and 6.8. This happens because its histogram is not confined to a particular region. So to solve this problem, adaptive histogram equalization is used. In this, image is divided into small 8x8 blocks called "tiles". Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, contrast limiting is applied. If any histogram bin is above the specified contrast limit, those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied. This technique is known as *Contrast Limited Adaptive Histogram Equalization*, and it has been introduced by Zuiderveld in [5]. The result of this process is given in figures

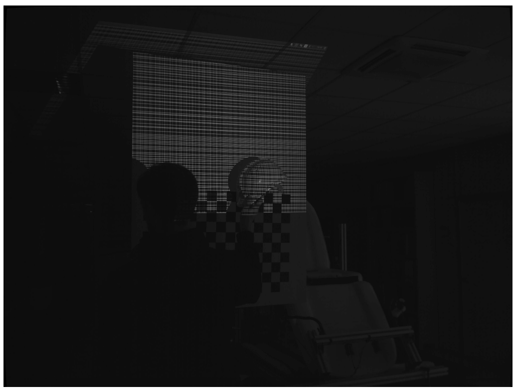


Figure 6.7: Original Red Channel of the image

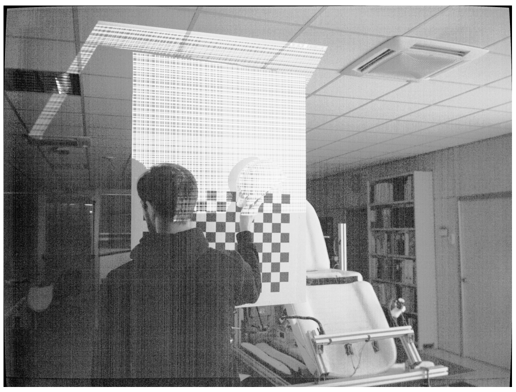


Figure 6.8: The Red Channel after the histogram equalization

6.9 and 6.10.

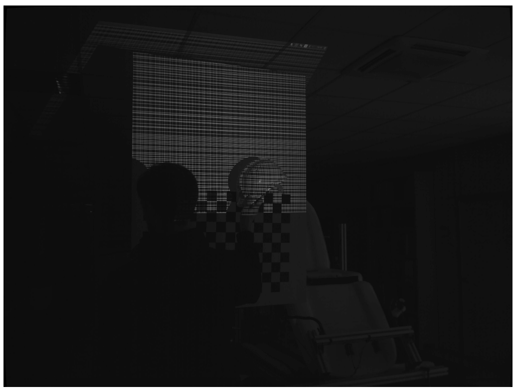


Figure 6.9: Original Red Channel of the image

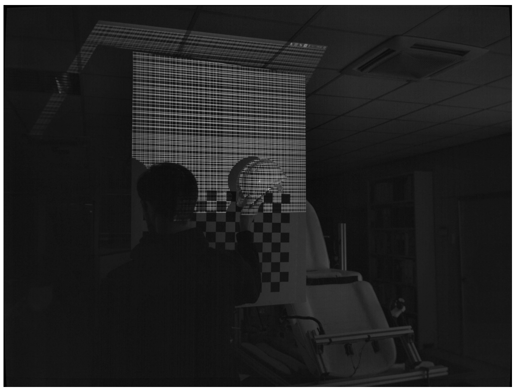


Figure 6.10: The Red Channel after the contrast limited adaptive histogram eq.

It is possible to notice, though the small size of the image in the paper, that there is a substantial improvement in the contrast of the image and the result is way better than the simple histogram equalization.

6.3.2 Detrending

Since the pattern projected onto the object may be seen as a small variation in the intensity value of the image captured by the camera, the operation of detrending can help in isolating the part of the image from the rest of the scene. Detrending is the process of subtracting the mean or a best-fit line

(in the least-squares sense) from a data distribution. In this case, the data distribution corresponds to the intensity value of pixels of a specific row of the image. Removing a trend from the data enables to focus the analysis on the fluctuations about the trend, and this is what the pattern is in the camera plane: a small fluctuation of 4-5px around a mean value.
The process of detrending has been done on a window of 7 px, i.e. the mean of a set of 7 pixels has been subtracted from the intensity value of the pixels within the set.

6.3.3 Thresholding

At this stage of the pipeline the image has been processed correctly and it is easier to segmentate the two different sets of lines. Unfortunately, this is not enough. We want to isolate only the pixels where is present the projected pattern. This section compares two methods of thresholding the image: a single threshold, and an adaptive threshold.

Simple Thresholding

The basic thresholding definition is straightforward: if pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The threshold is global, i.e. all pixels of the frame are compared to that threshold. This is a nice solution for an image with global uniform lighting conditions. In practice this is hard to be true. Ambient Lights constitute a big issue using this algorithm.

Adaptive Thresholding

A global value of thresholding may not be good in all the conditions where image has different lighting conditions in different areas. In this case, the adaptive thresholding is the most robust approach. The algorithm calculate the threshold for a small region of an image. The threshold can be either the mean of the region, or a cross-correlation with a Gaussian window. So, it uses

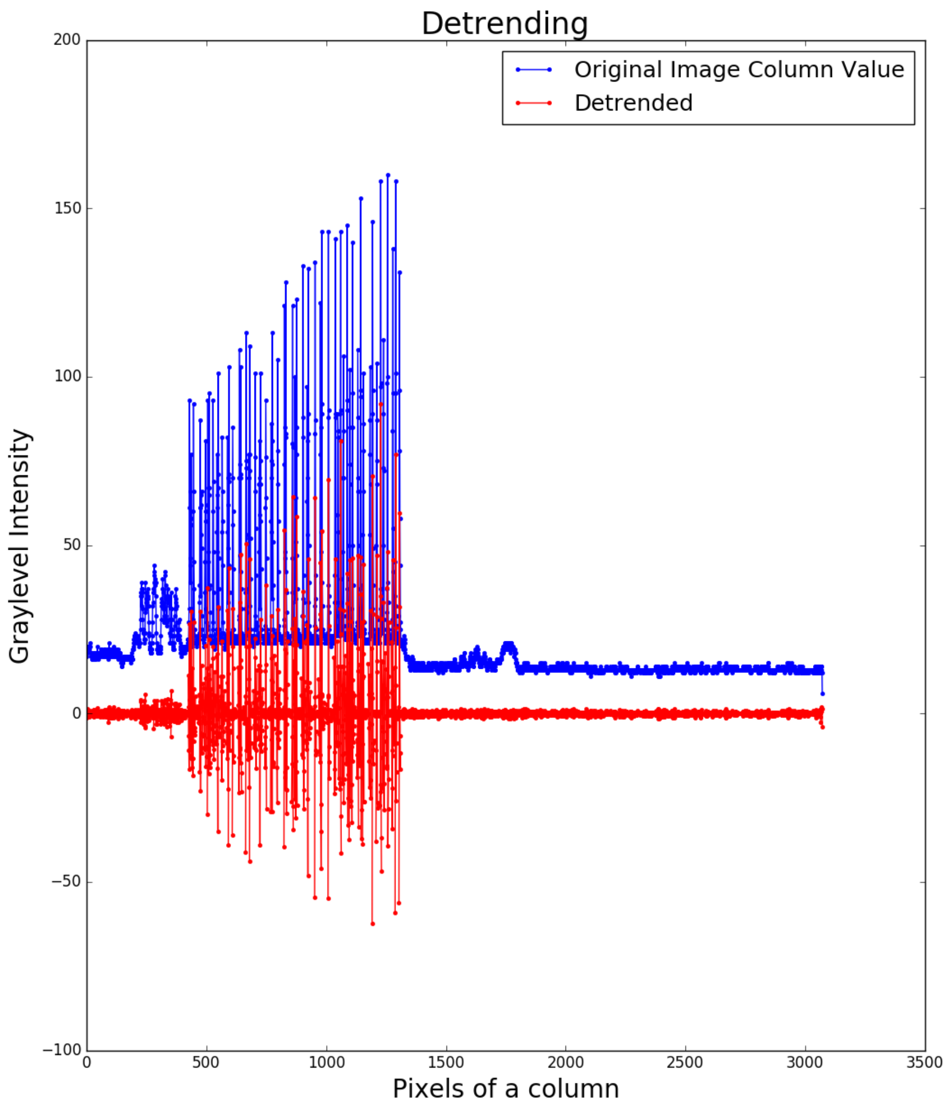


Figure 6.11: Detrending: in this image has been chosen the channel containing the horizontal stripes of the pattern and has been taken a column. The red graph represents the detrended values of intensities. The big spikes are the pattern's stripes. It can be noticed the attenuation of the noise around the pattern.

different thresholds for different areas of the same image. In the image 6.12 it is possible to see how much more robust is the adaptive approach.

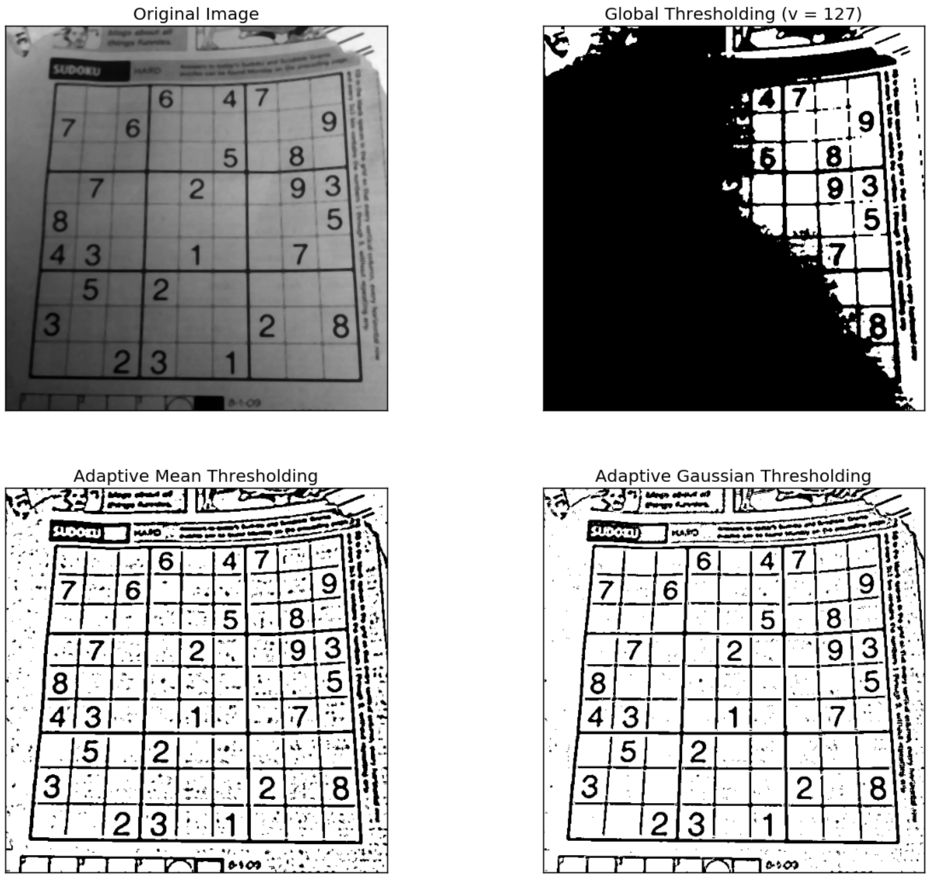

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Figure 6.12: Threshold vs Adaptive Thresholding: Ambient Lights can be a challenge for the global threshold.

6.3.4 Morphological Operations

In theory projecting a pattern with two separates primary colors allows an easy segmentation between horizontal and vertical lines: it is sufficient to take the corresponding image channel. Actually, this may be not true. As it has been written in the system design chapter, the color filter of the camera's sensor is not perfect and do not distribute color in an uniform way.
Hence, it is needed to design an algorithm to ensure the segmentation between vertical and horizontal lines. This separation must be as accurate as possible,

because the reconstruction algorithm depends on the concept of *horizontal and vertical neighborhood* of a pixel.
Morphological Operators are able to filter the vertical from the horizontal lines, as it is shown in Figure 6.14. Morphological transformations are some simple operations based on the image shape. They are normally performed on binary images. They need two inputs, one is the original image, and the second one is called *structuring element or kernel* which decides the nature of operation. Two basic morphological operators are *Erosion* and *Dilation*.

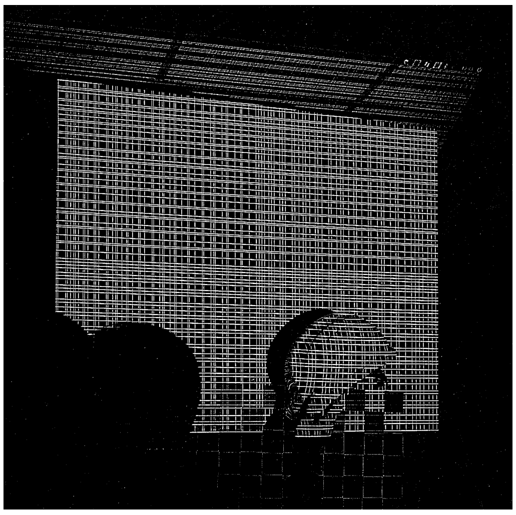


Figure 6.13: The Red channel after the Adaptive Threshold

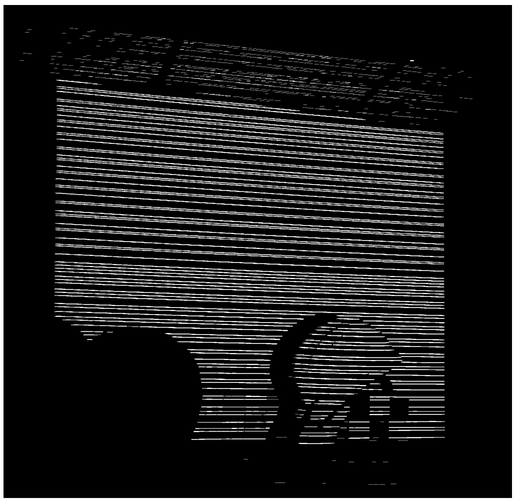


Figure 6.14: The Red Channel after the Morphological Filtering

Erosion

The erosion, as it is suggested by the name, erodes away the boundaries of foreground objects. In order to do so, the kernel slides through the image, like a 2D convolution. A pixel in the original image is considered on, only if all the pixels under the kernel are on, otherwise it is eroded. What happened is that, all the pixels near the boundary are discarded depending on the size of the kernel.

Dilation

The dilation is the opposite operator of the erosion. A pixel is considered to be on, if at least one pixel under the kernel is on. This operator increases the white region in the image or, the size of foreground objects.

Opening

The opening operation applies an erosion followed by a dilation with the same kernel. This operation is useful to remove noise (or unwanted part) from the image without changing the size of the foreground objects.
The opening operation has been used for removing the residual vertical part of the grid in the *horizontal channel*, and the opposite. In order to filter out the residual part of the grids, it is sufficient to choose a kernel similar to the characteristics of the line that it is wanted to maintain: e.g., for filtering out the vertical lines, we choose a rectangular kernel with height equals to 1px and a certain width.

6.3.5 Center Of Gravity Extraction

At this stage of the pipeline, the system has been able to generate two binary images: one contains only the vertical part of the grid, the other one contains the horizontal part.
These two images are applied as masks to the original grayscale image and this process generates two grayscale images with separated lines. The network extraction algorithm wants as input an image where the lines have unitary width (thinned), in order to correctly reconstruct the topology of the pattern. The problem to reduce a line to unitary width, i.e. a width of one pixel, is to correctly guess where is actually the line, at which exact pixel. In fact, a line imaged by the camera at a distance of 2.5m may be not a perfect unitary width line. To estimate precisely the position of the line, for thinning, two strategies can be applied: peak detection and COG.
Peak detection is the simplest algorithm, it takes the maximum graylevel value of the line. This approach is not robust, in fact, in presence of noise the location of the peak can change, and consequently the location of the line.
The Center Of Gravity (COG) method, as Bailey writes in his survey [6], gives

more robustness to noise and it is able to reach sub-pixel accuracy on the estimation of the line location. It works as following:

- Find the connected components of the picture
- Extract the bounding box of each component
- Mask the bounding box in such a way to have only the desired line in the image
- Performs a weighted sum of the pixel intensities column by column for horizontal lines, and row by row for vertical lines. This operation retrieves the accurate position of the row (the column) of horizontal lines (vertical lines), column by column (row by row) of the bounding box.

Once this process has been done for all the lines present in the images, it generates two images one with horizontal thinned lines and one with vertical thinned lines. In order to simplify the process of network extraction, the pixels of the horizontal image are set to 0 if off and to 1 if on; the pixels of the vertical image are set to 0 if off and to 2 if on.
In this way, when the two images are added to reconstruct the full grid, the corners has pixel value equal to 3 and they can be immediately recovered.

6.4 2D Network Extraction

The Reconstruction Algorithm is based on the principle of neighborhood of pixels: horizontal and vertical. This means that it is needed to reconstruct the topology of the grid as it has seen by the camera.
The network is extracted by following a simple algorithm:

- Get the two thinned images from the Line Detection Pipeline and add them in a single image
- Going through the image column by column, looks for a 3, i.e. for a corner of the grid
- When it finds a corner grid it follows the horizontal paths that start from the corner. It is done by following the pixel that has value of 1 in a 3×3 neighborhood of the corner.
- When finds a new corner (a 3 in the neighborhood), add an edge to the graph and go on
- Do the same with the vertical paths.

This procedure is repeated until all corners has been found. The output of this stage is an undirected graph where the vertices are the corner of the grid, and the edge are the lines that connects them. For the sake of simplicity, this procedure generates other two graphs: one for the horizontal lines, and one for the vertical lines.

6.5 Reconstruction Algorithm

The Reconstruction Algorithm follows what it is written in [4]. It is reported here for the sake of completeness.

6.5.1 Overview

It is a structured light approach consisting of a data projector and a single color camera. Both of which are calibrated with respect to a world coordinate system. The projector projects a known static grid pattern onto the 3D surface. An image of the object illuminated by the projected grid and captured by the camera is used to determine the depth information of points illuminated by grid crossings and observed by the camera. Producing the correct depths depend on being able to solve the so-called correspondence problem. Matching pairs of features in the camera and projector images need to be identified, as in the perspective of stereo vision. Note that after correspondences are established, triangulation is trivial and depth information can be obtained easily.
An exhaustive search for the correspondences of grid points observed by the camera is intractable. However, using a pattern which disclosures spatial neighbor information for all the feature points, e.g. a grid pattern, the correspondence problem for the whole grid is simplified to identifying the correspondence for a single grid point, i.e. all grid points in a network can be uniquely identified based on a single correspondence. This can be done efficiently without assuming relative ordering of grid points and by exploiting simple geometric constraints.
At first, finding a single correspondence might seem to require a search through the whole grid points in the projector image. However, exploiting the epipolar constraints, the search space reduces to a single line. It shrinks even further because we know the points could have come from only grid crossings along the epipolar line.

6.5.2 Reconstruction using a grid pattern

The reconstruction algorithm takes as input the networks extracted in the previous step. However, in general they may not result in a single connected grid network due to shadows, occlusions and sharp edges. The network is typically composed of several connected components.

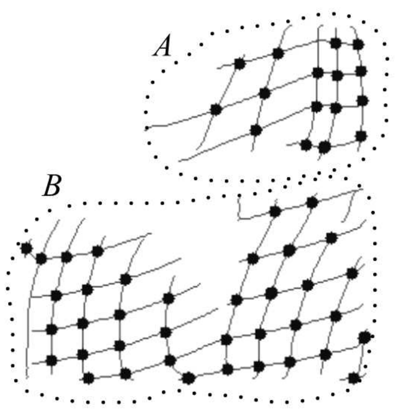


Figure 6.15: The grid networks can present holes: this is an example of two connected components.

This approach solves the correspondence problem for each connected component independently. The steps of the algorithm are as follows:

- Find a connected component
- Pick a grid point u in the selected connected component.
- Using its epipolar line, find possible candidates for correspondence in the projector image. This is done by imposing that the distance between the epipolar line and a grid crossing in the projector plane has to be less than a certain threshold.
- For each candidate, propagate the correspondence to all grid point in the network.
- Choose the solution which matches the projected grid best via a 1D search.

6.5.3 Propagation Algorithm

Having obtained a list of correspondence candidates for a grid point, the rest of the points can be inferred quite efficiently and without assuming relative ordering. We denote grid points horizontally linked to u as $u_{h-neighbors}$ and points vertically linked as $u_{v-neighbors}$. Both $u_{h-neighbors}$ and $u_{v-neighbors}$ are lists of grid points all linked to u. It is known the $u_{h-neighbors}$'s row correspondence and the $u_{v-neighbors}$'s column correspondence in the projector plane. From this knowledge, it can be computed the $u_{h-neighbors}$'s column correspondence and the $u_{v-neighbors}$'s row correspondence by intersecting the epipolar line of the neighbor point with its known correspondence. As a result, we obtain both row and column correspondences for u's neighbors. For points that are not u's horizontal or vertical neighbors once the correspondences for the neighbors have been computed, they can propagate their solution to their own neighbors. Below, it is described an efficient algorithm that traverses each grid point at most once and guarantees that every point in the grid network is given a correspondence.

Algorithm 1 Propagation Algorithm

```
1: {The correspondence for $u$ denoted as $(\pi_v(u), \pi_h(u))$}
2: $Q \leftarrow u$ {Initialize queue Q with $u$}
3: while $Q$ is not empty do
4:     $e \leftarrow Q[1]$ {Get the first element from Q}
5:     if $\pi_v(e)$ exists then
6:         Compute $\pi_h(e)$
7:         $\pi_h(e_{h-neighbors}) \leftarrow \pi_h(e)$
8:         Remove $(Q \cap e_{h-neighbors})$ from $Q$
9:         Add $(e_{h-neighbors} \setminus Q)$ to $Q$
10:    if $\pi_h(e)$ exists then
11:        Compute $\pi_v(e)$
12:        $\pi_v(e_{v-neighbors}) \leftarrow \pi_v(e)$
13:        Remove $(Q \cap e_{v-neighbors})$ from $Q$
14:        Add $(e_{v-neighbors} \setminus Q)$ to $Q$
```

6.5.4 Search for Correspondence

The propagation provides a set of solution candidates for a given grid network. As a final step, we need to identify the correct solution through the comparison of computed grid points $s' \in S'$, and the projected grid points $s \in S$ on the projector image, where S' is the set of computed grid points, and S is the set of projected grid points.

Ideally, for the correct correspondence, the computed grid points s' must appear exactly on the projected grid point s. However, this, in general, may not be satisfied due to imperfections in calibration and image processing. Provided that the noise level is not excessively high, the location of a computed grid point s' still provides a good neighborhood which includes the true correspondence s.

Considering that a computing grid could move along any xy direction due to 2D imperfections (noise plus imperfections in calibration), it is assumed that s' came from one of the four closest neighbors in the projected grid $s_{neighbors}(s')$. The error function is defined to be the minimum cost of choosing either one of these neighbors.

It is expected that the correct choice s satisfies the corresponding epipolar line u', that is the camera image point corresponding to the computed s'. Thus, a single cost is

$$d(s', s_{neighbors}(s')) = \min_{s \in s_{neighbors}(s')} (|L(u') \cdot s|) \tag{6.1}$$

which is defined to be the minimum Euclidean distance of a discrete neighbor to the epipolar line $L(u')$. Summing this distance metric for all the intersections, it can be defined the error function

$$E(u, s) = \sum_{s' \in S'} d(s', s_{neighbors}(s')) \tag{6.2}$$

which changes for various candidates s along the epipolar line $L(u)$. The best match is the one that minimizes the error function.

6.5.5 Grid Considerations

This result of searching a correspondence is unique for a De Bruijn pattern because it empirically ensures the uniqueness of the correspondence also in presence of imperfections in the image. When the spacings between grid lines are uniform, the presence of noise may create ambiguities in the searching of the correspondence.

Another important consideration on the pattern is related to the alphabet of the De Bruijn sequence used to generate the grid.

To decide on the parameters (k, n), consider the two extreme cases: when k is large and n is small and vice versa. In the first case, there will be many different spacings which will eventually lead to large spacings since the spacing alphabet is discrete. Thus, the reconstruction will get sparse. However, even a very small patch of the pattern can easily be recovered since n is small. On the other extreme, when k is small and n is large, the pattern will be composed of a few large unique patches. Thus, reconstruction will require a small number of patches to be identified since only a few can be enough to cover a whole surface. Note that one may not be able to reconstruct a surface if the detected patches are smaller than the needed size. This decreases the robustness to noise, depth discontinuities and texture.

6.6 Implementation Details

The 3D Reconstruction System is composed of three main modules:

- The Grid Segmentator
- The Network Extractor
- The Correspondence Finder

Inside these main modules there are sub-modules that can be mixed together to adapt the behavior of the system to the working conditions of the place where it has to be used. The modules are written in Python3 and in C++14, for performance reasons. The C++14 modules makes use of the Boost.Graph Library for managing undirected graphs. Moreover, to parse JSON files they use the "JSON for Modern C++" Library (`https://github.com/nlohmann/json`).

6.6.1 The Grid Segmentator

The Grid Segmentator Module is responsible for the correct separation between the grid pattern and the background. It has been written in Python3 and uses the version 3.1 of the OpenCV Library.
The module may be configured to work following a sequence of the following sub-modules:

- Colors Segmentation: RGB, CIELAB, or HSV
- Thesholding: Global or Adaptive
- Detrending
- Contrast Enhancement: Histogram Equalization, or CLAHE (Contrast Limited Adaptive Histogram Equalization)
- Morphological Operations
- COG Line Detector

These modules are parameterized using command line arguments. It takes as input an image, it apply an undistortion map on it and separates the horizontal from the vertical lines. As output, it generates two files one for the horizontal stripes and one for the vertical stripes.

6.6.2 The Network Extractor

The Network Extractor is the module responsible of reading the output file of the segmentation stage and produce a graph. It has been developed in C++14 using the Boost.Graph Library. It follows the algorithm described in the previous sections.
It produces as output, three files for the graph in the DOT format:

- one contains the full graph, and it is useful for connected component analysis;
- one contains the horizontal lines graph
- one contains the vertical lines graph

Moreover, it produces another binary file for associating the index of the vertices to specific points on the image.

6.6.3 The Correspondence Finder

The Correspondence Finder implements the algorithm proposed by Ulusoy et al. [4]. It has been developed in C++14 with the use of the Boost.Graph Library, mainly for the connected component analysis on the graph.
It extracts two sets of lists from the the horizontal and vertical graphs generated by the Network Extractor. These lists are useful to find the neighbors, horizontally and vertically linked, of a node during the algorithm.
Thanks to the Boost.Graph Library functions, it divides the graph into separated connected components subgraphs. For each subgraphs, it applies the algorithm described in the sections above.
The linear algebra computations of this module, such as the Fundamental Matrix and the computation of the epipolar line, has been done by using the functions provided by the OpenCV Library, that is based on the Eigen Library.

Finally, the three-dimensional world coordinates has been recovered by performing a triangulation, by using an OpenCV function called *triangulatePoints*. The 3d points are then stored in a xyz file, and can be imported in programs such as MeshLab, Matlab for more accurate analysis.

Chapter 7

Software Architecture

The software architecture has been defined underlining the following needs and objectives:

- Flexibility in the choice of the algorithms
- Support for different types of sensors
- Support for the parallel and distributed computing

Starting from the previous considerations it has been chosen to use a network of interconnected machines based on Ubuntu Linux (16.04 LTS). On these machines it runs a middleware software that supports the high-level data exchange, the activation of modules and their diagnostics. The chosen software is the Robotic Operating System (ROS) that is the de-facto standard for robotic applications and, more in general, of vision robotic oriented.
The software system utilizes a set of scripts and tools written in C++14 and Python3. The developed software makes use of a set of reference libraries in the field of Computer Vision: Eigen for matrix computations, OpenCV and the Point Cloud Library. These libraries have a huge user's community and provide a lot of testing set. A graphical overview of the software architecture

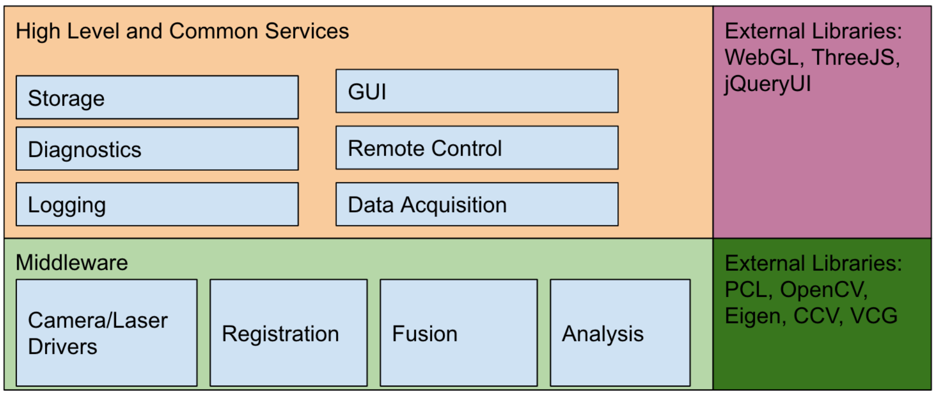

High Level and Common Services
Storage
Diagnostics
Logging
GUI
Remote Control
Data Acquisition
External Libraries: WebGL, ThreeJS, jQueryUI
Middleware
Camera/Laser Drivers
Registration
Fusion
Analysis
External Libraries: PCL, OpenCV, Eigen, CCV, VCG

Figure 7.1: The layers of software used to develop the vision system

7.1 Data Management

The data management system of this project is relevant, because it must guarantee the consistency of the acquired data, the correct management, retrieval, and since the amount of data is big, the efficiency of their manipulation.

7.1.1 Data Types

The type of data managed by the system can be identified in these categories:

- Raw RGB Images
- Point Cloud given by the 3D reconstruction
- Intermediate Processing Data Structures, like surface features or volumetric structures
- Metadata Reports generated by the algorithms
- Alternative representations of the point cloud: mesh, hierarchical structures optimized for a distributed system
- CAD models and other representations for the history management

7.1.2 Pipelines Management

The management of the working pipelines makes a distinction between the phase of acquisition and the post-processing. The same can be said for the

acquired raw data and the processed one. The acquisition data comes at high rate, onerous for the system, and it requires a logging mechanism that enqueue the information as fast as possible. Once they are acquired, the system gives a backup system for their recovery.
The processed data management is slightly different, anyway it requires a mechanism to create different pipelines of processing and replay it. This is related to the principle of *data provenance*. Data provenance documents the inputs, entities, systems, and processes that influences data of interest, providing a historical record of the data and its origin.

7.1.3 Data Storage

Metadata can be stored easily in a database RDBMS (e.g., Postgres or MySQL) that manages the relations between them through its relational system. In the specific case of the calibration and the 3D Reconstruction system, the metadata has been stored in the JSON format.
A different approach is to be taken into account for the huge amount of data coming from the vision system. Databases provide mechanisms for storing big data, such as BLOB, Binary Large Objects, but they do not meet the speed requirement of the project. Thus, this kind of data is memorized separately from the database.
One of the most common solution is the system HDF5. It is a container of tree-structured data that is able to store multidimensional arrays, annotations and links. Although it is very common in the scientific field, it shows some drawbacks:

- Cumbersome Specification
- Existence of an unique complicated open source library, written in C
- Some limitations recognized in the data management (e.g. removal)
- Limitations in the memory-mapped access of data

Then, it has been chosen to use the ROS bag system. ROS bag is the mechanism provided by ROS for data storage. They provide storage in compressed data chunks, automatic description of the content, indexing. The messages embedded in a ROS bag have to be pre-loaded. These bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

7.2 ROS Organization

In ROS is possible to define packages. A package is the basic unit of ROS that have the goal of providing an easy mechanism for software reusing. Packages can contain data, scripts, messages and even services. For the purpose of the thesis we can divide the process in some packages:

- Camera Calibration, the package that calibrates the camera
- Projector-Camera Calibration, the package that calibrates the intrisics of the projector and the extrinsics of the system
- Image Acquisition, the package that acquires images from the camera, and creates the dataset for performing the 3D reconstruction. It stores into ROS bags a frame with the pattern, and a frame without the pattern, and so on. The characteristics of the image, such as the presence of the pattern, are encoded in the indexing of the ROS bag.
- Image Extractor, the package that filter the ROS bags generated by the Image Acquisition package, to separate the images containing the pattern from the images without it.
- 3D Reconstruction, the package that performs the 3D Reconstruction. This package is parameterized in order to adapt to different working modes: with different number of cameras, with different types of pattern.

This packages can contain also other packages. This behavior is obtained by means of ROS meta-packages. This approach improves the modularity of the system and, since this packages are run as nodes in a distributed system, it improves the interoperability between the modules. Packages can be also easily parameterized by means of the roslaunch mechanism. Roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on. Parameterization is a fundamental properties, since it allows to rapidly reconfigure the system in different environments.

Chapter 8

Test and Results

This chapter contains the description and some results of the test performed for assessing the correct behavior of the algorithms that has been implemented in this work. The test has been made with the Emergent Camera HT12000 with a resolution of 4096×3072px and a DLP Dell 5100 Projector with a resolution of 1400×1050px. The devices are mounted on a rigid structure composed of Bosch Rexroth aluminum elements.

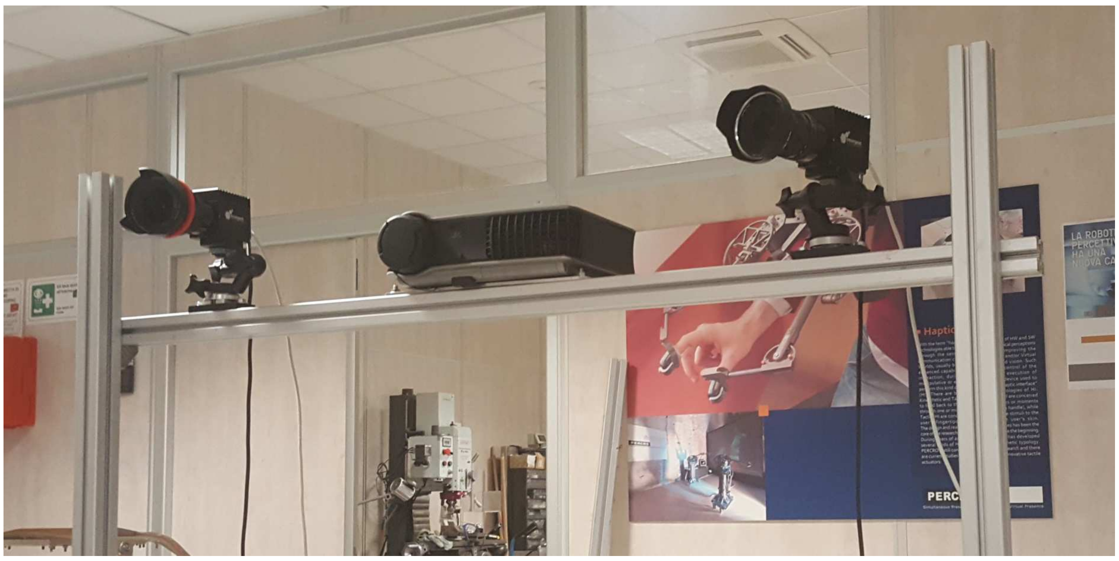


Figure 8.1: The rigid structure where are mounted the devices

8.1 Calibration System

8.1.1 Camera Calibration Results

The Camera Calibration has been performed using a plane made in Forex mounted on a tripod at a distance of 2.5m from the camera. It is important to calibrate the camera near to the working conditions, since the process of calibration is basically an estimation. Onto the plane has been glued a chessboard. The chessboard has been designed to occupy an area as large as possible in the camera frame. So, it has been chosen to be printed in the A1 paper size format (594×841 mm). The chessboard has an odd number of rows and an even number of columns to make the recognition of corners orientation-invariant.
The tripod guarantees a better stability of the plane, limiting the oscillations. Moreover, it is more practical to place the plane in different orientations in order to make the calibration more robust. The figure below represent the

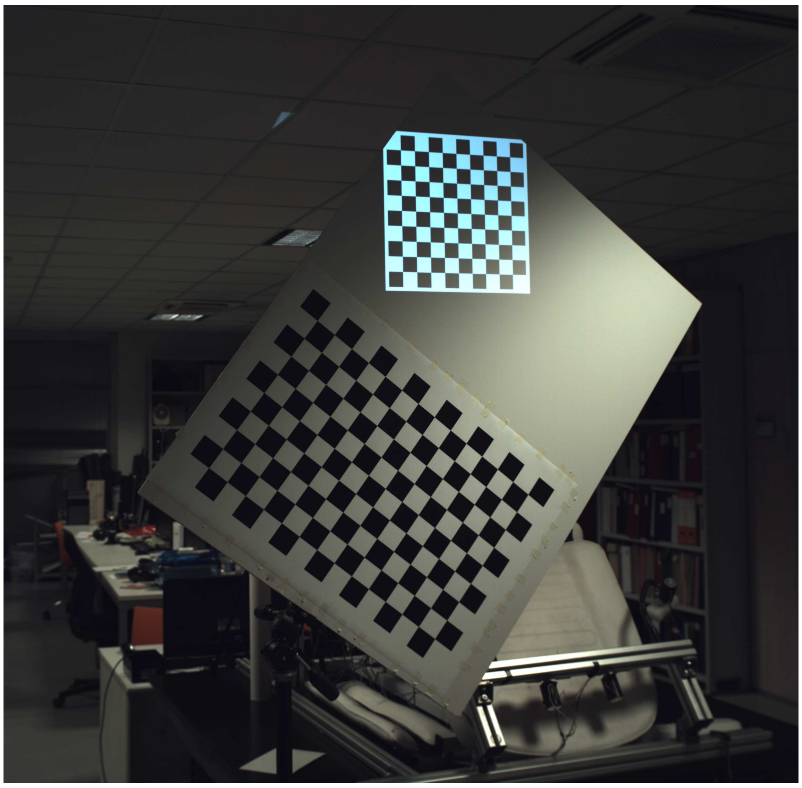


Figure 8.2: A photo of the calibration plane

Reprojection Errors of the calibration data set. As it can be seen, the errors forms a sort of circle around the point $(0, 0)$. This is a sign of a good calibration.

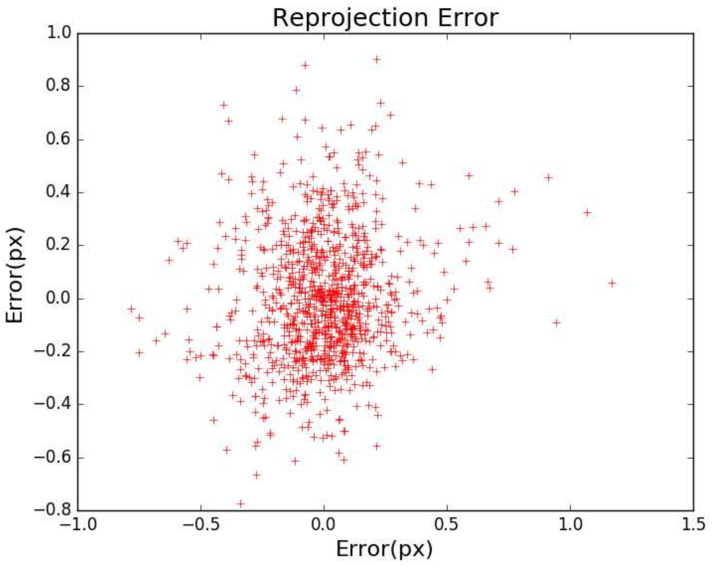


Figure 8.3: The distribution of reprojection errors in camera's calibration

The RMS of the calibration is 0.32px.

8.1.2 Projector Calibration Results

The Projector Calibration has been performed by projecting a chessboard pattern onto the calibration plane. It has been developed an OpenGL tool to control the size, the position and the type of the chessboard to project in order to project correctly the pattern onto the plane. The figure below shows the reprojection errors expressed in pixels.

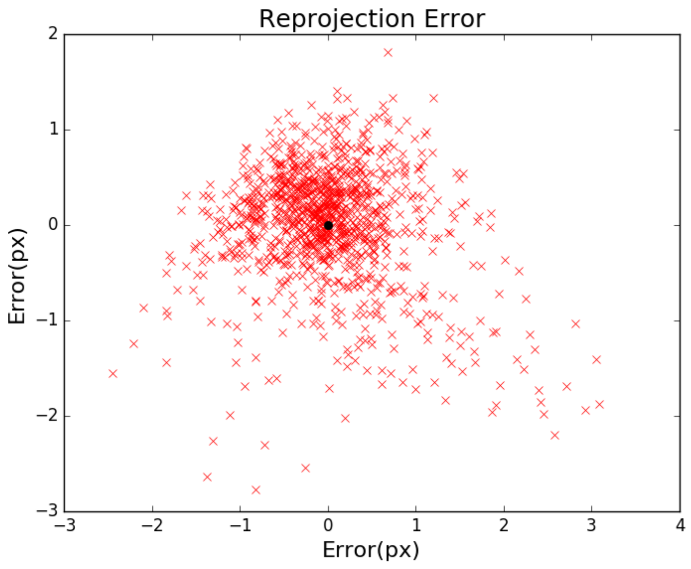


Figure 8.4: The distribution of reprojection errors in projector's calibration

The RMS of the calibration is 0.95px. The projector calibration's tool gives also the Euclidean transformation of the projection with respect to a reference frame placed inside the camera's sensor.

8.2 3D Reconstruction System

For analyzing the behavior of the 3D Reconstruction System we take an image as a reference throughout the following sections from the segmentation to the point cloud. The pattern that has been used for these tests is a De Bruijn colored grid, with horizontal lines colored in red and vertical lines in green. The alphabet size of the De Bruijn sequence has been set to 125, with a combination of $(k, n) = (5, 3)$.

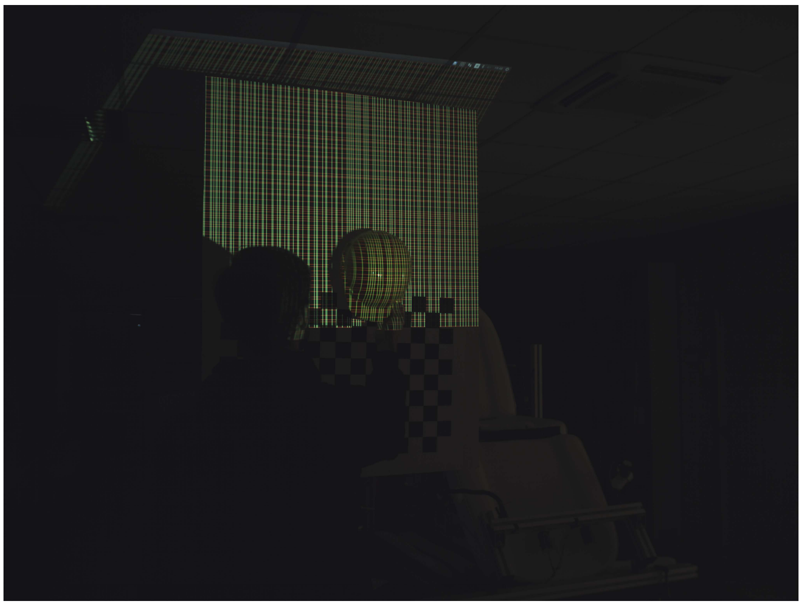


Figure 8.5: A capture of the dataset for the 3D Reconstruction System Test

8.2.1 Segmentation

The process of segmentation consists of extracting the lines from the image and generating the input data for the network extraction. The segmentation pipeline has been described in chapter 6.
As we can see, the majority of the grid lines has been correctly detected, and the level of noise is very low.

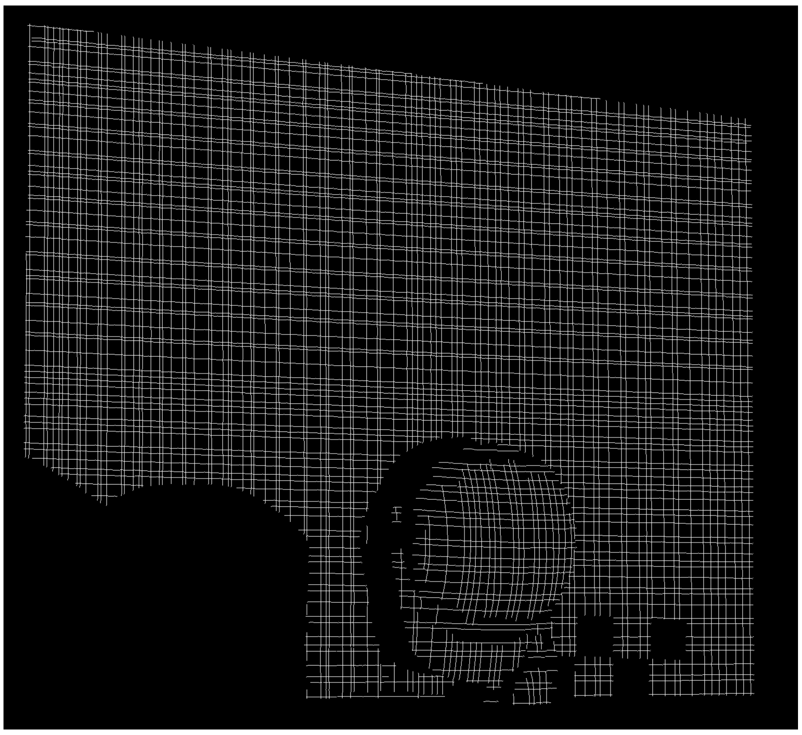


Figure 8.6: A capture of the dataset for the 3D Reconstruction System Test

8.2.2 Network Extraction

In this section, it is shown the result of the process of network extraction from the segmented image.
In the picture 8.7, the vertices of the network are the brightest white circles and the edge are depicted as lines connecting the grid crossing. The number of vertices that this process has been able to retrieve is almost equal to 6000. Considering that the total number of nodes of the grid is 15000 and half of the projected area did not cover the plane, this is an encouraging result.

8.2.3 Correspondence Search

The extracted network is the input file of the correspondences search algorithm. During the check of correspondence candidates it generates an error function, that provides a mean for establishing the quality of the reconstruction.
As we can see from the figure 8.8, the error function has a global minimum, and the corresponding candidate is, therefore, the *unique* correct solution for

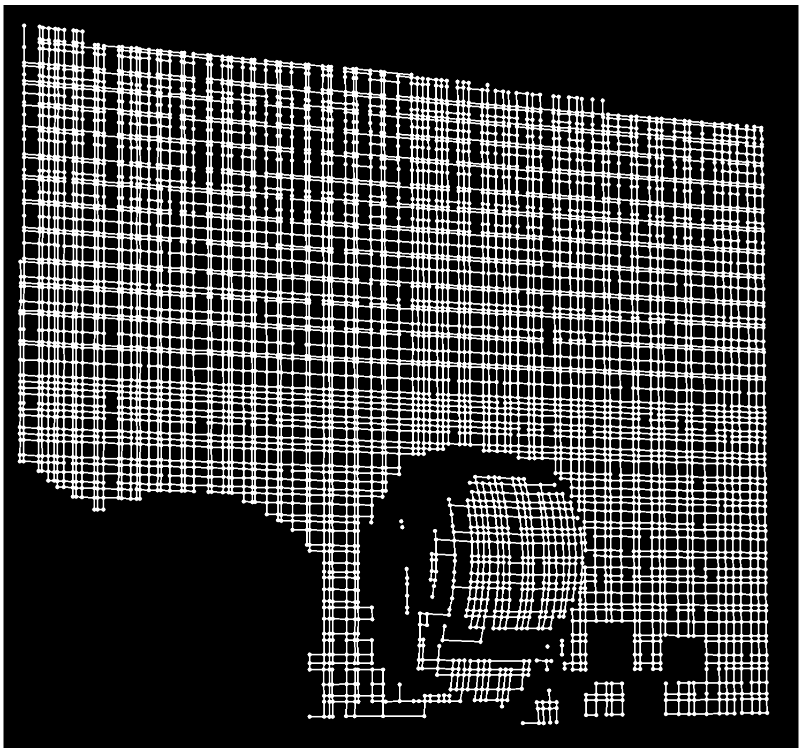


Figure 8.7: This picture gives a graphical representation of the network extracted from the pattern. The bright circles on the image are the vertices of the network that correspond to the grid crossings. Edges are represented as lines that links two vertices.

the correspondence problem. This has been made possible thanks to the use of De Bruijn spaced grids, which have the property of *window uniqueness*. Moreover, it can be seen that the error is very low, thus guaranteeing an accurate reconstruction of the point cloud of the object.

8.2.4 Point Cloud

Finally, once the correspondences has been computed it is possible to perform a triangulation and recover the point cloud of the scene.
With the point cloud is possible to perform some easy analysis. For example, if we think that the train has to be at a Z distance of 2.5m the system could give a raw estimation of possible defects by comparing the z-coordinate of points with the selected threshold level. In this capture the plane was at a distance

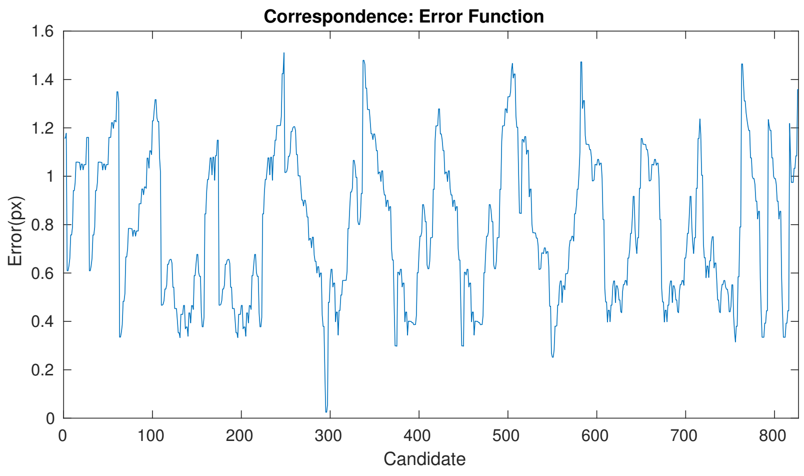


Figure 8.8: The error function, related to a connected component, computed during the process of correspondences searching. We can see that has a global minimum that corresponds to the correct solution of the problem. This is possible thanks to De Bruijn spaced grids' property of *window uniqueness*.

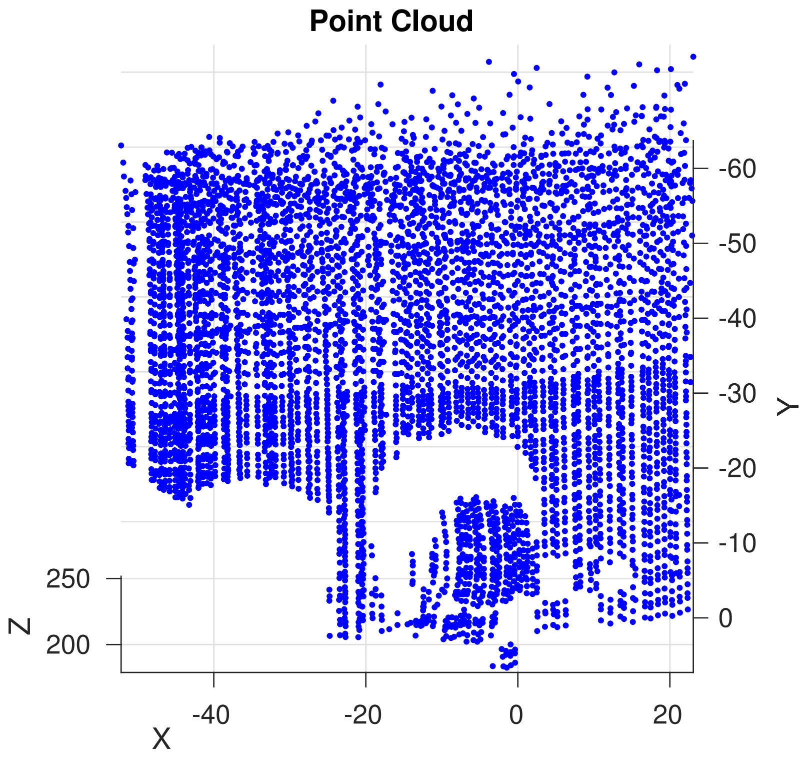


Figure 8.9: The reconstructed 3D Scene

of 2m from the vision system, thus it is possible to set the threshold equal to 1.9m. Whatever is nearer than this distance is depicted with a red color.

The result of this test is shown in Figure 8.10.

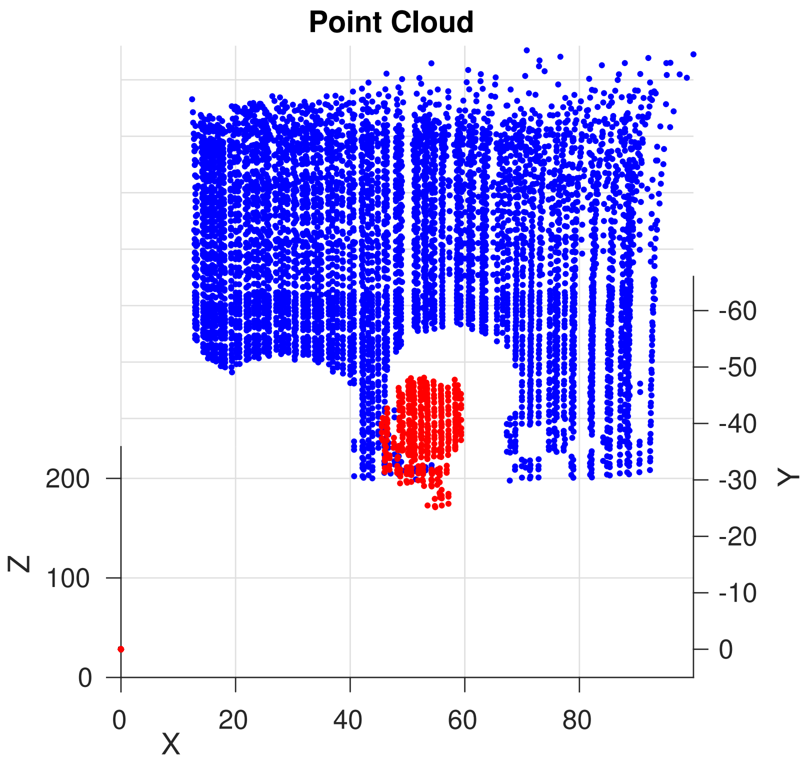


Figure 8.10: An example of processing on a point cloud. The helmet is colored in red because it stands in front of the plane, at a distance of 1.8m from the vision system.

Chapter 9

Conclusions

In this work has been analyzed an application in the railway industry. It has been shown the process of recovering the requirements from the analysis of the scenario. Successively, it has shown how to design the vision system in order to meet the requirements. This process implied the knowledge of the basic principles of the devices and some comparative analysis between different technologies. Then, we presented some general theory concepts of computer vision systems: the pinhole camera model, some concepts of stereo vision, the epipolar geometry and a brief taxonomy of structuring light techniques.
In the following chapters it has been analyzed the calibration system, both for the camera and the projector. For the calibration of the projector has been used the procam approach by Falcao described in [3]. This part of the system sets a bound on the accuracy of the reconstruction.
Then, we showed a complete pipeline to realize a 3D Reconstruction System, robust to noise and lighting conditions, and able to reconstruct large areas. The robustness of the reconstruction has been obtained with a carefully designed image processing pipeline, that is able to extract most of the projected lines captured by the camera, with sub-pixel accuracy.
After that, it has been proposed a simple algorithm to generate a 2D Network of the grid pattern, in order to gain the topological information, in particular the concept of neighborhood of a grid crossing. Then, it has been implemented the algorithm proposed by Ulusoy et al. in [4], to find the correspondence between points of the camera plane and the projector plane. Once the correspondence has been found, the three-dimensional world coordinates of the detected grid

crossing has been computed with a simple triangulation.
Then, we analyzed the challenge imposed by the huge amount of data to the software architecture. It has been described a way of managing all the modules produced for this thesis for the integration and the modularization. The solution used ROS, the Robotic Operating System, to manage nodes and store the huge quantity of information into ROS bags. Finally, it has been shown the result of some of the tests performed in the laboratory.

Bibliography

[1] J. Salvi, J. Pages, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern recognition*, vol. 37, no. 4, pp. 827–849, 2004.

[2] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[3] G. Falcao, N. Hurtos, and J. Massich, "Plane-based calibration of a projector-camera system," *VIBOT master*, vol. 9, no. 1, pp. 1–12, 2008.

[4] A. O. Ulusoy, F. Calakli, and G. Taubin, "One-shot scanning using de bruijn spaced grids," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on.* IEEE, 2009, pp. 1786–1792.

[5] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphic Gems IV. San Diego: Academic Press Professional*, 1994, pp. 474–485.

[6] D. G. Bailey, "Sub-pixel estimation of local extrema," in *Proceeding of Image and Vision Computing New Zealand*, 2003, pp. 414–419.

[7] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[8] R. Szeliski, *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

List of Figures

List of Tables

Ringraziamenti

Questo lavoro di tesi conclude il mio percorso da studente universitario. Sette anni tanto intensi quanto importanti per la mia crescita umana e professionale. Per questo motivo, ringrazio tutti i Professori che, nel bene o nel male, mi hanno aiutato a diventare un professionista più forte e capace.

In modo particolare voglio ringraziare il Prof. Carlo Alberto Avizzano e il Prof. Emanuele Ruffaldi, relatori di questo lavoro di tesi, per la loro estrema disponibilità e competenza.
E' altresì doveroso ringraziare di cuore tutti i ragazzi del Laboratorio di Robotica Percettiva della Scuola Superiore Sant'Anna (PERCRO). In special modo: Erika Di Stefano, Paolo Gasparello, Andrea Piarulli, Alessandro Graziano, Michele Mambrini e Alessandro Nicoletti. Mi hanno fatto sentire veramente a casa ed è stato bello poter lavorare quotidianamente con loro.

Ringraziate le persone che mi hanno supportato durante il lavoro di tesi, è la volta di chi mi ha 'sopportato'.

Un grande ringraziamento va ai miei genitori che hanno investito tanto su di me e mi hanno sempre sostenuto.

Una grande spinta a non mollare mai, anche nei momenti più difficili, è venuta anche e soprattutto dagli amici e colleghi. Insieme a Emiliano, che conosco dalla prima superiore, e Daniel abbiamo composto il gruppo di livornesacci di questo corso di studi. Abbiamo formato un gruppo molto unito e ci siamo scambiati motivazioni e consigli fondamentali fino a raggiungere risultati eccellenti in un percorso di studi così difficile. La prova vivente del detto *l'unione fa la forza.*

Un abbraccio a tutti gli amici che hanno dovuto sopportare spesso lamentele e grandi assenze da parte mia. Voglio ringraziare in particolar modo i miei due migliori amici: Francesco e Dario.

Un pensiero anche a chi non c'è più e avrebbe voluto esserci.