Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Development of pilot system of artificial vision for the acquisition of a point cloud using 3D vision technologies

A project presented by

Deyby Maycol Huamanchahua Canchanya

Submitted to the

School of Engineering and Sciences

in partial fulfillment of the requirements for the degree of

Master in Engineering

specializing

Automation and Control

Monterrey, Nuevo León May, 2015

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

School of Engineering and Sciences

The committee members, hereby, certify that have read the thesis presented by **Deyby Maycol Huamanchahua Canchanya** and that it is fully adequate in scope and quality as a partial requirement for the degree of **Master of Engineering** in **Automation and Control**,

Dr. Federico Guedea Elizalde
Tecnológico de Monterrey
Assistant Professor of CIDyT
Pricipal Advisor

Dr. Adriana Vargas-Martinez
Tecnológico de Monterrey
Director of Master in Automation and Control

Monterrey, Nuevo Leon, May, 2105

Declaration of Authorship

I, **Deyby Maycol Huamanchahua Canchanya**, declare that declare that this project titled, **"Development of pilot system of artificial vision for the acquisition of a point cloud using 3D vision technologies"** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Deyby Maycol Huamanchahua Canchanya
Monterrey, Nuevo León, May, 2015

Acknowledgements

To my advisor, Dr. Federico Guedea for supporting me and guide me throughout this process and be part of this achievement.

To my family, to be a support for me at all times, support me unconditionally and giving me tranquility in the most difficult moments throughout the process my master.

To DEMAQ Technologies, for giving a friendly atmosphere and allowing develop the project of the best way and Salvador Ramirez for the contribution during the project.

To ITESM, for giving qualified teachers and quality knowledge that will allow me to grow and develop in many areas in my professional life.

To CONACYT, for helping me with my support around the time of my master that allowed me to study without financial worries.

Development of pilot system of artificial vision for the acquisition of a point cloud using 3D vision technologies

Deyby Maycol Huamanchahua Canchanya
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2015
Dr. Federico Guedea Elizalde

One of the goals of artificial vision is to permit that a computer analyzed a real scene, as if a person does. To achieve this purpose it is necessary to create a 3D model of that scene using a reconstruction. 3D reconstruction is the process by which the shape and the appearance of a three dimensional object or scene from a volume by analyzing the digital information provided by different types of sensors is recovered. The sensors may be passive (not interact with the object, such as different types of cameras based on the light of the visible spectrum) or active (the interaction comes from an object in response reflected waveform that is captured by the device).

On the other hand, 3D reconstruction has several applications, such as robot navigation, allowing it to know in what part of the scene is located and being able to plan its movements without needing human help. It is also useful for determining quantities such as distances, areas or volumes, which may be applicable for quality controls as it can verify the processes and areas of objects that are being manufactured. Another application is the digitization of historical monuments and museums to create virtual tours, which users can access from the Internet. Besides, 3D reconstruction are given in the area of biomedical engineering. Anatomical reconstruction from medical images such as MRI structures has become an important tool in medical diagnosis and therapy planning and surgical procedures. Apart from the above applications, 3D reconstruction has many more applications in different areas. These are some of the many uses of three-dimensional reconstruction and for this reason there is a need to develop this project.

The purpose of this project is to do an algorithm that, based on images, obtain a points cloud of an object. To achieve this aim, in first place; the different techniques developed about 3D reconstruction were studied in order to know the different possibilities. Some of these techniques such as telemetry laser, stereo vision, flight time or structured light which obtain models that are very accurate or not, but with the disadvantage of using expensive equipment in some cases. In second place, perform camera calibration using a calibration method. Finally, get the point cloud object to rebuild.

This work shows that an algorithm can be done to reconstruct an object in three-dimensions, leaving for future developments the optimization for all kinds of objects. In addition it is an important basis for future developments, as many different techniques for image analysis were studied and compared.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In recent years, the algorithms for the reconstruction of real 3D objects have received significant attention not only in Artificial Vision, but also as tools for a variety of applications in medicine, manufacturing, robotics, archeology and other fields that require modeling three-dimensional real environments. So, the main objective of the 3D reconstruction is to obtain a model from an image. Using Artificial Vision for 3D reconstruction is desired to mimic the ability of human beings to see the same object in 3D when shown a picture of the object in 2D. This goal is seen as necessary to get a graphical language for communication between computer and human.

It can be said that the Artificial Vision describes the automatic deduction of the structure and properties of a three-dimensional world possibly dynamic, from one or more two-dimensional images of the world. The images can be monochrome (gray level) or colors, can come from one or more cameras and even each camera can be stationary or mobile. The structures and properties of 3D world don't include only their geometric properties (shape, size), but also their material properties (color, illumination).

An Artificial Vision consists of a number of essential items:

- The first element is the physical system that obtains image or any other system as ultrasonic, thermal, etc; allowing the capture of the model to be studied. Typically, this physical system is a camera that is obtaining images of the scene continuously or when snapshot and send the images to the computer to be studied.
- The second is a system capable of receiving the signal from the previous system which is usually an analog signal and reconstruct it into a digital signal that can be "understood" by the computer. When cameras are used is known as the digitizer card or frame grabber, although they can be captured by cards of electrical, thermal, etc.
- The third is the computer itself that receives digital signals from the respective element hardware and performs the operations for which it is programmed.

Nowadays, everyone is witnesses to the rapid growth of 3D reconstruction in a variety of applications in different areas such as modeling, industrial metrology, inspection and quality control, robotics to 3D printing and rapid prototyping, etc. A common representation of 3D content is through detailed 3D digital surface models (usually in the form of point clouds or 3D surface triangular meshes), rendered with photo texture from real imagery. Ideally, the 3D models must be generated rapidly and accurately by automatic techniques. Responding to this growth are different approaches and technologies for 3D model acquisition, typically classified in the two main categories of "passive" and "active" techniques.

Passive methods or older methods rely on the processing of recorded ambient radiation (usually light reflectance); they include stereovision and image based on modeling, but also shape from silhouettes and shape from shading.

On the other hand, Active methods for 3D surface reconstruction employ radiation, mainly laser or light emitted into the object surface and triangulated with the image optical rays. Among these approaches, most common are 3D laser scanning, single image split scanning and structured light scanning.

1.2 Problem Definition

Artificial Vision is one of the fields that is being studied in particular the Reconstruction. 3D Reconstruction of scenes or objects from images has been applied in various area such as medicine, entertainment, robotics, metrology, history, etc. Without the help of the 3D Reconstruction, designers that create three-dimensional models waste many hours in applications to model and get objects that belong to the real world.

Eventhough 3D Reconstruction is a great help to facilitate real-world objects, this has drawbacks when execution. One problem area is Artificial Vision 3D metric reconstruction of the image acquired by cameras calibration using component or accessible devices and techniques that most used devices are not obtainable or easily available. This problem involves the following: The acquisition of the images, The calibration of the cameras and parameter estimation and Obtaining of the point cloud.

1.3 Project Proposal

For the project, the completion of the reconstruction of a 3D object is proposed through the use of cameras and a projector. In addition, the use of a vision system software (HALCON) which is used throughout the project from the acquisition of images to reconstruct object generation.

For the development of this project, Artificial Vision techniques are used such as image processing, camera calibration algorithms, 3D reconstruction algorithms and triangulation algorithms. Furthermore, their mathematical formulations for each algorithm are applied as 3D transformations and Poses.

The scope for this project is obtain the 3D reconstruction of an object from one of the techniques

of reconstruction of Artificial Vision and implement the optimal algorithms for reconstruction technique chosen in the vision software (HALCON).

1.4 General Objective and Specific Objectives

The general objective is implement an image processing algorithm for the reconstuction of 3D objects from images acquired by a camera and a projector using a vision system software (HALCON).

To achieve the general objective, the following specific objectives were established:

- Testing of selected equipment for the project including hardware and software.
- Selecting the programming language and therefore the software to be used for the project.
- Management programming language software vision system for this project.
- Acquisition of images using the selected software.
- Calibration of the two cameras using a proprietary software grid.
- Selection of technical and development patterns to use.
- Getting the point cloud object implemetandos rebuild using algorithms in the software.
- Evaluation of the technique used for 3D reconstruction.
- Analysis and interpretation of results obtained during calibration and 3D reconstruction.

1.5 Motivation of the Project

Artificial Vision is one of the research topics that currently has a broader spectrum of potential industrial applications, and that in the immediate future still acquire greater importance. At present, the development of new imaging techniques and the evolution of computers, allows to include the third dimension as a real goal.

Estimating the three-dimensional coordinates of an object in a scene is useful in many applications:

- **Quality Control:** In industrial quality control have become very useful three-dimensional technologies, because they can verify the processes and the surfaces of the objects being manufactured.
- **Mobile Robots:** In guiding a mobile robot can take 3D reconstructions and be able to detect, locate and identify objects for easy navigation through the scene.
- **Mapping and surveying:** To prepare dimensional maps and 3D images of land.

- **Medicine:** It is now common that computers and robots are helping doctors with operations that previously could not be performed. Also, for operations, can also be used to study diseases and detection of tumors.
- **3D Modeling:** In creating models or reconstruction of virtual tours in cities, museums, etc.

In the reconstruction in three dimensions, different methods some already mentioned above are used and therefore the aim of this project is to employ those techniques that best suit both effectiveness and speed in the reconstruction of objects. The aspect of speed is important, because in almost all industrial applications it is required that the systems operate in real time, so a reconstruction technique that enables lower processing time is used.

1.6 Descriptive Project Route

This document has the following contents:

- **Chapter 1:** Includes the introduction, it is mentioned on Artificial Vision, also the problem will be defined and what are the reasons why the project is done. Also, objectives and scope are mentioned.
- **Chapter 2:** It covers all the theoretical framework from what a camera with their respective characteristics, lighting, optics, transformations until what is digital image processing.
- **Chapter 3:** A review of the state of the art focused primarily on the acquisition of three-dimensional information and the various methods of alignment of 3D point clouds, with particular emphasis on those based on structured light.
- **Chapter 4:** Programming platform used throughout the project. Some commands and own software requirements are mentioned. Finally, tests were performed.
- **Chapter 5:** All the calibration process for the cameras. Some definitions of the parameters, some calibration techniques and methods are included. Finally, implementation of the calibration of the cameras was performed.
- **Chapter 6:** This last chapter is all about 3D reconstruction from its definition, the mathematical formulations, structured light and final deployment with the algorithms proposed in the vision software.

Chapter 2

Theoretical Framework

2.1 Capturing system

2.1.1 Introduction

The camera use a lens game reconstructs an image on a sensitive element and transmits the acquisition system of the computer. Such transmission may be digital or analog. In other words, the capturing systems are transducers that convert light reflected radiation into electrical signals.

2.1.2 Camera

The camera [1] is the device responsible for transforming light signals appearing on the scene in analog signals capable of being transmitted through a medium. It is divided into two parts, the sensor that captures the properties of the object in the form of light signals and converts it into analog signals, and the optical charge of drafting the appropriate elements of the scene by setting an appropriate focal length.

According to the needs of the system when selecting a camera it must consider the following:

- **Resolution:** number of pixels that make up the captured image.
- **Sensitivity:** minimum light level sensor can capture.
- **Dynamic Range:** margin that can capture light sensor.
- **SNR** influence among pixels.
- **Frame Rate (FPS):** frames per second, measured in frames per second.
- **Shutter:** device that controls the exposure time.
- **Asynchronous Capture :**allows imaging of objects centered at a particular position.
- **Digital I/O :** possible interactions with other system elements.

- **Autoiris:** allows automatic adjustment of the size of the lens opening in response to changes in lighting.

On the other hand, there are different technologies for capturing images; either built around CMOS or CCD sensor technology, with the nature of the project, determining the most appropriate in each case. After taking into account the type of sensor, the choice of shutter should also be noted at this point. There are two options: global or rolling shutter. Once the type of sensor and the shutter is defined, then the next consideration is the frame rate, ie, the number of images that a camera should deliver per second to handle the job perfectly. Although this issue seems complex at first glance, the decisions are actually cutting very clear once the requirements of the image processing system are known.

2.1.2.1 Sensor

Nowadays, [2] there are two types of technologies used to manufacture digital camera sensors. This is the CCD (Charge Coupled Device) or CMOS (Complementary Metal Oxide Semiconductor). Both sensors are formed in essence by metal oxide semiconductor (MOS) and are arranged in matrix form.

Its function is to accumulate an electric charge in each of the cells in this array. These cells are called pixels. The electrical charge stored in each pixel, will depend at all times of the amount of light incident on the pixel. While more light falls on the pixel, the greater the burden that this purchase.

Although in its essence, CCD and CMOS function in a very similar way, there are some differences that distinguish the two technologies.

2.1.2.1.1 CCD Sensor

*In the case of CCD, this makes loads of matrix cells in voltage and delivers an analog output signal, which is then digitized by the camera. In the CCD, a reading of each of the values corresponding to each of the cells is made. Then, this information is an analog-digital converter as data translated. In this case, the internal structure of the sensor is very simple, but have the disadvantage of the need for an additional chip to handle the processing of information provided by the sensor, resulting in a larger spending and bigger equipment. In terms of dynamic range, the CCD sensor is the overall winner because CMOS exceeds a range of two. The dynamic range is the ratio of the saturation of the pixels and the threshold below which no signal capture. In this case the CCD, to be less sensitive, the ends of the light much better tolerated. As for the noise, are also higher than CMOS. This is because the signal processing is performed on an external chip, which can be optimized to perform this function better. In contrast, in the CMOS, the whole process of the signal within the sensor performed, the results will be worse, because there is less room for the photo-diodes that collected light.

The uniform response is the expected outcome of a pixel under the same level of excitement as the others, and that it does not present significant changes in the signal obtained. In this regard, a CMOS sensor which is constituted by individual pixels, making it more prone to failure. In the

CCD, when the entire pixel array uniform, has a better behavior. Nevertheless, the addition of feedback circuits enables us to overcome this problem in the CMOS, CCD are also slightly above.

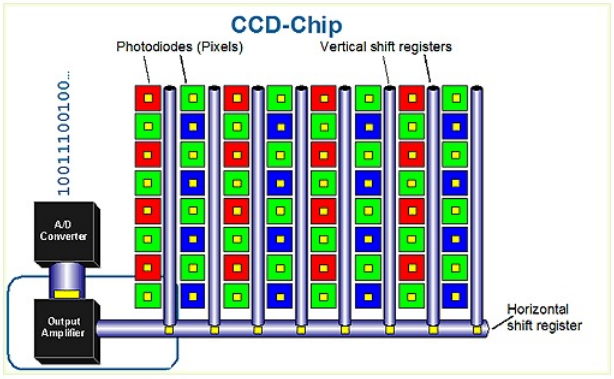


Figure 2.1: CCD Sensor

2.1.2.1.2 CMOS Sensor

In the case of CMOS, here each cell is independent. The main difference is that here the scanning of the pixels is performed internally in transistors that takes each cell, so that all work is carried out within the sensor chip and no external charge of this function is necessary. With this, we incur lower costs and smaller teams.

Besides offering more quality, CMOS are cheaper to manufacture precisely why we mentioned above. Another major advantage is that CMOS sensors are more sensitive to light, so in poor lighting conditions behave much better. This is primarily due to signal amplifiers are found in the cell itself, so there is a lower consumption for equal power. The opposite happened in the CCD.

As for speed, CMOS is clearly superior to CCD because all processing is performed within the sensor itself, providing greater speed.

Another aspect that CMOS sensors are superior to the CCD is blooming. This phenomenon occurs when a pixel is saturated by light incident on it and then begins to saturate those around him. Although this defect can be remedied thanks to some tricks in construction, in the case of CMOS forget the problem.

The nature of the project that will determine the most appropriate in each case. Certainly, CMOS sensors can reduce the cost of machine vision applications, but in turn, has a more limited use.

2.1.2.2 Shutter

The type of shutter [3] may not seem important at first sight when choosing a sensor, but don't be fooled: It is crucial that the plug matches the application. The two options are global and rolling. The shutter protects the sensor within the chamber against incoming light, opening only at the time of exposure. The shutter or selected exposure time gives the correct "dose" of light and determines how long the shutter stays open. The difference between the two variants of the plug is in the way they handle exposure to light.

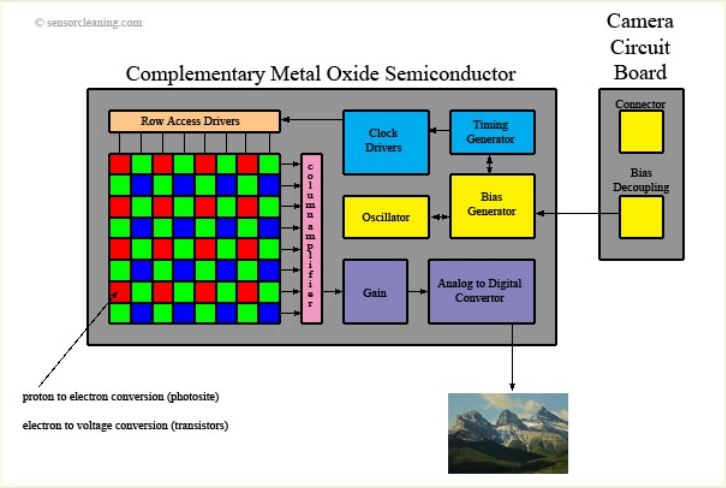


Figure 2.2: CMOS Sensor

2.1.2.2.1 Global

The overall approach shutter opens to allow light to strike the entire surface of the sensor at a time. Depending on the frame rate (typically cited in fps), a moving object is exposed in quick succession. The global shutter is the optimal choice for applications that must capture objects moving very fast, as in the fields of traffic and transportation, logistics and inspection of printed materials.

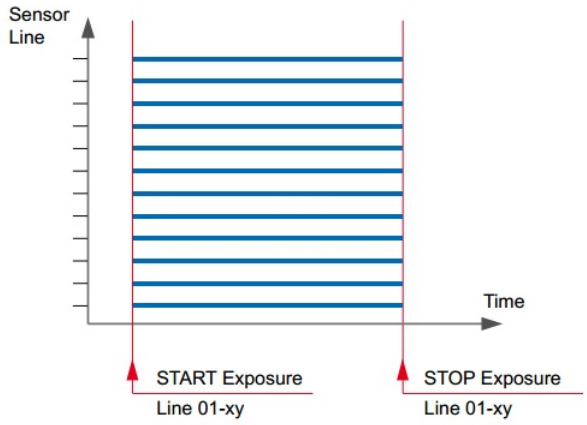


Figure 2.3: Global Shutter

2.1.2.2.2 Rolling

The rolling shutter instead exposes the image line by line. Briefly, each image is composed of horizontal rows. Each line is in turn made of a number of pixels. The type of pixel depends on the resolution. In short: High resolution = more pixels, low = less pixels resolution. Exposure of line by line occurs in stages.

Once the last line in the image has been completely captured, then recording the next shot starts again with the first line. Depending on the exposure time is selected and is the velocity of the

object also, distortions can occur when photographing moving objects during the exposure process. This makes the method unsuitable for some applications. The distortions occur when the captured individual lines are recomposed into a single image and are known as the blind effect.

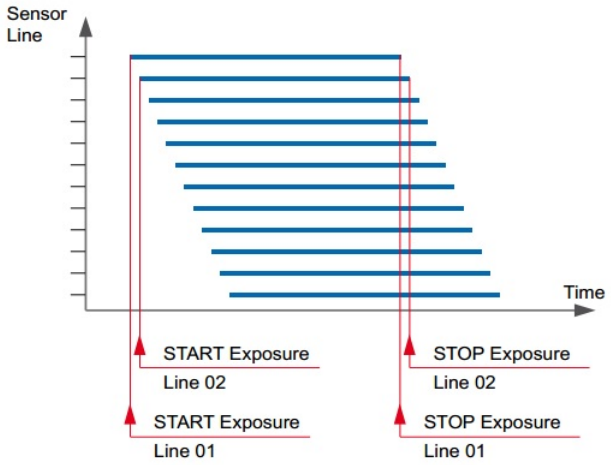


Figure 2.4: Rolling Shutter

CCD sensors always use global shutter, while CMOS sensors offer both model variants. Both sensor technologies have their pros and cons. These should be used to form the basis for a decision. In many cases, the problems related to the rolling shutter effect can be circumvented through proper configuration of the exposure times and the use of an external flash. In other words there's no need to abandon the possibility of a rolling shutter just because the objects are moving.

Global shutter at standard resolution

Rolling shutter at standard resolution

Figure 2.5: Global and Rolling shutter

2.1.2.3 Frame Rate

[3] This term is synonymous with 'frames per second' or 'fps'; For line scan cameras, the terms 'line rate' or 'line frequency' are used. It describes the number of images that the sensor can capture and transmit per second.

The higher the frame rate, the quicker the sensor, meaning more captured images per second and with it higher data volumes. For area scan cameras, these volumes can vary greatly depending on the interface and whether a low rate of 10 fps or a high (fast) speed of 340 fps is being used. Just which frame rates are possible or even necessary depends on what the cameras in the image processing system must record. For fast-moving applications like inspections of printed images, with newspapers moving at high speeds past the camera inspection point, the cameras must be able to 'shoot' in milliseconds. This is a far cry from some microscopic inspections used in medicine and industry, which typically require only low frame rates.

2.1.2.4 Resolution

[3] In practice, resolution describes a measurement of the smallest possible distance between two lines or points such that they can still be perceived as separate from one another within the image.

So, "2048x1088" describes the number of pixels (the dots that form the image) per line, in this case 2048 pixels for the horizontal lines and 1088 pixels in the vertical lines. Multiplied together, the numbers indicate a resolution of 2,228,224 pixels, or 2.2 megapixels (million pixels, or 'MP' for short).

A simple formula (2.1) is used to determine which resolution is required for your application:

R = Resolution

O_Z = Object Size

Z_D = Size of the detail to be inspected

$$R = \frac{O_Z}{Z_D} \tag{2.1}$$

2.1.2.5 Pixel Size

[3] Large surfaces, both on the sensor and the individual pixels themselves, offer more space to capture incoming light. Light is the signal used by the sensor to generate and process the image data. The greater the available surface, the better the Signal-to-Noise Ratio (SNR), especially for large pixels measuring 3.5 m or greater. A higher SNR translates into better image quality. A measurement of 42 dB is considered a solid result.

Another benefit of a large sensor is the larger space onto which more pixels can fit, which produces a higher resolution. The real benefit here is that the individual pixels are still large enough to ensure a good SNR – unlike on smaller sensors, where there is less space available and thus smaller pixels must be used.

2.1.2.6 Sensitivity

[3] Sensitivity refers to the ability of the camera to capture images in dim lighting. For example, imaging of tissues treated with fluorescent agent requires very sensitive cameras. The sensitivity is inversely proportional to the minimum brightness that the camera can capture and usually expressed in units "lux".

2.1.2.7 Dynamic Range

[3] The dynamic range of an image can be basically defined as the ratio of the white area of the image and the darkest area, being in the middle of all the intermediate gray. The most important concept is that dynamic range will depend on the device you have. When possesses dynamic range, more tones will be able to differentiate between what the device can be represented as pure black and pure white. In a camera, dynamic range is set by the maximum amount of light that the sensor can measure before clipping (white dot), and the minimum amount of light that the camera can measure before confused with electronic noise (black dot).

2.1.2.8 Asynchronous capture

[3] This method of capturing images is very important in many machine vision applications where capturing moving objects in front of the field of view of the camera and where should capture the object in a given image position. Asynchronous capture allows images to be captured at a specific time. This concept allows the vertical synchronization of the camera is activated at the right time so that the object is centered in the field of view of the image. Asynchronous capture is activated from an external trigger.

2.1.2.9 Autoiris

[3] The ability to control the iris opening of a camera plays an important role in image quality. The iris is used to maintain the optimum light on the image sensor so that the images can be sharp, clear and with correct exposure and contrast and good resolution. The iris can also be used to control the depth of field.

2.2 Optical

The optics are used to transmit light to the camera sensor in a controlled manner and obtain a focused image of the object. To determine the appropriate lens for the application to be solved, should take into account a number of parameters: sensor size of the camera distance between the camera and the object view or object size. With these data and using a formula, it can calculate the optical to be used in each application, but before performing these calculations it need to know some basics of optics.

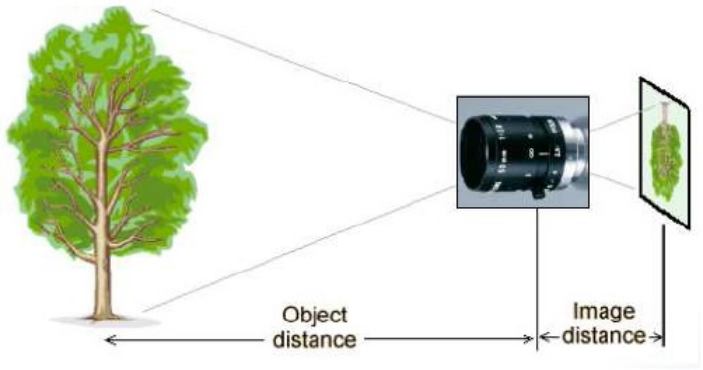


Figure 2.6: Parameter of Optical

2.2.1 Focal Length

The focal length is the degree (mm) of the distance between the lens and the sensor element. The image is inverted because the objective lens, which reverses the image to receive the rays of light. The objectives of the cameras have a variable or fixed focal length, depending on the type of target. By varying the focal length can close in or close out of the object, this is commonly called zoom.

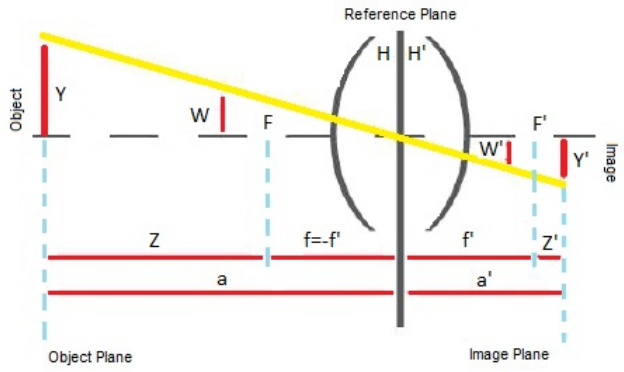


Figure 2.7: Focal Length

Being at the same working distance, with different focal length lenses show the same image in different sizes.

Figure 2.8: Example of focal length

2.2.2 Focus

The optics used in machine vision systems have a variable focal length, also called focus or approach. It is important to analyze the object is perfectly focused for further analysis. The range offers a lens sharpness is between infinity and a distance this distance depends on the focal length.

Figure 2.9: Focus

2.2.3 Depth of Field

Is the distance at which objects appear in focus. A smaller aperture, the greater the depth of field.

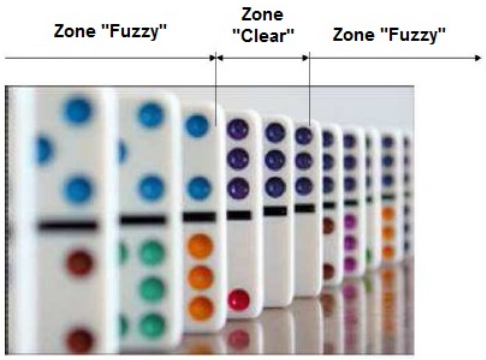


Figure 2.10: Depth of Field

2.2.4 Diaphragm Aperture

The diaphragm is a part of the lens which limits the level of light entering the camera and in turn to the CCD. Works like the human iris, opening or closing allowing more or less light. The position of opening or closing the lens aperture is called.

Figure 2.11: Ring Opening

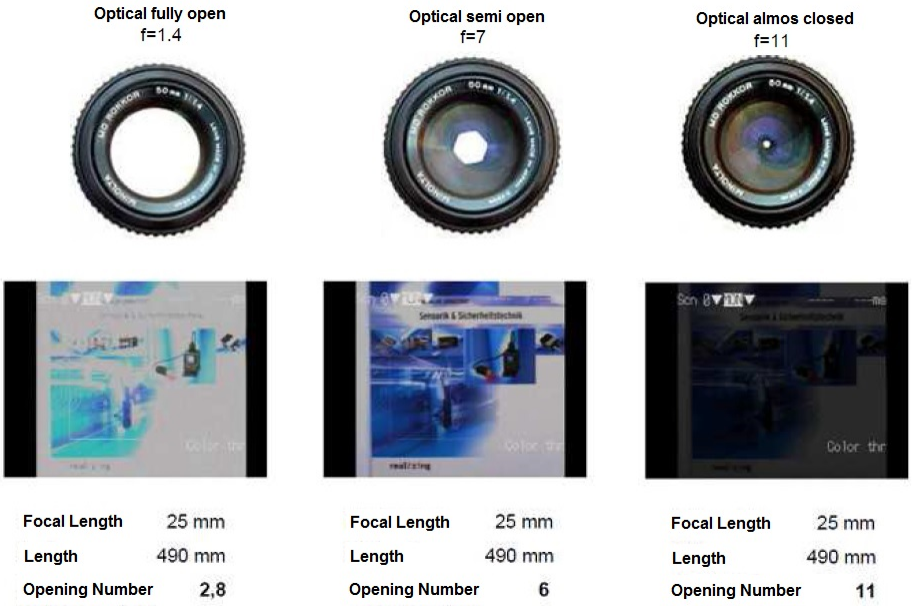


Figure 2.12: Diaphragm Aperture

2.2.5 Extension Tube

An extension tube or a ring placed between the camera and the lens to increase the distance between it and the CCD / CMOS sensor. When inspecting small objects the minimum working distance of the lens can be insufficient, so we can use extension tubes to reduce the working distance. The use of extension tubes also brings some problems such as the optics will not focus to infinity or central image appear brighter than the edges that should be used when they are really needed. In figure 2.13, it can see the relationship between the focal length of the lens and using extension tubes. So a 12mm lens and an extension tube 5mm, it frame an image between 2 and 3 cm.

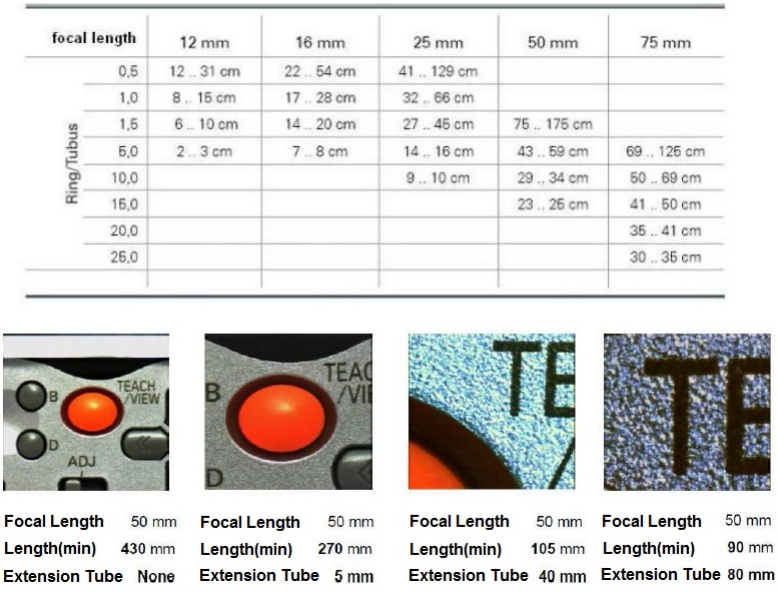


Figure 2.13: Extension Tube

2.2.6 Selection of optical

Once that know all the parts of an object comes time to determine the ideal lens for the application at hand.

Using the equation: Features that are needed for the correct selection of the focal length.

b = CCD sensor size

B = Width of the object

f = Focal length

D = Working distance

c = Conversion factor sensor size

$$f = \frac{b * D}{B} * c \tag{2.2}$$

Example: The width of a CCD sensor 1/3' is 4.8 mm; working distance = 300mm; field of view = 85mm image.

$$f = \frac{b * D}{B} * c \quad \longrightarrow \quad \text{f=}\frac{4.8mm * 300mm}{85mm} * 072 \quad \longrightarrow \quad \text{f} \approx 12mm$$

2.3 Lighting

The lighting [4] [5] in a machine vision system is definitely a very important role in the resolution of the application factor. Consider that the cameras capture light reflected from objects for subsequent analysis. The purpose of lighting is to control how the camera will see the object to determine if it meets the required specifications. Surely if the correct lighting is used in an application, it will be easy to solve. Conversely, if an improper lighting is used, the application may be impossible to solve.

The objectives of lighting are: optimize contrast, normalize any change in ambient lighting and simplify the process of further processing of the image (if filters are used by software processing time is increased).

In the market, it find different options for lighting, here an examples.

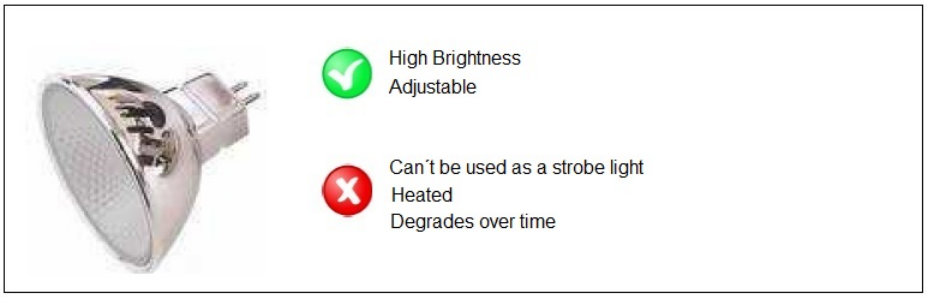


Figure 2.14: Halogen Lighting

Figure 2.15: Incandescent Lighting

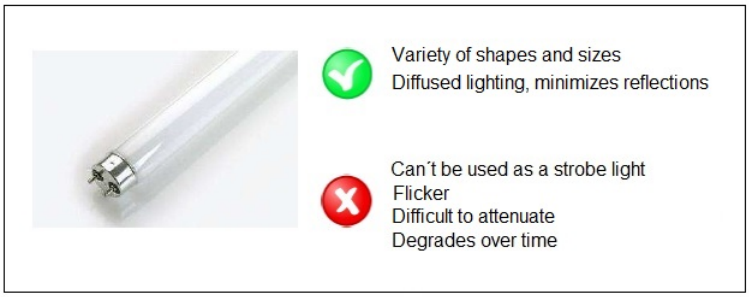


Figure 2.16: Fluorescent Lighting

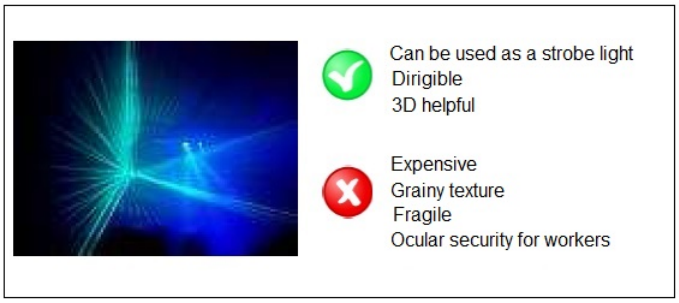


Figure 2.17: Laser Lighting

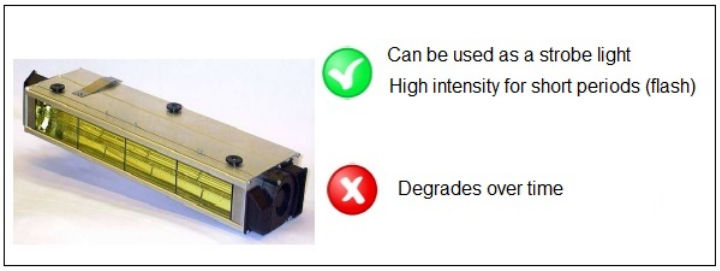


Figure 2.18: Xenon Lighting

Figure 2.19: LED Lighting

There are different lighting techniques to highlight aspects of the test objects. It explain more lighting systems used in computer vision:

2.3.1 Front lighting

The camera is positioned facing the object in the same direction as the light. This reduces shadows, softens textures and minimizes the influence of scratches, dust and imperfections that would have the object. The camera receives the light reflected from the object. This type of lighting is achieved by light rings.

Applications: suitable for surfaces with low reflectivity: paper, cloth, etc. to mark detection different colors, characters and detection of everything that involves a color change on virtually any surface.

Advantages: eliminates shadows, can be used over long distances camera / object.

Disadvantages: intense reflections on reflective surfaces.

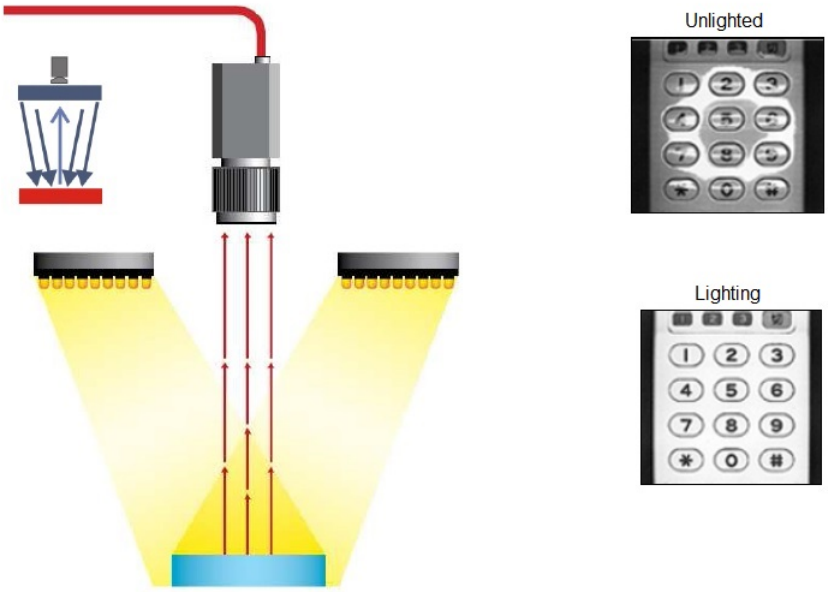


Figure 2.20: Front lighting

2.3.2 Side Lighting

The camera is positioned facing the object while the light direction is lateral to the object. The pitch of the light emitting element is determined by the desired degree projection of the reliefs.

Applications: suitable to highlight edges, scratches and cracks in a certain direction.

Advantages: highlights the reliefs however small objects, resulting in a very defined shadow.

Disadvantages: with small angles to the horizontal, the produced light shades in all reliefs and the workpiece contour.

2.3.3 Dark Field Lighting

The light is emitted laterally with a small angle by a ring in all directions, bouncing off the object to analyze defects and affecting the camera.

Applications: suitable to highlight scale and alphanumeric codes with low contrast in metal on metal or gray on gray. Widely used in the verification of laser engraving or die type.

Advantages: stand out details on surfaces with very little contrast.

Disadvantages: is not recommended for surfaces that absorb light.

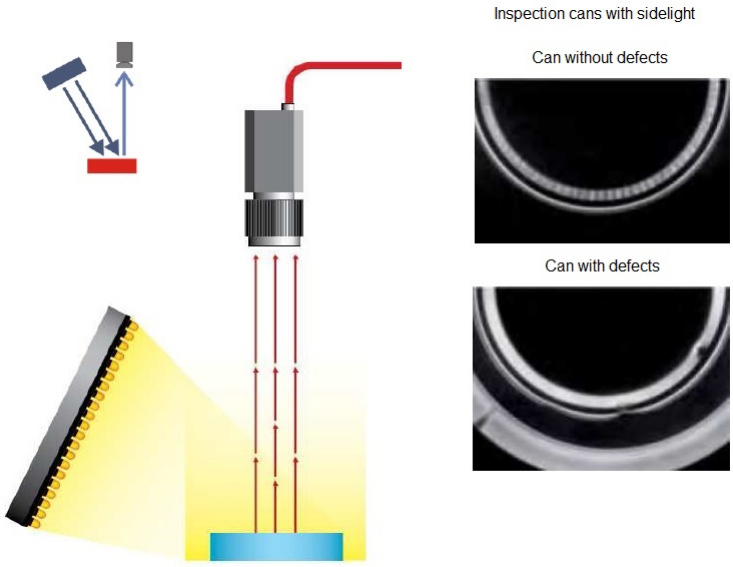


Figure 2.21: Side Lighting

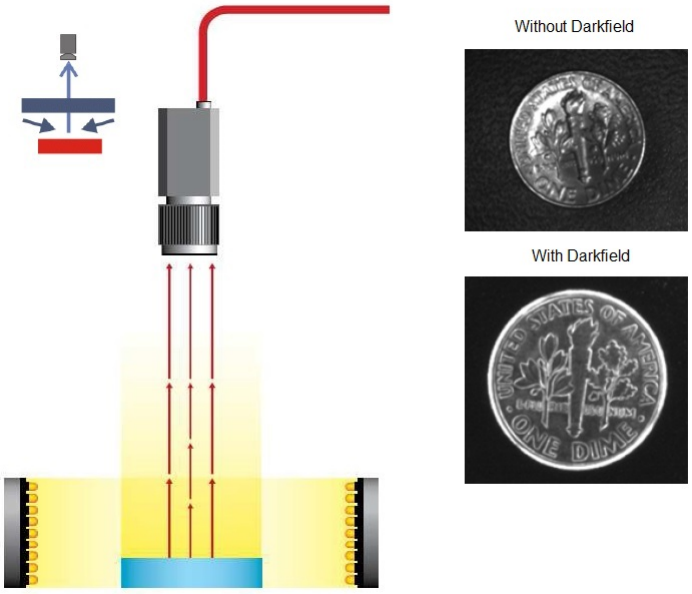


Figure 2.22: Dark Field Lighting

2.3.4 Backlight Lighting

The light is emitted from the back of the object being that between the light source and the camera. The lighting has to be uniform across the surface of the object. The camera inspects the silhouette of the object contrast can make very precise measurements as they completely eliminate shadows from the lighting.

Applications: suitable for the inspection of the silhouette of the object. Also used in translucent or transparent materials to detect spots, stripes, cracks.

Advantages: allows inspections silhouettes with more accurate and impurities in transparent or translucent objects measurements.

Disadvantages: fails to recognize the object surface details, codes, inscriptions, etc.

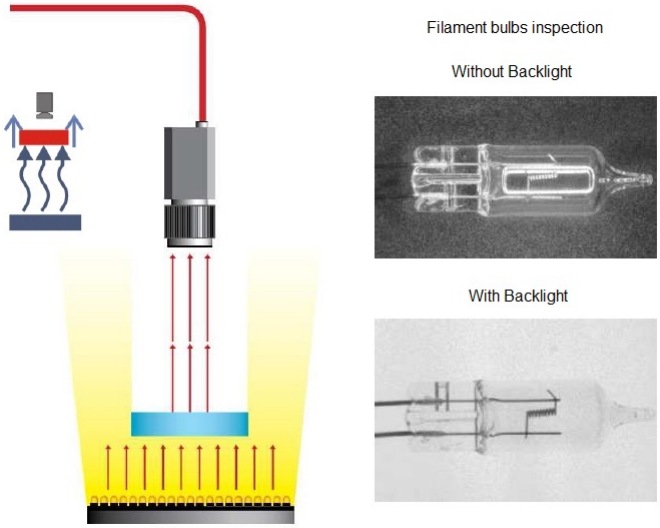


Figure 2.23: Backlight Lighting

2.3.5 Axial diffuse lighting

Light is emitted laterally being reflected by a semitransparent mirror 90 which deflects the light beams in the same direction as the axis of the chamber, obtaining a homogeneous diffuse lighting. In reflective flat surfaces if this method of lighting is not used, the camera would be reflected its own objective.

Applications: suitable for the flat reflective surfaces inspection as PCB, reflective labels, print inspection on aluminum or deep cavities.

Advantages: allows inspection of highly reflective materials codes.

Disadvantages: Do not allow to recognize the object.

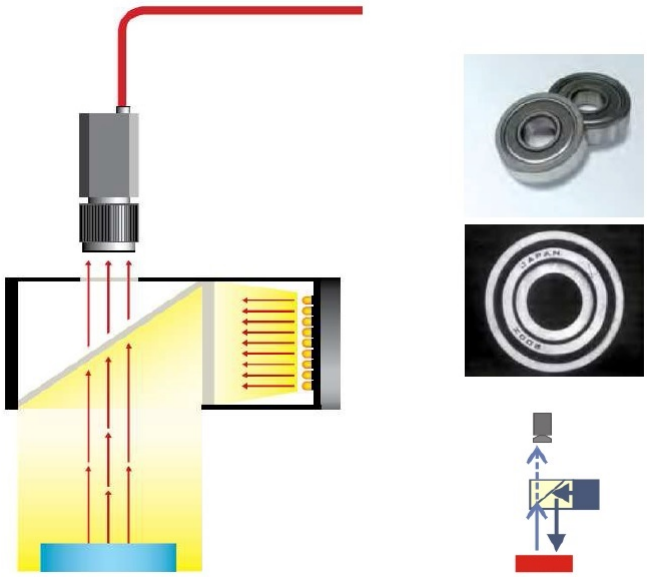


Figure 2.24: Axial diffuse lighting

2.3.6 Diffuse Dome lighting

The light is emitted in a spherical dome resulting diffuse light from all directions, eliminating shadows and reflections, smoothing texture and minimizing the influence of scratches, dust, reliefs and curvatures likely to have the inspected object. This type of lighting it is also called lighting cloudy day not produce any shadow to the object.

Applications: suitable for inspection of surfaces such as medical devices, mirrors, compact disk, cans, etc.

Advantages: eliminating shadows and wrinkle minimizer, dust and reliefs.

Disadvantages: high cost.

2.3.7 Laser Lighting

The illumination or structured laser light is typically used to highlight or identify the third dimension of the object. It is placing the laser light source at an angle to the object known to illuminate and the camera, so seeing the light distortion can be interpreted depth of the objects to be measured. It is also used to indicate the path along which you must set a process, for example in cutting applications. To perform an inspection of a 3D object, a light line is projected. The line distortions are variations in height. From here you can peel a 3D shape sensing the lack or excess of material or get to make three-dimensional reconstruction of the object.

Applications: adjustment processes cutting depth control objects, etc.

Advantages: will not affect the external lighting.

Disadvantages: high cost if cylindrical lenses are used to obtain a line or a specific pattern, the laser does not have the same light intensity along the pattern.

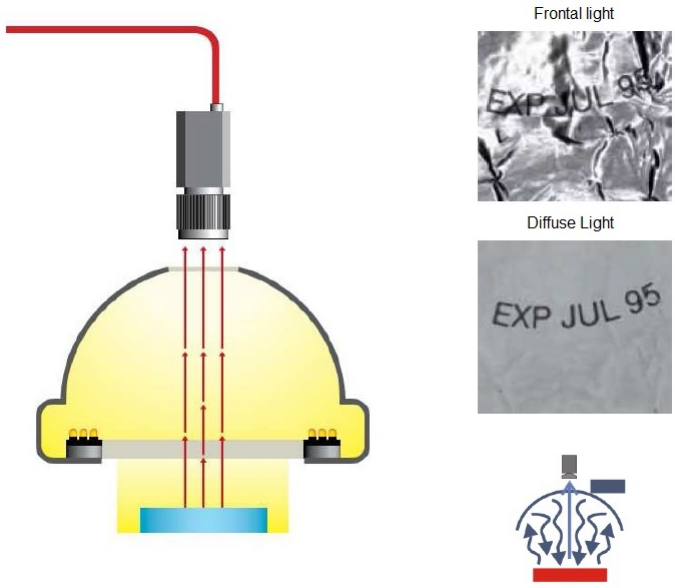


Figure 2.25: Diffuse Dome lighting

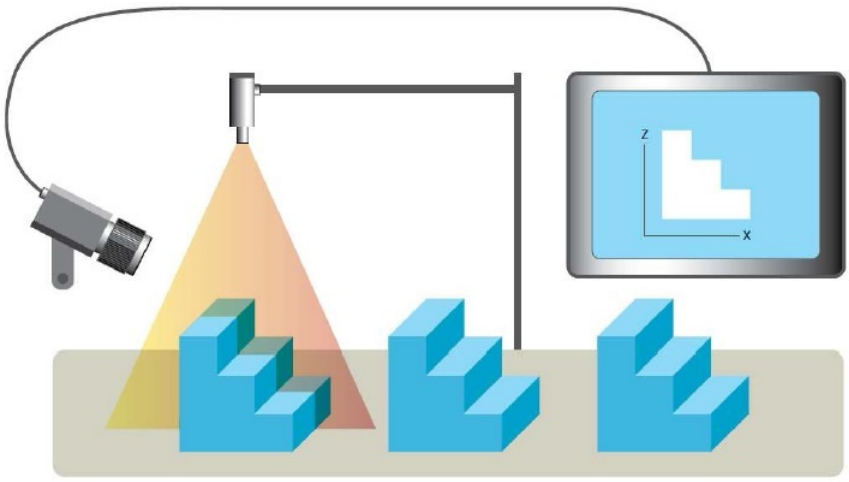


Figure 2.26: Laser Lighting

2.4 3D Transformations and Poses

[6] [7] [8] [9] Before that it start explaining how to perform 3D vision, it take a closer look at some basic questions regarding the use of 3D coordinates:

- How to describe the transformation (translation and rotation) of points and coordinate systems.

- how to describe the position and orientation of one coordinate system relative to another.
- how to determine the coordinates of a point in different coordinate systems, i.e., how to transform coordinates between coordinate systems.

In fact, all these tasks can be solved using one and the same means: homogeneous transformation matrices and their more compact equivalent, 3D poses.

2.4.1 3D Coordinates

[10] The position of a 3D point P is described by its three coordinates (x_p; y_p; z_p). The coordinates can also be interpreted as a 3D vector (indicated by a bold-face lower-case letter). The coordinate system in which the point coordinates are given is indicated to the upper right of a vector or coordinate. For example, the coordinates of the point P in the camera coordinate system (denoted by the letter c) and in the world coordinate system (denoted by the letter w) would be written as:

$$P^c = \begin{pmatrix} x_p^c \\ y_p^c \\ z_p^c \end{pmatrix} \quad P^w = \begin{pmatrix} x_p^w \\ y_p^w \\ z_p^w \end{pmatrix}$$

Figure 2.27 depicts an example point lying in a plane where measurements are to be performed and its coordinates in the camera and world coordinate system, respectively.

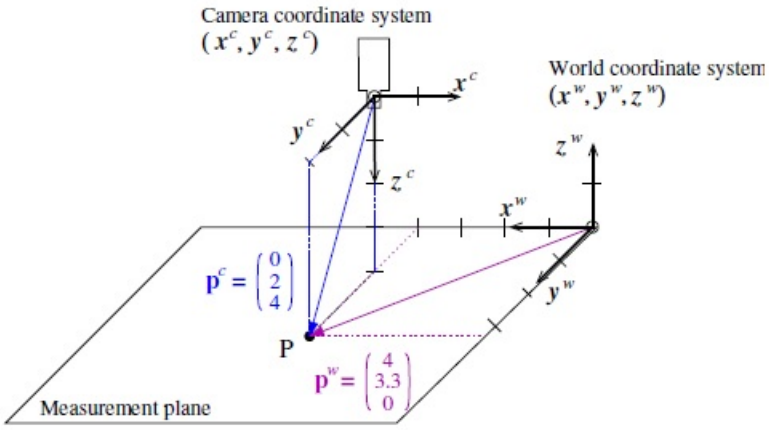


Figure 2.27: Coordinates of a point in two different coordinate systems.

2.4.2 Translation

2.4.2.1 Translation of Points

In figure 2.28, our example point has been translated along the x-axis of the camera coordinate system.

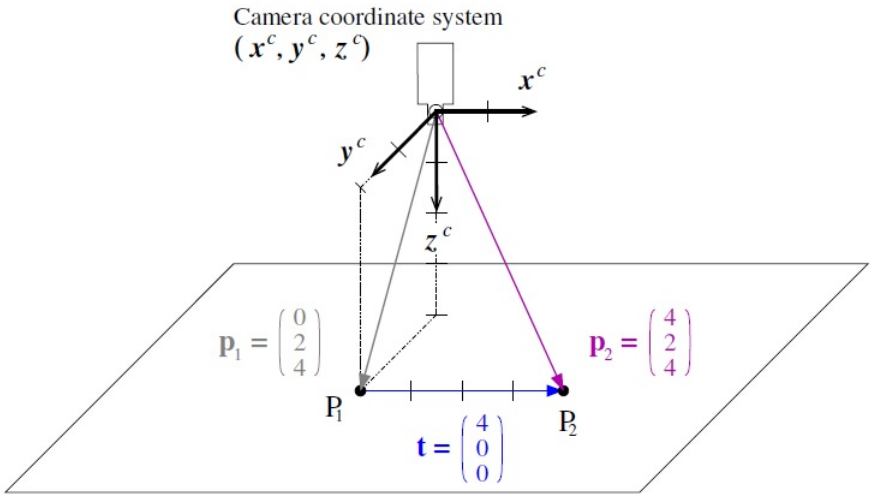


Figure 2.28: Translating a point.

[4] The coordinates of the resulting point P_2 can be calculated by adding two vectors, the coordinate vector P_1 of the point and the translation vector t:

$$P_2 = P_1 + t = \begin{pmatrix} x_{p1} + x_t \\ y_{p1} + y_t \\ z_{p1} + z_t \end{pmatrix} \tag{2.3}$$

Multiple translations are described by adding the translation vectors. This operation is commutative, i.e., the sequence of the translations has no influence on the result.

2.4.2.2 Translation of Coordinate Systems

Coordinate systems can be translated just like points. In the example in figure 2.29, the coordinate system c_1 is translated to form a second coordinate system, c_2. Then, the position of c_2 in c_1, i.e., the coordinate vector of its origin relative to c_1 (O_{c2}^{c1}), is identical to the translation vector:

$$t^{c1} = O_{c2}^{c1} \tag{2.4}$$

2.4.2.3 Coordinate Transformations

Let's turn to the question how to transform point coordinates between (translated) coordinate systems. In fact, the translation of a point can also be thought of as translating it together with its local coordinate system. This is depicted in figure 2.29: The coordinate system c_1, together with the point Q_1, is translated by the vector t, resulting in the coordinate system c_2 and the point Q_2. The points Q_1 and Q_2 then have the same coordinates relative to their local coordinate system, i.e., q_1^{c1}=q_2^{c2}.

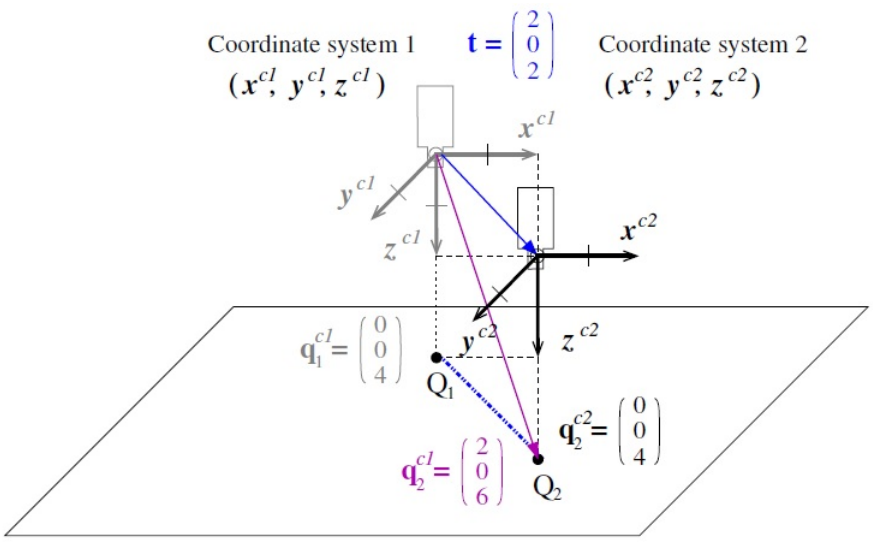


Figure 2.29: Translating a coordinate system (and point).

If coordinate systems are only translated relative to each other, coordinates can be transformed very easily between them by adding the translation vector:

$$q_2^{c1} = q_2^{c2} + t^{c1} = q_2^{c2} + O_{c2}^{c1} \tag{2.5}$$

In fact, figure 2.29 visualizes this equation: q_2^{c1}, i.e., the coordinate vector of Q_2 in the coordinate system c_1, is composed by adding the translation vector t and the coordinate vector of Q_2 in the coordinate system $c_2(q_2^{c2})$.

The downside of this graphical notation is that, at first glance, the direction of the translation vector appears to be contrary to the direction of the coordinate transformation: The vector points from the coordinate system c_1 to c_2, but transforms coordinates from the coordinate system c_2 to c_1. According to this, the coordinates of Q_1 in the coordinate system c_2, i.e., the inverse transformation, can be obtained by subtracting the translation vector from the coordinates of Q_1 in the coordinate system c_1:

$$q_1^{c2} = q_1^{c1} - t^{c1} = q_1^{c1} - O_{c2}^{c1} \tag{2.6}$$

2.4.2.4 Summary

- Points are translated by adding the translation vector to their coordinate vector. Analogously, coordinate systems are translated by adding the translation vector to the position (coordinate vector) of their origin.
- To transform point coordinates from a translated coordinate system c_2 into the original coordinate system c_1, you apply the same transformation to the points that was applied to the coordinate system, i.e., you add the translation vector used to translate the coordinate system c_1 into c_2.

- Multiple translations are described by adding all translation vectors; the sequence of the translations does not affect the result.

2.4.3 Rotation

2.4.3.1 Rotation of Points

In figure 2.30a, the point P_1 is rotated by $-90°$ around the z-axis of the camera coordinate system.

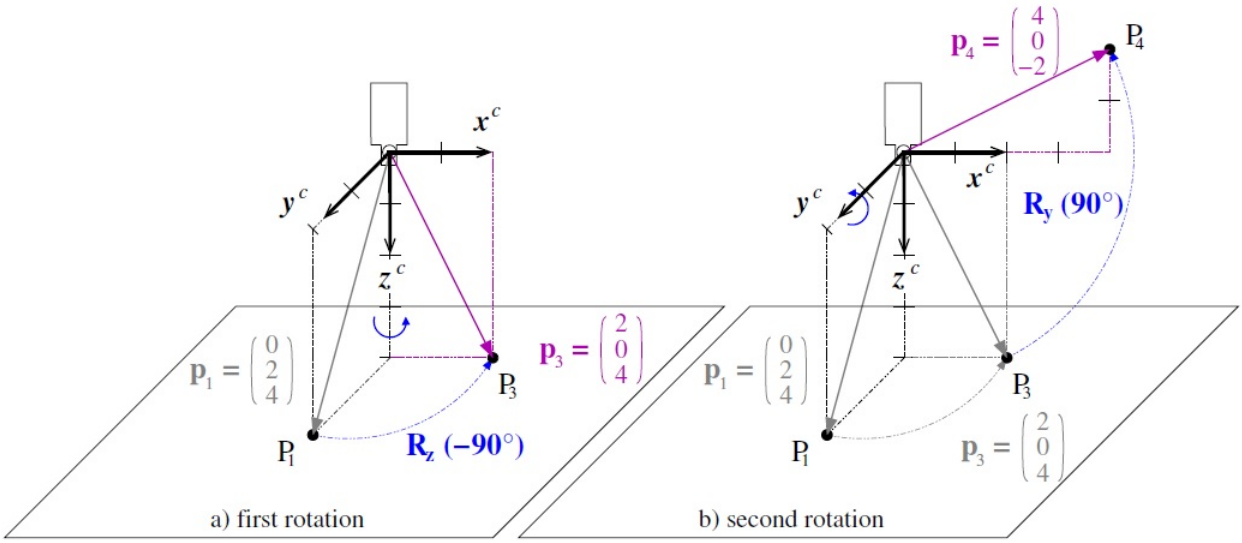


Figure 2.30: Rotate a point: (a) first around the z^c-axis; (b) then around the y^c-axis.

[8] Rotating a point is expressed by multiplying its coordinate vector with a 3x3 rotation matrix R. A rotation around the z-axis looks as follows:

$$P_3 = R_z(\gamma).P_1 = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_{p1} \\ y_{p1} \\ z_{p1} \end{pmatrix} = \begin{pmatrix} cos\gamma.x_{p1} - sin\gamma.y_{p1} \\ sin\gamma.x_{p1} + cos\gamma.y_{p1} \\ z_{p1} \end{pmatrix} \tag{2.7}$$

Rotations around the x- and y-axis correspond to the following rotation matrices:

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{bmatrix} \quad R_z(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \tag{2.8}$$

2.4.3.2 Chain of Rotations

In figure 2.30b, the rotated point is further rotated around the y-axis. Such a chain of rotations can be expressed very elegantly by a chain of rotation matrices:

$$P_4 = R_y(\beta).P_3 = R_y(\beta).R_z(\beta).P_1 \tag{2.9}$$

Note that in contrast to a multiplication of scalars, the multiplication of matrices is not commutative, i.e., if you change the sequence of the rotation matrices, you get a different result.

2.4.3.3 Rotation of Coordinate Systems

In contrast to points, coordinate systems have an orientation relative to other coordinates systems. This orientation changes when the coordinate system is rotated. For example, in figure 2.31a the coordinate system c_3 has been rotated around the y-axis of the coordinate system c_1, resulting in a different orientation of the camera. Note that in order to rotate a coordinate system in your mind's eye, it may help to image the points of the axis vectors being rotated.

Just like the position of a coordinate system can be expressed directly by the translation vector (see equation 2.4), the orientation is contained in the rotation matrix: The columns of the rotation matrix correspond to the axis vectors of the rotated coordinate system in coordinates of the original one:

$$R = \begin{bmatrix} x_{c3}^{c1} & y_{c3}^{c1} & z_{c3}^{c1} \end{bmatrix} \tag{2.10}$$

For example, the axis vectors of the coordinate system c_3 in figure 31a can be determined from the corresponding rotation matrix $R_y(90°)$ as shown in the following equation; you can easily check the result in the figure.

$$R_y(90°) = \begin{bmatrix} cos(90°) & 0 & sin(90°) \\ 0 & 1 & 0 \\ -sin(90°) & 0 & cos(90°) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$x_{c3}^{c1} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \quad y_{c3}^{c1} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad z_{c3}^{c1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

2.4.3.4 Coordinate Transformations

Like in the case of translation, to transform point coordinates from a rotated coordinate system c_3 into the original coordinate system c_1, you apply the same transformation to the points that was applied to the coordinate system c_3, i.e., you multiply the point coordinates with the rotation matrix used to rotate the coordinate system c_1 into c_3:

$$q_3^{c1} = {}^{c1}R_{c3}.q_3^{c3} \tag{2.11}$$

This is depicted in figure 2.31 also for a chain of rotations, which corresponds to the following equation:

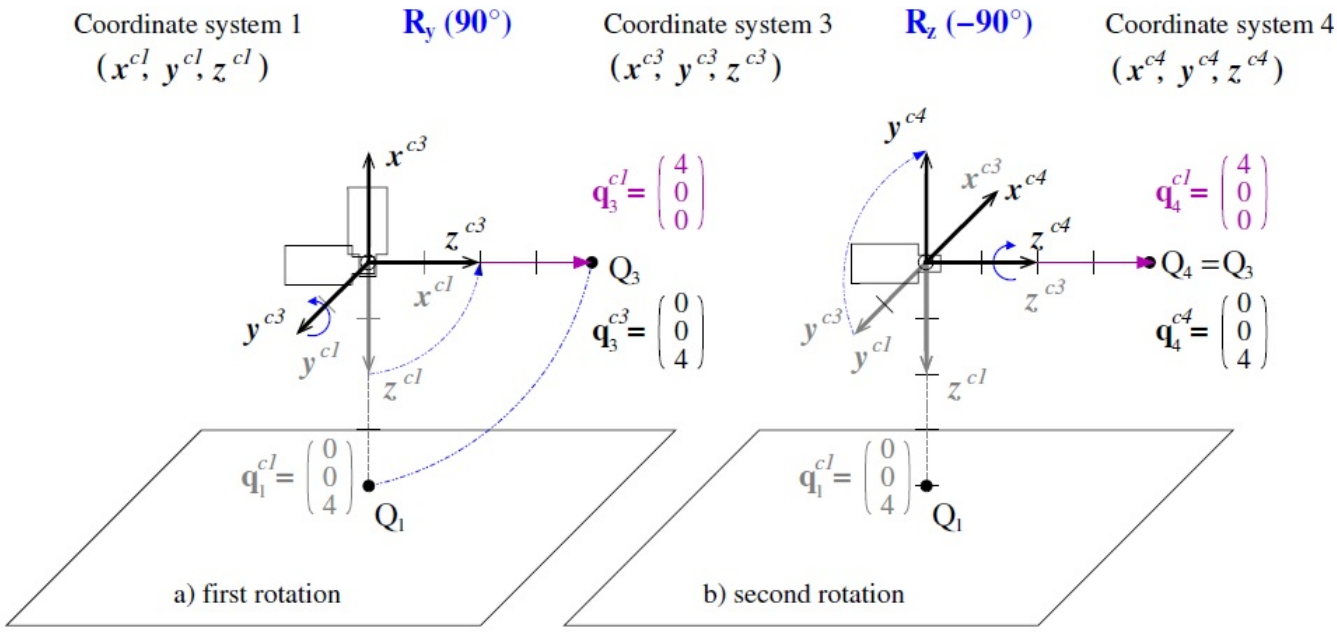


Figure 2.31: Rotate coordinate system: (a) first around the y^{c1}-axis; (b) then around the z^{c3}-axis.

$$q_4^{c1} = {}^{c1}R_{c3}.{}^{c3}R_{c4}.q_4^{c4} = R_y(\beta).R_z(\gamma).q_4^{c4} = {}^{c1}R_{c4}.q_4^{c4} \tag{2.12}$$

2.4.3.5 In Which Sequence and Around Which Axes are Rotations Performed?

If you compare the chains of rotations in figure 2.30 and figure 2.31 and the corresponding equations 2.9 and 2.12, you will note that two different sequences of rotations are described by the same chain of rotation matrices: In figure 2.30, the point was rotated first around the z-axis and then around the y-axis, whereas in figure 2.31 the coordinate system is rotated first around the y-axis and then around the z-axis. Yet, both are described by the chain $R_y(\beta).R_z(\gamma)$

The solution to this seemingly paradox situation is that in the two examples the chain of rotation matrices can be "read" in different directions: In figure 2.30 it is read from the right to left, and in figure 31 from left to the right.

However, there still must be a difference between the two sequences because, as it already mentioned, the multiplication of rotation matrices is not commutative. This difference lies in the second question in the title, i.e., around which axes the rotations are performed.

Let's start with the second rotation of the coordinate system in figure 2.31b. Here, there are two possible sets of axes to rotate around: those of the "old" coordinate system c_1 and those of the already rotated, "new" coordinate system c_3. In the example, the second rotation is performed around the "new" z-axis.

In contrast, when rotating points as in figure 2.30, there is only one set of axes around which to rotate: those of the "old" coordinate system.

From this, we derive the following rules:

- When reading a chain from the left to right, rotations are performed around the "new" axes.
- When reading a chain from the right to left, rotations are performed around the "old" axes.

As already remarked, point rotation chains are always read from right to left. In the case of coordinate systems, you have the choice how to read a rotation chain. In most cases, however, it is more intuitive to read them from left to right.

Figure 2.32 shows that the two reading directions really yield the same result.

2.4.3.6 Summary

- Points are rotated by multiplying their coordinate vector with a rotation matrix.
- If you rotate a coordinate system, the rotation matrix describes its resulting orientation: The column vectors of the matrix correspond to the axis vectors of the rotated coordinate system in coordinates of the original one.
- To transform point coordinates from a rotated coordinate system c_3 into the original coordinate system c_1, you apply the same transformation to the points that was applied to the coordinate system, i.e., you multiply them with the rotation matrix that was used to rotate the coordinate system c_1 into c_3.
- Multiple rotations are described by a chain of rotation matrices, which can be read in two directions. When read from left to right, rotations are performed around the "new" axes; when read from right to left, the rotations are performed around the "old" axes.

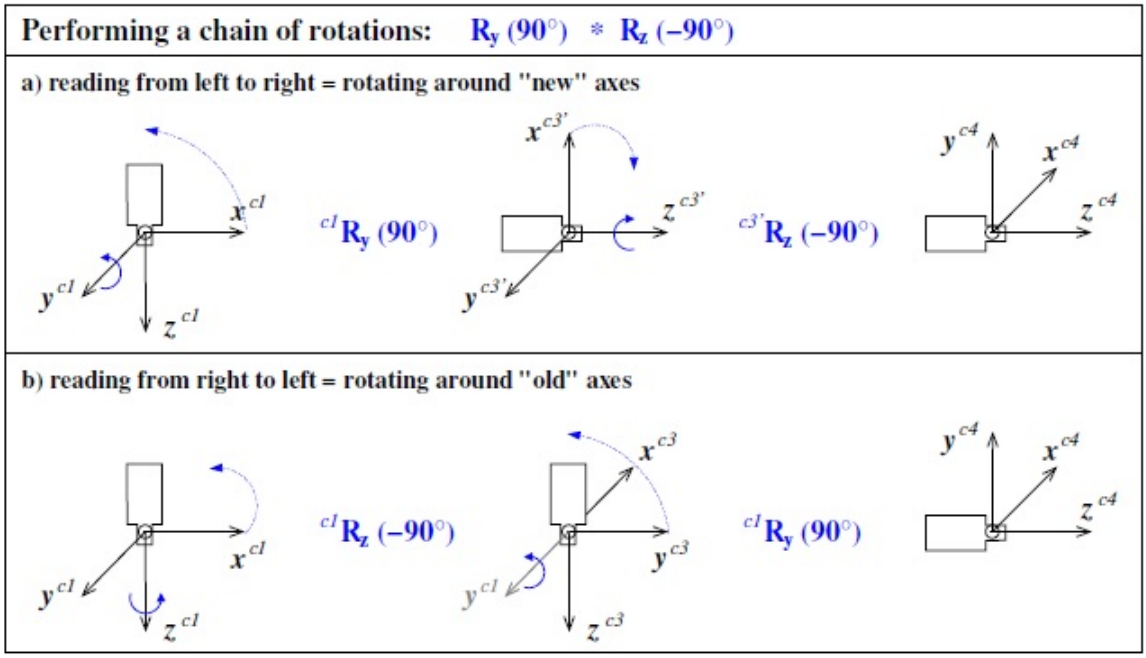


Figure 2.32: Performing a chain of rotations (a) from left to the right, or (b) from right to left.

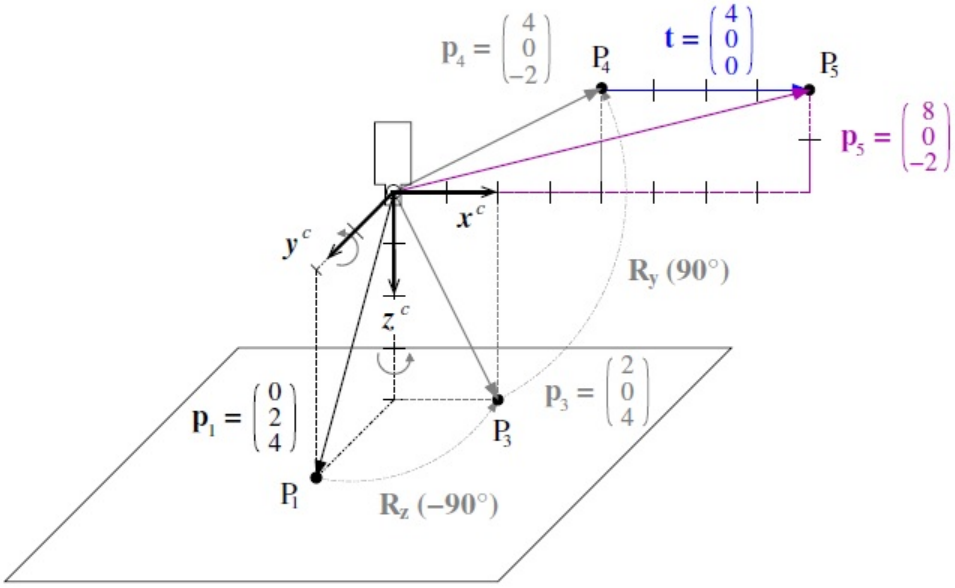


Figure 2.33: Combining the translation from figure 2.28 and the rotation of figure 2.30 to form a rigid transformation.

2.4.4 Rigid Transformations and Homogeneous Transformation Matrices

2.4.4.1 Rigid Transformation of Points

[11] If it combine translation and rotation, it get a so-called rigid transformation. For example, in figure 2.33, the translation and rotation of the point from figures 2.28 and 2.30 are combined. Such a transformation is described as follows:

$$p_5 = R.p_1 + t \tag{2.13}$$

For multiple transformations, such equations quickly become confusing, as the following example with two transformations shows:

$$p_6 = R_a.(R_b.p_1 + t_b) + t_a = R_a.R_b.p_1 + R_a.t_b + t_a \tag{2.14}$$

An elegant alternative is to use so-called homogeneous transformation matrices and the corresponding homogeneous vectors. A homogeneous transformation matrix H contains both the rotation matrix and the translation vector. For example, the rigid transformation from equation 2.13 can be rewritten as follows:

$$\begin{pmatrix} p_5 \\ 1 \end{pmatrix} = \begin{bmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} p_1 \\ 1 \end{pmatrix} = \begin{pmatrix} R.p_1 + t \\ 1 \end{pmatrix} = H.p_1 \tag{2.15}$$

The usefulness of this notation becomes apparent when dealing with sequences of rigid transformations, which can be expressed as chains of homogeneous transformation matrices, similarly to the rotation chains:

$$H_1.H_2 = \begin{bmatrix} & R_a & & t_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & R_b & & t_b \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R_a.R_b & & R_a.t_b + t_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

As explained for chains of rotations, chains of rigid transformation can be read in two directions. When reading from left to right, the transformations are performed around the "new" axes, when read from right to left around the "old" axes.

In fact, a rigid transformation is already a chain, since it consists of a translation and a rotation:

$$H = \begin{bmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & & & 0 \\ & R & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = H(t).H(R) \tag{2.17}$$

If the rotation is composed of multiple rotations around axes as in figure 2.33, the individual rotations can also be written as homogeneous transformation matrices:

$$H = \begin{bmatrix} & R_y(\beta).R_z(\gamma) & & t \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & & & 0 \\ & R_y(\beta) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & & & 0 \\ & R_z(\gamma) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reading this chain from right to left, it can follow the transformation of the point in figure 2.33: First, it is rotated around the z-axis, then around the ("old") y-axis, and finally it is translated.

2.4.4.2 Rigid Transformation of Coordinate Systems

Rigid transformations of coordinate systems work along the same lines as described for a separate translation and rotation. This means that the homogeneous transformation matrix ${}^{c1}H_{c5}$ describes the transformation of the coordinate system c_1 into the coordinate system c_5. At the same time, it describes the position and orientation of coordinate system c_5 relative to coordinate system c_1: Its column vectors contain the coordinates of the axis vectors and the origin.

$${}^{c1}H_{c5} = \begin{bmatrix} x_{c5}^{c1} & y_{c5}^{c}1 & z_{c5}^{c1} & O_{c5}^{c1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

As already noted for rotations, chains of rigid transformations of coordinate systems are typically read from left to right. Thus, the chain above can be read as first translating the coordinate system, then rotating it around its "new" y-axis, and finally rotating it around its "newest" z-axis.

2.4.4.3 Coordinate Transformations

As described for the separate translation and the rotation, to transform point coordinates from a rigidly transformed coordinate system c_5 into the original coordinate system c_1, it apply the same transformation to the points that was applied to the coordinate system c_5, i.e., it multiply the point coordinates with the homogeneous transformation matrix:

$$\begin{pmatrix} P_5^{c1} \\ 1 \end{pmatrix} = {}^{c1}H_{c5} \cdot \begin{pmatrix} P_5^{c5} \\ 1 \end{pmatrix} \tag{2.19}$$

Typically, it leave out the homogeneous vectors if there is no danger of confusion and simply write:

$$P_5^{c1} = {}^{c1}H_{c5}.P_5^{c5} \tag{2.20}$$

2.4.4.4 Summary

- Rigid transformations consist of a rotation and a translation. They are described very elegantly by homogeneous transformation matrices, which contain both the rotation matrix and the translation vector.

- Points are transformed by multiplying their coordinate vector with the homogeneous transformation matrix.

- If it transform a coordinate system, the homogeneous transformation matrix describes the coordinate system's resulting position and orientation: The column vectors of the matrix correspond to the axis vectors and the origin of the coordinate system in coordinates of the original one. Thus, it could say that a homogeneous transformation matrix "is" the position and orientation of a coordinate system.

- To transform point coordinates from a rigidly transformed coordinate system c_5 into the original coordinate system c_1, it apply the same transformation to the points that was applied to the coordinate system, i.e., it multiply them with the homogeneous transformation matrix that was used to transform the coordinate system c_1 into c_5.

- Multiple rigid transformations are described by a chain of transformation matrices, which can be read in two directions. When read from left to the right, rotations are performed around the "new" axes; when read from the right to left, the transformations are performed around the "old" axes.

2.4.5 3D Poses

[12] Homogeneous transformation matrices are a very elegant means of describing transformations, but their content, i.e., the elements of the matrix, are often difficult to read, especially the rotation part. This problem is alleviated by using so-called 3D poses.

A 3D pose is nothing more than an easier-to-understand representation of a rigid transformation: Instead of the 12 elements of the homogeneous transformation matrix, a pose describes the rigid transformation with 6 parameters, 3 for the rotation and 3 for the translation:(TransX; TransY; TransZ; RotX; RotY; RotZ). The main principle behind poses is that even a rotation around an arbitrary axis can always be represented by a sequence of three rotations around the axes of a coordinate system.

2.4.5.1 How to Determine the Pose of a Coordinate System

The previous sections showed how to describe known transformations using translation vectors, rotation matrices, homogeneous transformation matrices, or poses. Sometimes, however, there is another task: How to describe the position and orientation of a coordinate system with a pose.

Figure 2.34 shows how to proceed for a rather simple example. The task is to determine the pose of the world coordinate system from figure 2.27 relative to the camera coordinate system.

In such a case, it recommend to build up the rigid transformation from individual translations and rotations from left to right. Thus, in figure 2.34 the camera coordinate system is first translated such that its origin coincides with that of the world coordinate system. Now, the y-axes of the two coordinate systems coincide; after rotating the (translated) camera coordinate system around its (new) y-axis by 180°, it has the correct orientation.

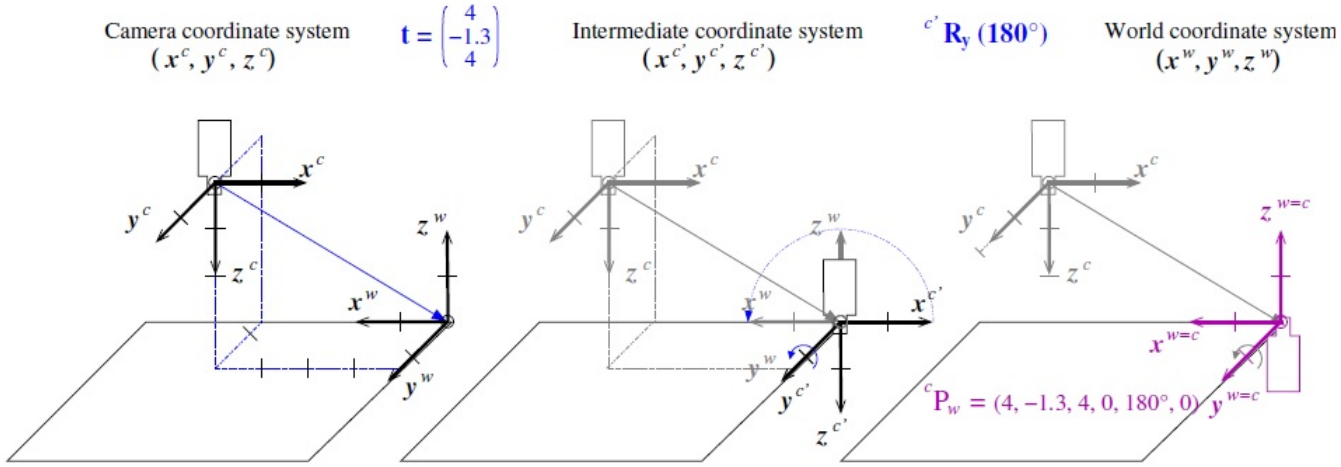


Figure 2.34: Determining the pose of the world coordinate system in camera coordinates.

2.5 Digital Images Processing

[13] The digital image processing is a term used to describe the operations applied to an image, to modify them in some way or obtain objective information of the scene captured by a camera, or other capture device.

In recent years, digital image processing has been widely used for various disciplines such as Medicine, Biology, Physics and Engineering.

As typical applications can include: detecting the presence of objects, automatic visual inspection, measurement of geometric features and color, object classification, image restoration, and improvement of the quality of the images.

2.5.1 Techniques of Digital Image Processing

2.5.1.1 Filters

Filters [13] are used for modifying images either to detect the edges of a scene or to modify the appearance, another function of the filters is, for removing image noise.

2.5.1.2 Histogram Equalization

The histogram [11] is a graph that relates the intensity levels of an image and the number of pixels that have that level of intensity. Histogram equalization is applied when it want to achieve a more uniform between the number of pixels based on the different levels of intensity in the image distribution, that is, a histogram extends in the range of gray levels from zero at maximum gray. The extension of the values of gray levels is performed using information present in the image. The result of the equalization maximizes the contrast of an image without losing structural information, ie, retaining its entropy (information).

2.5.1.3 Hough Transform

The Hough Transform [14] is an algorithm used in pattern recognition in images to find certain forms within an image, such as lines, circles, etc. The simplest version is to find lines. Its mode of operation is mostly statistical and is that for each point it want to find out if it is part of a line. To do an operation within a certain range is applied, so that any lines that may be part of the point are ascertained. This is continued for all the points in the image, in the end determines which lines were the most possible points had and those are the lines in the image.

2.5.1.4 Segmentation

Segmentation[15] subdivides an image into its constituent parts or objects. The level at which this subdivision takes place depends on the problem to be solved. That is, the segmentation should stop when the objects of interest in an application have been isolated.

In general the autonomous segmentation is one of the most dificilles image processing tasks. This process step determines the eventual success or failure of analysis.

2.5.1.5 Binarization

The binarization [16] of a digital image is to convert the digital image into an image in black and white, so that the essential properties of the image are preserved.

One method to a digital image is binarized by the image histogram. Through histogram obtain a graph where the number of pixels shown by each gray level in the image displayed. To binarize the image, it must choose an appropriate value within the gray levels (threshold), so that the histogram forms a valley at that level. All levels below the calculated threshold gray will become black and all major blank.

2.5.1.6 Grayscale

Grayscale [17] uses tints of black to represent objects. The grayscale objects have a brightness value ranging from 0% (white) and 100% (black). Images produced using scanner black and white or grayscale are typically displayed in grayscale. Grayscale also lets it convert color illustrations in illustrations in high quality black and white. The gray levels (shades) of the converted objects represent the luminosity of the original. When it convert grayscale objects to RGB, is assigned to the color values of each object the gray values of the original objects. It can also convert grayscale objects in CMYK objects.

2.5.1.7 Control of the brightness of an image

Sometimes, the appearance of an image can be enhanced visually adjusting the brightness of it. This is accomplished by adding or subtracting a constant value to each pixel of the input image.

The effect of such change on the histogram of the image, scroll right (brighter area) in case a constant or conversely value add, it moves to the left (darker area) when subtracted a constant value.

2.5.2 Optical Methods for 3D Acquisition

Range measurements can be performed using different techniques. The method most applied in industry consists in determining the three-dimensional position of points on the measurement surface by means of a probe guided by a mechanical positioning system. Must, moreover, techniques for passive and active vision image acquisition range. In the passive viewing the light illuminating the scene is taken from the environment; however, active techniques in the sensor incorporates an energy source, such as a laser beam. In general, the methods of data acquisition range are usually classified into two broad categories: active and passive [18].

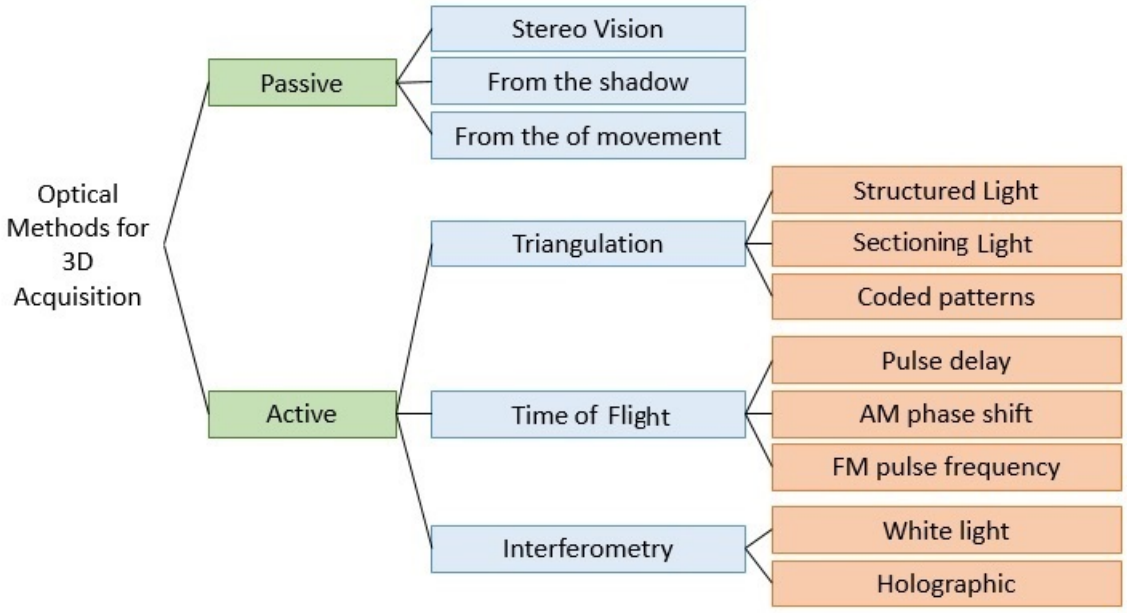


Figure 2.35: Optical Methods for 3D Acquisition

2.5.2.1 Passive Systems

Passive methods have the advantage that don't require additional light source than the ambient light illuminating the scene, so it can be used outdoors. For this reason, a large number of applications using such methods. Although only use cameras and they provide two-dimensional information, it will need to perform some additional processing to obtain depth information. The most common technique is passive stereo vision.

2.5.2.1.1 Stereo Vision

The reconstruction of three-dimensional scenes from images taken from different viewpoints ago was done [19] [8]. Inspection processes which applies stereo vision, usually two more cameras are used, it is possible to use a camera taking views at different relative positions of the camera with respect to the measurement surface.

In general, passive techniques such as stereo vision are relatively inexpensive compared to active, as laser triangulation, but estimates of the depth data of the scenes are much more complex and involve high computational costs. The main problem in stereo vision is to determine the points in the images acquired from different views that correspond to a point in the 3D scene.

After obtaining corresponding points between different views can perform the calculation of the depth from the scene geometry. Between 2000 and 2006 have been presented in some works in which stereo vision is used primarily by the specifics of the inspection task to perform. In [20] Stereo vision system was proposed for the revision of the weights of the lanes station. The acquisition system consists of two video cameras installed under the train, using stereo vision techniques for feature extraction and performing the inspection of weights.

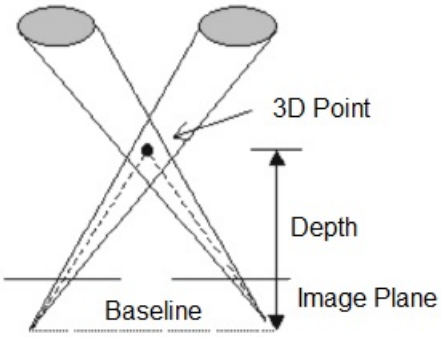


Figure 2.36: Stereo Vision

2.5.2.2 Active Systems

Active techniques using a well defined light source to simplify the correspondence problem. Currently, these active systems are superior in accuracy, cost and ruggedness.

2.5.2.2.1 Triangulation

Active systems of data acquisition range is divided in two main approaches: those who apply the principle of triangulation, and those in the calculation of depth from the time of flight of the signal from the sensor power source [21]. This section describes the systems in which the principle of triangulation is used, while based on flight time are discussed in a later section are discussed. In cameras apply the principle of active triangulation scene is illuminated by a beam of light in one direction and the reflected light is detected by a sensor from another direction. The lighting angle, the viewing angle and the base line between the light source and sensor are triangulation parameters.

2.5.2.2.1.1 Structured Light

In this case the illuminated projecting a pattern of light on the scene, such as points, lines, grids or stripes. Pattern reflected deformations give information of the topology of the surface. In Bertagnolli and Dillmann [22] a system of inspection of rigid components uses an optical sensor mounted on an industrial robot for surface measurements was proposed. The acquisition step is based on the projection principle and recording light bands. The system applies a control scheme based on the comparison of measured data with the manufacturing model. The positioning of the sensor isnt based in robot kinematics but short range photogrammetry using reference marks.

In [23] measuring system for quality control surfaces auto parts based on the technique of structured light reflection, which can detect and measure small undulations and defects in the curvature of reflective freeform surface is presented. The system allows to reconstruct the 3D model using Bezier polynomials and calculating the curve at each point of the surface. Unlike techniques structured light projection, which are sensitive to surface topography, the structured light the reflection is essentially sensitive to gradient and therefore allows detection and measurement of defects that are imperceptible curvature using the techniques projection.

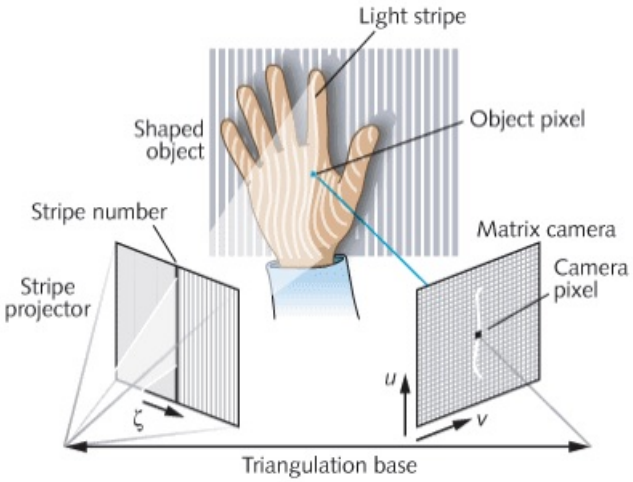


Figure 2.37: Structured Light

2.5.2.2.1.2 Sectioning Light

This is another method using structured light, in this case a plane of light projected on the surface to be examined [24]. A complete view of the 3D surface is constructed sequentially, section by section, moving the object transverse to the plane of light projected. This acquisition method is especially useful when the surfaces on which you are performing the inspection are extended in one dimension, as p. eg. long metal sheets [11] [25]. In [6] a system for inspecting raw steel blocks is presented. The system applies the technique of sectioning light for reconstruction of the surface of the blocks. From data obtained by applying the same technique, the same author in a previous paper [26], used hidden Markov chains coupled to the extraction of surface section lines in noise and classification of surface defects.

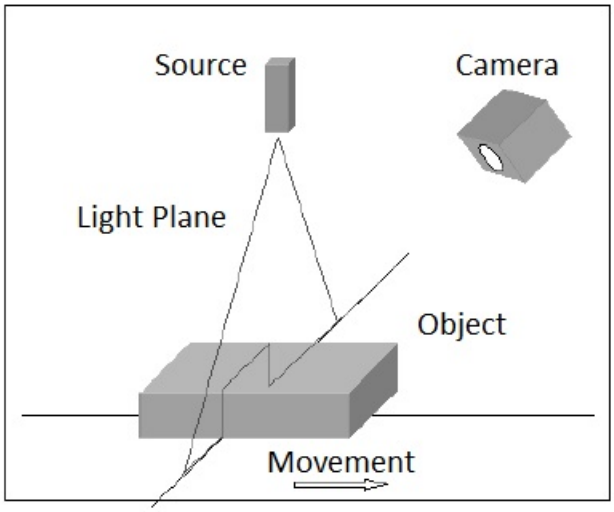


Figure 2.38: Sectioning Light

2.5.2.2.1.3 Coded Patterns

Instead of projecting light plane onto the scene and process multiple images (one for each section), or resolve the ambiguities of a pattern of multiple lines, you can project a pattern of coded light. Because of the pattern are coded, the data range can be calculated from a data table. In [27] classification of different coding strategies structured light used for data acquisition 3D surfaces is presented.

Although in the works cited in the acquisition method followed by coded patterns was only used for reconstruction of 3D scenes, the acquired data could be used in industrial inspection processes. Acquisition system for reconstruction of 3D scenes employing a coded light pattern color was presented in [28] [12]. On the other hand, in [29] a measurement system based on surface were presented coded light. Monochrome cameras with multimedia projectors for acquiring data were combined. In [30] the authors presented a new method coded stripe patterns of colored light to acquire range images in one shot. In this method, edge detection and peak intensity combined.

2.5.2.2.2 Time of Flight

These systems have a signal transmitter, a receiver and a measuring time between emission and reception of the signal. As sound waves signal emitter, especially ultrasound, and light signals, from which the laser light is used. In the latter case we speak of laser radar or laser detectors flight time. Three types of sensors based range Flying time [7]:

1. Pulse delay: in this technique the time of flight of a laser pulse is measured.
2. AM phase shift: in this technique the phase difference between a laser beam emitted by amplitude modulation and the reflected beam is measured, which magnitude is proportional to flying time [31].
3. FM pulse frequency: in this case the frequency shift between the frequency modulated beam emitted and the reflected is measured, another magnitude proportional to flying time.

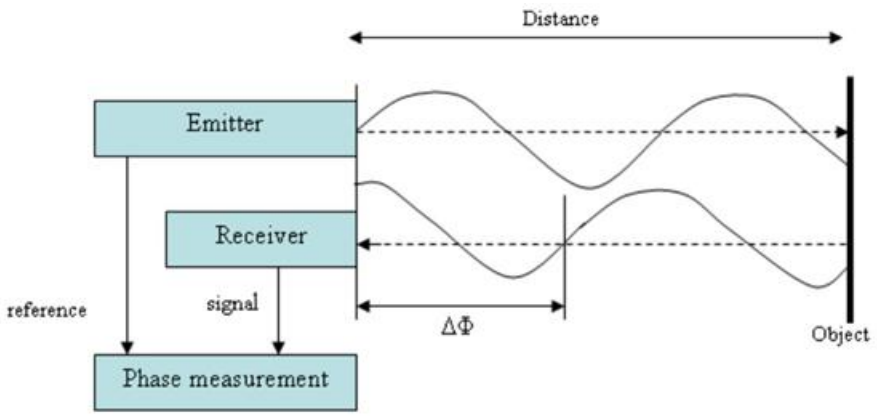


Figure 2.39: Time of Flight

2.5.2.2.3 Interferometry

2.5.2.2.3.1 White Light Interferometry

A light beam is split into two different optical paths: one goes to a reference plane and the other to the surface to be measured, then the reflected rays overlap, forming a pattern of intensity modulated. In [32] analysis of applicability of white light interferometry inspection systems assembly of electronic components are presented.

2.5.2.2.3.2 Holographic Interferometry

In holographic interferometry laser light is used to produce interference patterns due to the phase differences which have different optical paths. From these differences reconstructing the 3D surface of the object becomes. A method for surface reconstruction based on holographic interferometry presented in [33].

Chapter 3

State of the art

3.1 Introduction

The most widely used active methods in the field of vision-based, non-contact 3D measurements are based on structured light systems. Such systems are often composed of a camera and a structured light projector. The measurement principle is similar to the one used in stereo vision systems. However, instead of having a two cameras rig(passive stereo), one camera is replaced by an active light projector. As a result, the extraction of features from the images of the captured scene is simplified, as well as the correspondence process which is a main problem with passive stereo systems.

3.2 Stereo Vision

A multi-view stereo [34] pairs used to capture human faces sixty frames per second is presented. The objective of development is to have a system with the ability to acquire the 180 frontal human face with three pairs of stereo systems, where six cameras are synchronized for the acquisition of paintings by hardware.These requirements are due to the human face has areas that can not be projected to a single plane of 2D acquisition and multiple planes are necessary to make a more complete mapping of the projections of the elements in the real world our image planes 2D.

There are two possible solutions to the problem of acquisition of multiple planes, the first is how the system presented in this work, a moment of acquisition with three synchronized systems generating a plane system simultaneously. The second solution is a single movable flat acquiring system at different times. The choice of a single acquisition system in which each plane is obtained at a different time has the problem that an alignment error introduced since the study subjects will move between each acquisition and the system does not allow the acquisition of human expressions the extended time of purchase.

3.3 Time of Flight

3D Time-of-Flight (TOF) [35] technology is revolutionizing the machine vision industry by providing 3D imaging using a low-cost CMOS pixel array together with an active modulated light source. Compact construction, easy-of-use, together with high accuracy and frame-rate makes TOF cameras an attractive solution for a wide range of applications. In this article, we will cover the basics of TOF operation, and compare TOF with other 2D/3D vision technologies. Then various applications that benefit from TOF sensing, such as gesturing and 3D scanning and printing, are explored. Finally, resources that help readers get started with Texas Instruments' 3D TOF solution are provided.

3.4 Structured Light

The term structured light is defined as the projection of simple or encoded light patterns (i.e. points, lines, grids, complex shapes) onto the illuminated scene [13]. The main benefit of using structured light is that features in the images are better defined. As a result, both the detection and extraction of image features are simplified and more robust.

Concerning the light emitting device, it can be coherent (e.g. laser diodes) or incoherent (e.g. Liquid Crystal Device (LCD) projectors). The use of LCD projectors in active stereo systems can be tracked back to the early 80's [14]. These systems have become very popular, among the research works, thanks to their attractive cost [36] and less security constraints compared with their counterpart laser-based devices. On the other hand, laser projectors for active stereo systems may be the best choice for industrial applications [15]. These systems, often called range scanner, are commonly used in robotics application since the 80's [14]. Unlike LCD-based systems, laser-based systems can be smaller [37] and give a hight power illumination. In addition, thanks to their coherent characteristic, the camera can be equipped with special optical filter to improve both detection and extraction of image features [16].

The projected light patterns can be grouped into three categories [13]: spot patterns, stripe patterns (static or encoded) and colour patterns. The position, orientation and shapes of the light patterns can be either altered or left unchanged during the scanning process.

3.4.1 Spot Patterns

An active stereo system that projects a spot pattern is the simplest approach to measure distances between the rig and points on the surface of the scanned objects (cf. section IV). The correspondence problem is de facto trivial. However, only one measurement can be gathered from a single position of either the projector, the camera or the rig. As a result, all the scene needs to be scanned to collect dense 3D measurements. In order to increase measurement values from a single image, several works propose to project several spots either arranged in line [38] or grid [39], [17] patterns.

3.4.2 Stripe Patterns

The simple geometric pattern of this category, is the projection of a single light ray or plane onto the scene. The intersection of this light pattern with the surface of any object in the scene produces a visible light stripe in the captured images. This technique out-performs the spot-based technique as a larger set of 3D measurements can be gathered from a single image [15]. Again, in order to reduce the number of captured images several light stripes can be projected onto the scene at the same time. Such an active stereo system projects static stripe patterns which can be either arranged in parallel [40], [41], grid [42], [43] or concentric [44] patterns. However, for these systems, the correspondence problem is far from being trivial. In order to tackle this problem, encoded stripe patterns can be used. Indeed, by using the so-called temporal coded-light approach [45] the correspondence can be solved directly as each illuminated point on the surface of any object in the scene is associated with a unique binary code [14]. As a result, each point can be uniquely distinguished from its neighbours. Several methods can be found in the literature to build code sequences. Batlle et al. [14] give an excellent survey concerning this topic.

3.4.3 Color Patterns

A drawback of active stereo systems using encoded stripes patterns is that the image acquisition is sequential depending on the generated code. As a result, only static scenes can be scanned with such systems. To overcome this situation, the use of static stripe patterns or even colour patterns is mandatory. Indeed, colour information can be used to label each light plane of the projected pattern. This technique allows to gather dense 3D measurements from a single image. Such an active stereo system projects coloured stripe patterns which can be either arranged in parallel [46], [47] or grid [48] patterns. Moreover, in order to gain flexibility during the acquisition process adaptive colour techniques can be used [49], [50]. These active stereo systems can adapt the colour of the projected pattern to tackle the problem of light reflections generated by the scanned objects.

3.4.4 De Bruijin Secuences

The methods based on De Bruijn sequences are a group of techniques that define good constuidos patterns in the mathematical sense [51]. Most of them use a one-dimensional sequence of n symbols in which all possible windows of dimension l ($l \ll n$)are unique in the pattern. This idea has been applied directly to encode patterns columns or rows. [52] uses a pattern comprised of horizontal bands with six color components taking three windows, and identifies each strip with its two neighbors. Salvi [53] proposes a pattern composed of a grid of horizontal and vertical lines with six color coded. Each crosspoint is identified taking into account their respective colors and its two adjacent horizontal and vertical lines. Zhang [54] developed a technique that involves projecting fringe patterns five alternating colors using a window of three components. Thus, projected color transitions are mapped to those seen in the image edges. Recently Hall-Holt [55] projected onto the scene four patterns of vertical bars giving each part of the code pattern. This method can handle slow moving scenes despite projecting more than one pattern. Vuylsteke and Oosterlinck [56] use a single binary coded pattern that is implemented by means of black and white squares in two-dimensional windows ($2x3$) separated by two white squares. In fact, encode each point column to recover every $2x3$ window and get your code. This strategy can improve restrictions

based techniques dimensional windows. Certainly it is more compact and less sensitive to noise or depth discontinuities.

3.4.5 Identifying Points

Techniques that identify points rather than lines are also very efficient. In them it is usual to design a symbol map in two-dimensional space with 4/8 connectivity and where each two-dimensional window appears only once. If all possible windows are included in the pattern structure is called perfect map, otherwise called perfect submap. These techniques also differ in the way coding is inserted into the pattern. Research conducted by Griffin [57] on the maximum size of a coded two-dimensional array has made possible the development of several techniques. At one point pattern Griffin himself and his four neighbors is identified. The color code is assigned using the three primary colors. Later, Davies [58] illuminates the scene with a hexagonal pattern of dots of color. However this topology is not used for mapping. Through the adaptation of a three-dimensional Hough transform on image detected points near each epipolar lines and finally solve the correspondence problem. Morano [59] designs a perfect submap using an incremental algorithm where the Hamming distance between windows ($3x3$) is imposed.

3.4.6 Coding of Vertical and Horizontal Lines

There are patterns that encode vertical, horizontal or both lines. Maruyama [60] designs verticale a pattern formed by cuts in the image are determined by its own length and its six projected segments may vary depending on the surface normal, the optical axis and the depth, the reliability of the system is limited. Boyer and Kak [61] must consider the pioneers in structured light projection for dynamic scenes using a single pattern with vertical stripes, three color coded and separated with black stripes while Chen [62] improves its method using two cameras. This allows them to establish a simple correspondence between each pair of strips and calculate the 3D position after the triangulation process.

3.4.7 Coded Binary Pattern

The type of pattern affects the structured light performance including the resolution, speed, and accuracy. Time multiplexing has an advantage in dense 3D reconstruction; however, it cannot reconstruct moving targets [53]. Coded color patterns have a disadvantage if the target has a similar color to the colored pattern. If the pattern consists of a specific shape, then it may also be affected by the objects and background color. However, coded patterns can reconstruct a moving target in 3D; even if it is represented using binary symbols, it is independent of the color, shape, and other details from the background and objects. Coded patterns have a disadvantage in pattern decoding. Therefore, pattern coding methods, such as the pseudorandom, M-array, perfect map, and De Bruijn methods, have been investigated for a long time in order to facilitate the pattern decoding. These pattern encoding methods create symbols for uniqueness within a specific range. The projected pattern is compared with the imaged pattern for recognition. If the pattern is repeated within the search range, matching errors occur. Despite these methods, the light source remains visible and it is difficult to subtract the pattern from the image [63].

This research used a coded binary pattern that was projected using an infrared laser and DOE. There was a small object shape and the pattern is clearly represented. The infrared source was also affected by the quality of the material, but it was not visible. Figure 1 presents the image captured using the sensitive camera in infrared light. Clear patterns and shadows can be seen in the image. The background effect is eliminated and the object is recognized using the shape of the shadow. There is also a material effect: because the material of the cylindrical object has a lower level of light reflection, the projected pattern is faint and, in some areas, invisible. In this case, the pattern does not appear despite the visible light.

In order to reconstruct the 3D image, the surface is triangulated using the difference between the projected pattern and the imaged pattern. The difference is represented using the shift of correspondence points on the epipolar constraints. In this research, the correspondence points are dots that have symbols. The symbols consist of white dots and black dots. The white dots are recognized easily, but it is difficult to distinguish them from the background. Therefore, the black dot recognition uses the results of the white symbols.

3.4.8 Disorderly Pattern

The pattern [64] is a perfect submap connectivity where a sale w is defined as a set of seven symbols that satisfy the following properties: w appears only once in the pattern, w may contain repeated symbols and order in w is irrelevant.

Is M a set of m points in a two dimensional space. To design a M map symbol , seven colors (red, green, blue, magenta, cyan, yellow and white) is used to associate a color to each M point. From now on, is denoted by C the color that has been associated to c. The color has been introduced in M applying an iterative algorithm that starts with a random assignment. In each iteration, the repeated codes are detected and randomly changes the color of the central node. The convergence of the algorithm is achieved since repeated colors allowed in disordered codes and no restrictions on certain Hamming distances. Thus, convergence is easily achieved when m is far less than N.

Chapter 4

Programming Platforms

4.1 HALCON

4.1.1 Introduction

This chapter provides an introduction to the HALCON image acquisition interface and the underlying concepts. It is intended for users who are not familiar with topics like frame grabber hardware, A/Dconversion, synchronous or asynchronous mode of operation, buffering strategies, and the like. Although this manual is not intended to supply you with detailed knowledge about your image acquisition device's internals, we still want to give explanations of the basic terms and methods. Reading the manuals supplied with your image acquisition device is a necessity, of course, and possibly gives you a much more detailed view on the things being discussed here.

If someone are using a Linux system he have to consider the corresponding Linux syntax.

4.1.2 HALCON's Generic Image Acquisition Interface

HALCON provides a generic image acquisition interface that allows free integration of new frame grabbers or cameras on the fly, that is even without restarting a HALCON application.

The two basic concepts used are:

1. Encapsulation of the interface code in dynamically loadable modules.
2. A set of predefined HALCON operators for image acquisition, including operators for setting and retrieving specific hardware parameters. The latter allow the parameterization of even the most "exotic" acquisition devices.

If you have successfully developed a new HALCON image acquisition interface, then all you have to do to use your new image acquisition device is:

- Plug in the hardware and install the vendor-specific device driver and libraries.

- Copy the new HALCON interface (i.e., the loadable module with the encapsulated hardwaredependent code) to a directory within your search path for DLLs or shared libraries, respectively.
- Specify the name of the new image acquisition device (i.e., the name of the corresponding interface) in the *open_framegrabber* operator.
- Enjoy the performance of all the features you have integrated in your new image acquisition interface.

The HALCON operators used for image acquisition remain the same, so existing application code can be used without modification in most cases. HALCON automatically loads the interface during the first call to info_framegrabber or open_framegrabber. Thus, you can exchange/add image acquisition interfaces even without restarting your application. Special features of different image acquisition devices can be accessed through the general purpose parameter setting mechanism. Since digital cameras which are connected by USB, IEEE 1394 or GigE are not really based on an actual frame grabber board, we no longer use the term HALCON frame grabber interface.

Instead, it use the term HALCON acquisition interface, and the term image acquisition device is used as a substitute for either a frame grabber board or a digital camera. For backwards compatibility reasons, the names of the HALCON operators and also the underlying names for the data structures and error codes have been unchanged. Thus, operator names like open_framegrabber, info_framegrabber, and close_framegrabber may sound a little bit old-fashioned.

4.1.3 Image Acquisition Basics

Note that the following sections in this chapter mainly describe the case that an analog camera is connected to a frame grabber board. However, most of the principles are also important in case of digital capture devices like IEEE 1394, USB, or GigE cameras.

Basically, what a frame grabber does is to take a video signal, which can be understood as a continuous stream of video frames, and grab one or more video frames out of the sequence, whenever triggered to do so. In many cases, the video signal will be an analog one, although more professional equipment often uses digital signals nowadays. The most common analog video formats are:

- NTSC: 640 x 480 pixel, 30 frames per second.
- PAL: 768 x 576 pixel, 25 frames per second.

Both formats carry color information, although many image acquisition devices only deliver grayscale images, even from a color video signal. The following explanations assume that you are using an analog frame grabber board. With digital boards, things may be different. Let us take a look at the analog input signal: Actually, it is composed of many different signals: There are vertical and horizontal sync signals and, of course, the raw data signals as well. Sometimes, the color and brightness signals are overlaid (composite signal), sometimes they are delivered on separated input lines (Y/C, RGB). Since the frame grabber is usually synchronized by the video source, it has to wait for the next vertical sync signal to start grabbing a new image, see the next figure:

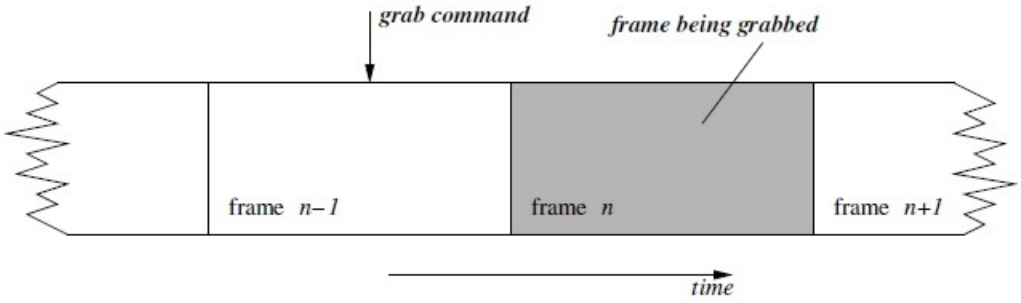


Figure 4.1: Grabbing one frame

This will cause a delay of half a frame on average when grabbing an image of random frames4. It also implies that you have to start grabbing the next frame immediately5 after receiving the previous frame if you want to achieve full frame rate. Consequently, there would be no time at all left to process images. In this synchronous mode, the host computer is exclusively busy triggering one grab after another. Therefore, HALCON also supports asynchronous grabbing as will explained.

4.1.4 Synchronous vs. Asynchronous Grabbing

To understand what asynchronous grabbing means, we first should take a look at what the image acquisition device does with a grabbed frame. It is easily understood, that a digitized frame must be stored in some kind of memory. Basically, there are three possibilities:

- Device memory on the image acquisition device
- Device memory on the host machine
- Host memory

Device memory on the board means dedicated memory, physically mounted to the board. This way, the image acquisition device can store the acquired image(s) directly in its own memory, with each process on the host being able to get the data at any time.

On the other hand, memory size is fixed. If it is too small, it may not be possible to keep several images in memory. If it is very big, the whole board can get rather expensive. Device memory on the host machine is non-paged system memory dynamically allocated by the frame grabber's device driver. Thus, the memory size can be easily adjusted. On the other hand, heavy bus traffic is likely to occur, if the image acquisition device is delivering data to the host computer's memory permanently.

Therefore, it usually do not use continuous grabbing modes provided by some image acquisition devices, but grab images only on demand. Host memory is allocated by the user somewhere in the address space of the application. Since this memory might use paging, the images delivered from the image acquisition device in general must be explicitly copied to this memory.

The host computer's job, as mentioned above, is to trigger the image acquisition device when a new image is needed, but it does not necessarily need to wait while the device digitizes the frame.

With device memory being on the board, this is self-evident, but also if the target memory is host-based, externally initiated data transfer is usually possible with techniques like DMA. So the "only thing" the host process has to do is to trigger the image acquisition device an average time of 1.5 frames before an image is actually needed, and then it can do some other processing while the new frame is captured by the board in the background.

This technique is called asynchronous grabbing. It is easily understood that this eases real-time grabbing, since the time needed for frame completion is rather long (40 ms with PAL, 33 ms with NTSC video) compared to the small time gap between two adjacent frames in the video stream.

Most image acquisition devices support asynchronous data transfer. Therefore, HALCON provides both synchronous (grab_image) and asynchronous grabbing (grab_image_async). The reason for supporting the somehow less powerful synchronous mode is the clearer semantics: The operator grab_image starts a grab and waits until it is finished.

Thus, the delivered image is per definition up to date. Using asynchronous grabbing needs a little bit more insight in the timing of the application. The grabbed images might be too old to be used otherwise. Now let us take a look at some memory management strategies useful for efficient image acquisition.

4.1.5 Examples of how to use the Halcon

4.1.5.1 Applying threshold changes to an image

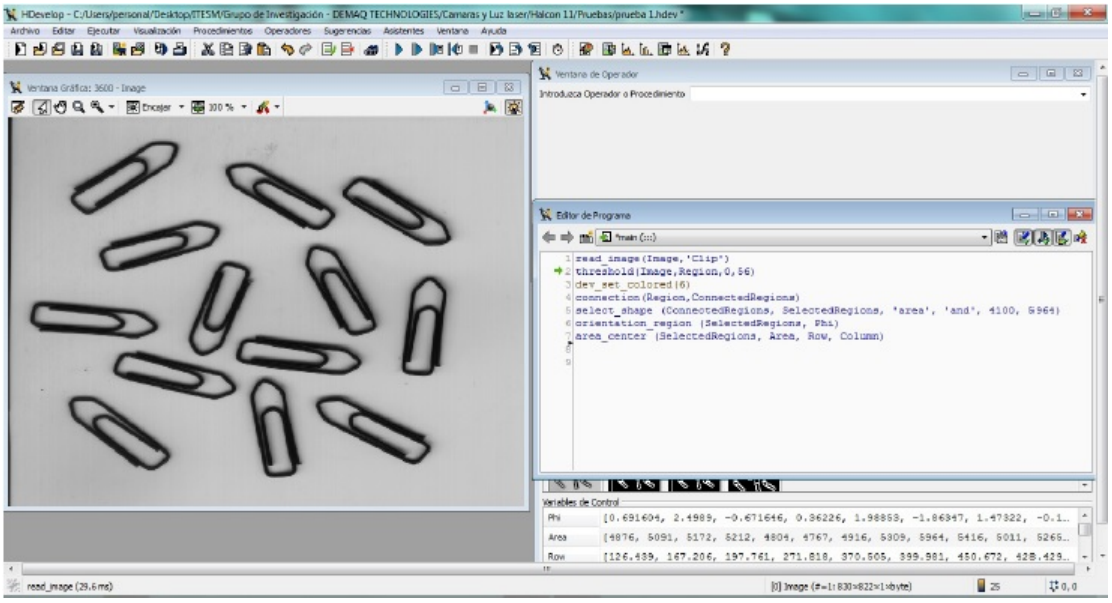


Figure 4.2: Read image

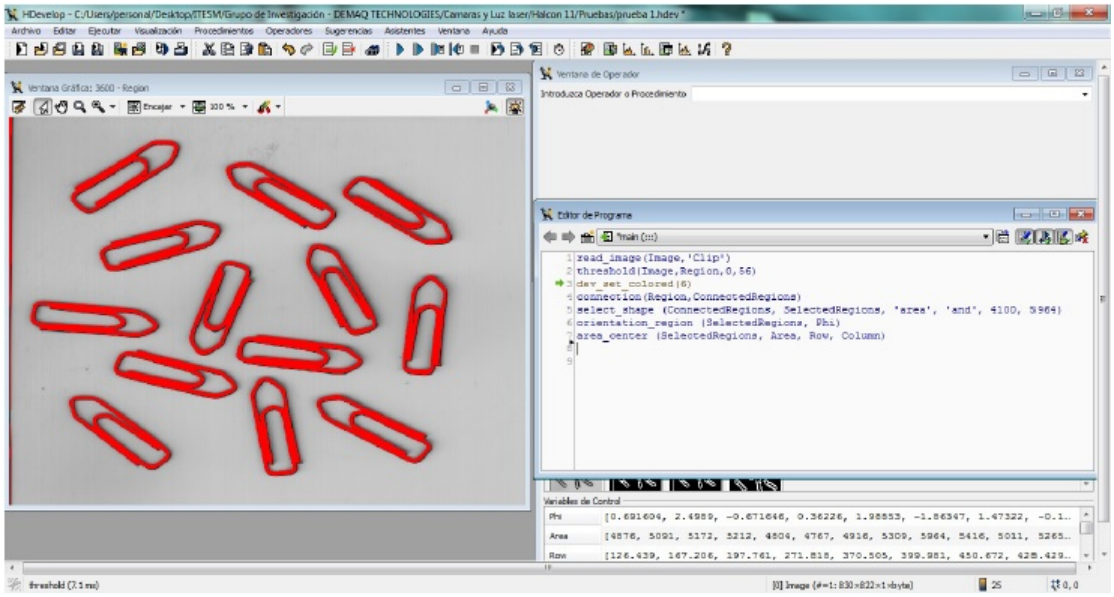


Figure 4.3: Applying threshold

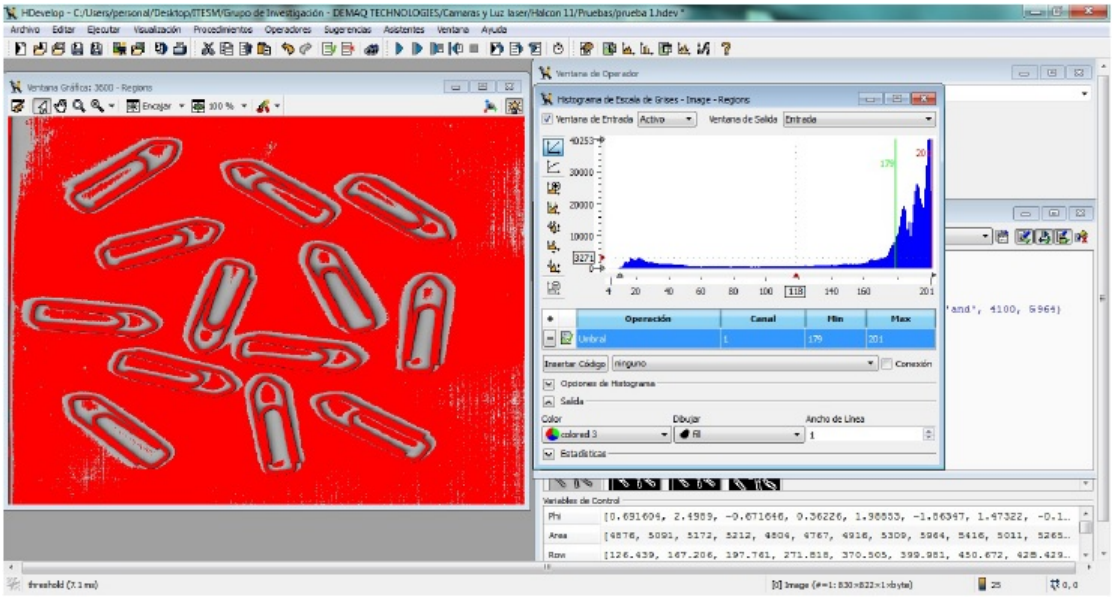


Figure 4.4: Applying grayscale histogram using region 1

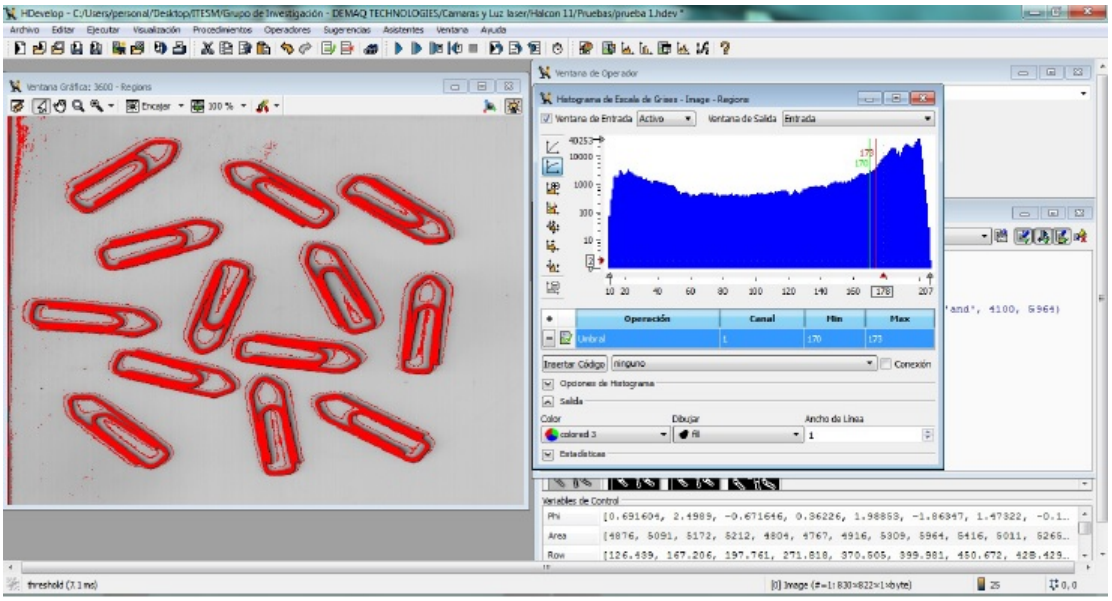


Figure 4.5: Applying grayscale histogram using region 2

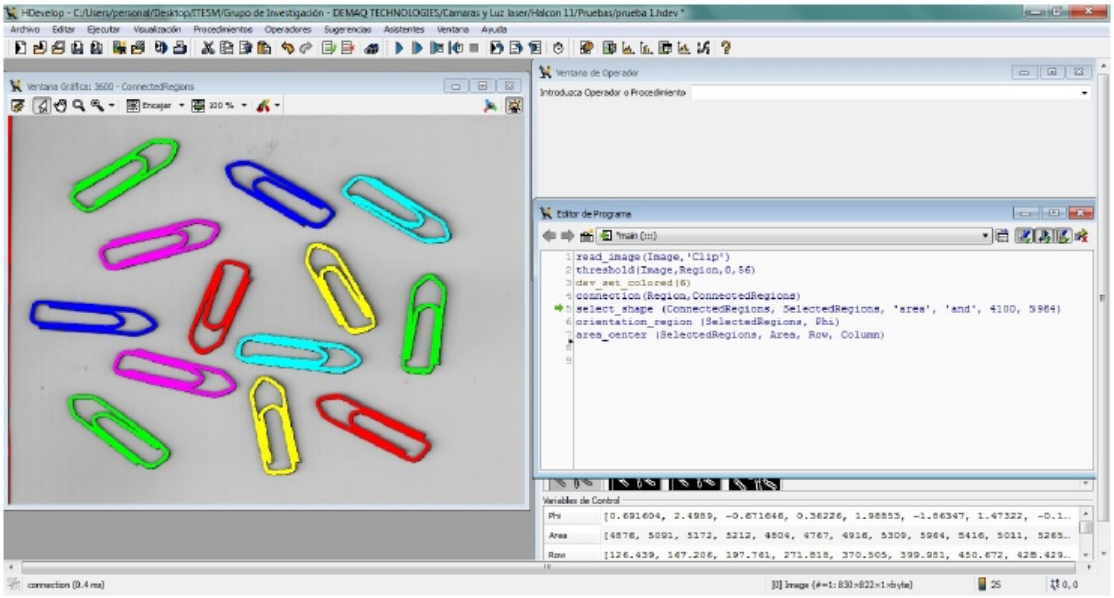


Figure 4.6: Applying conection regions

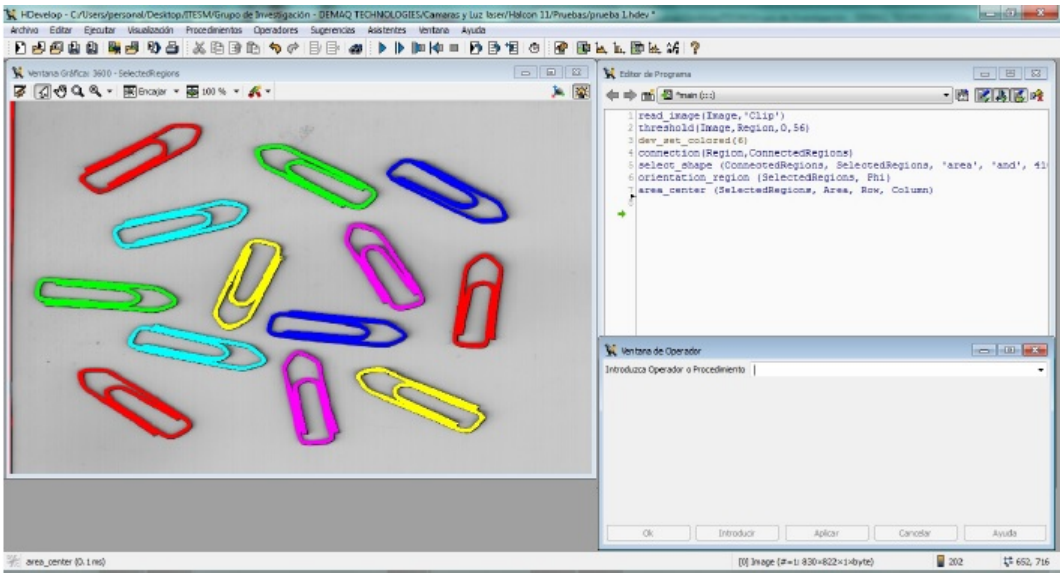


Figure 4.7: Applying conection regions and orientation

4.1.5.2 Acquisition of an images

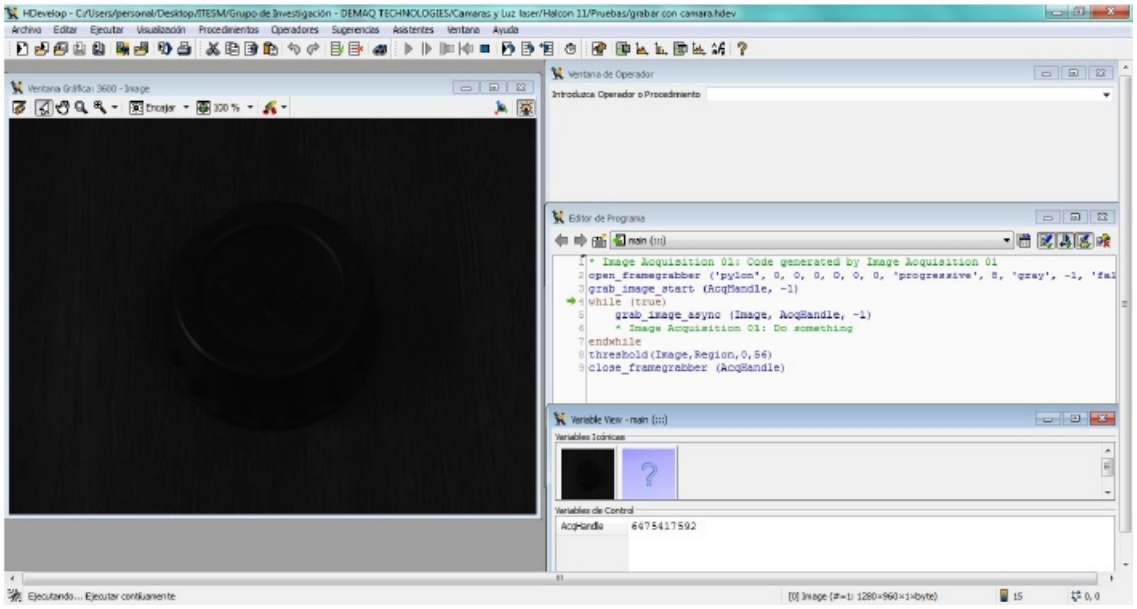


Figure 4.8: First Method using while

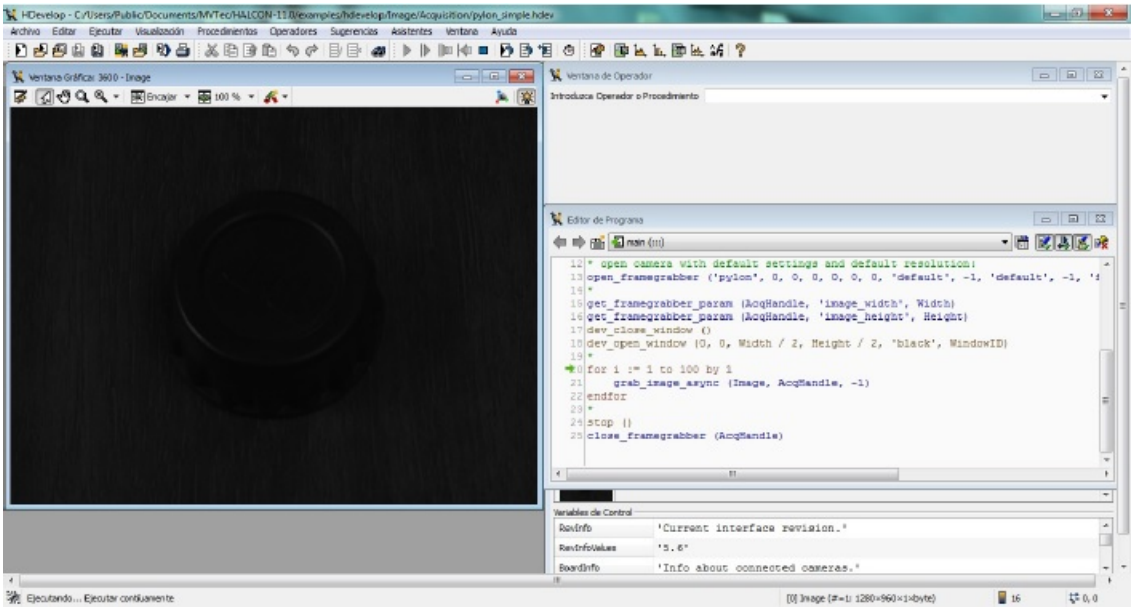


Figure 4.9: Second Method using for

4.1.5.3 Applying thresholds to an acquisition of an image

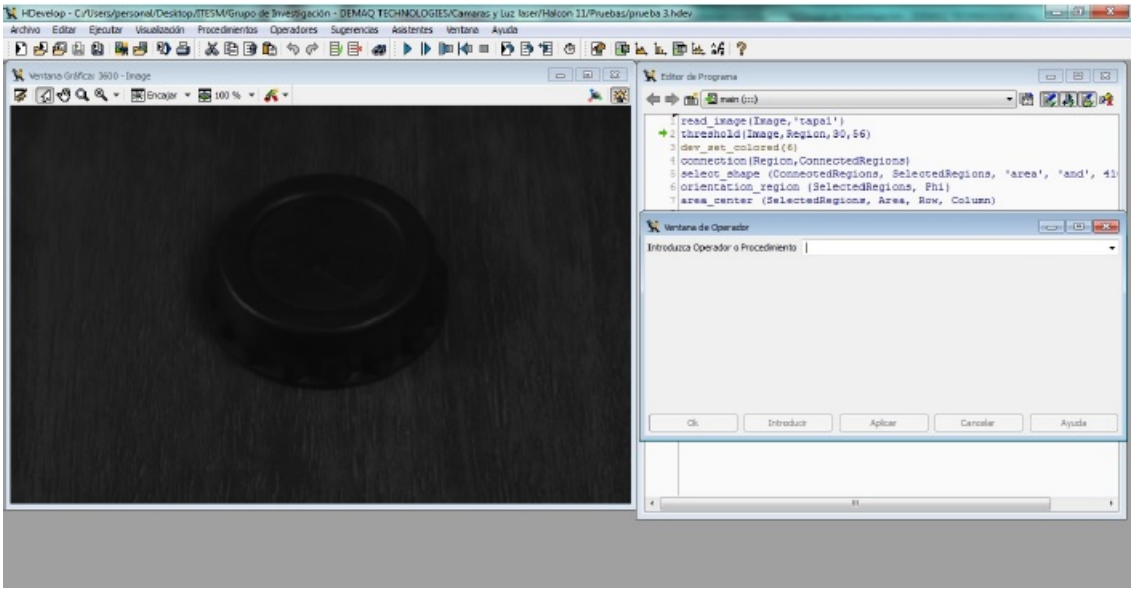


Figure 4.10: Acquisition of an image

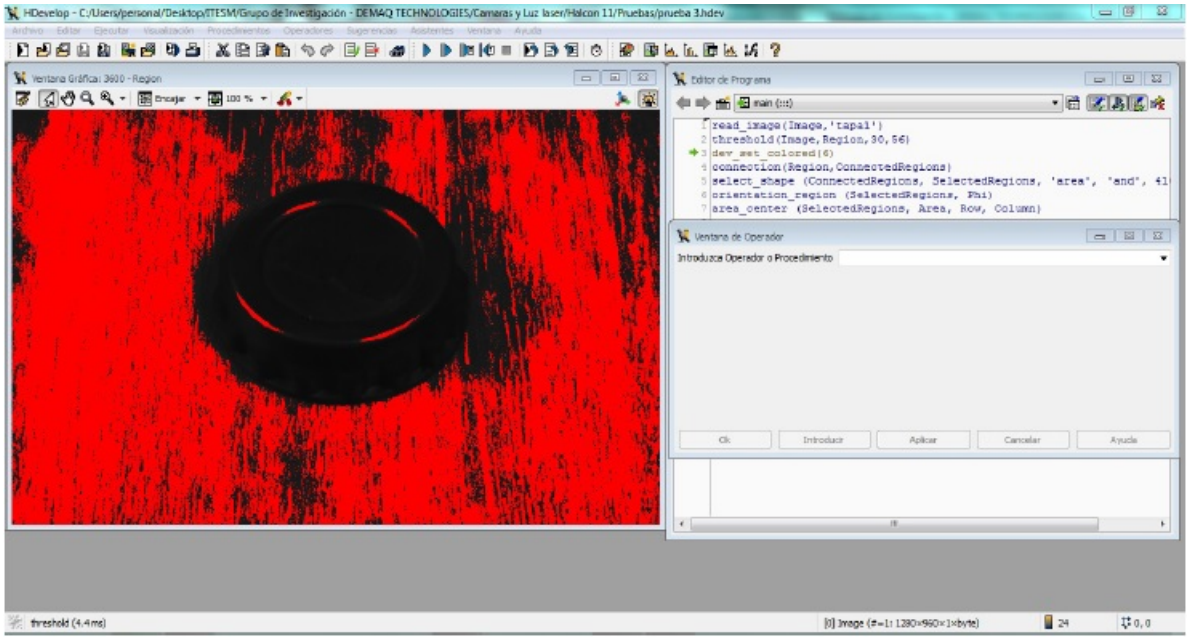


Figure 4.11: Applying threshold to an image

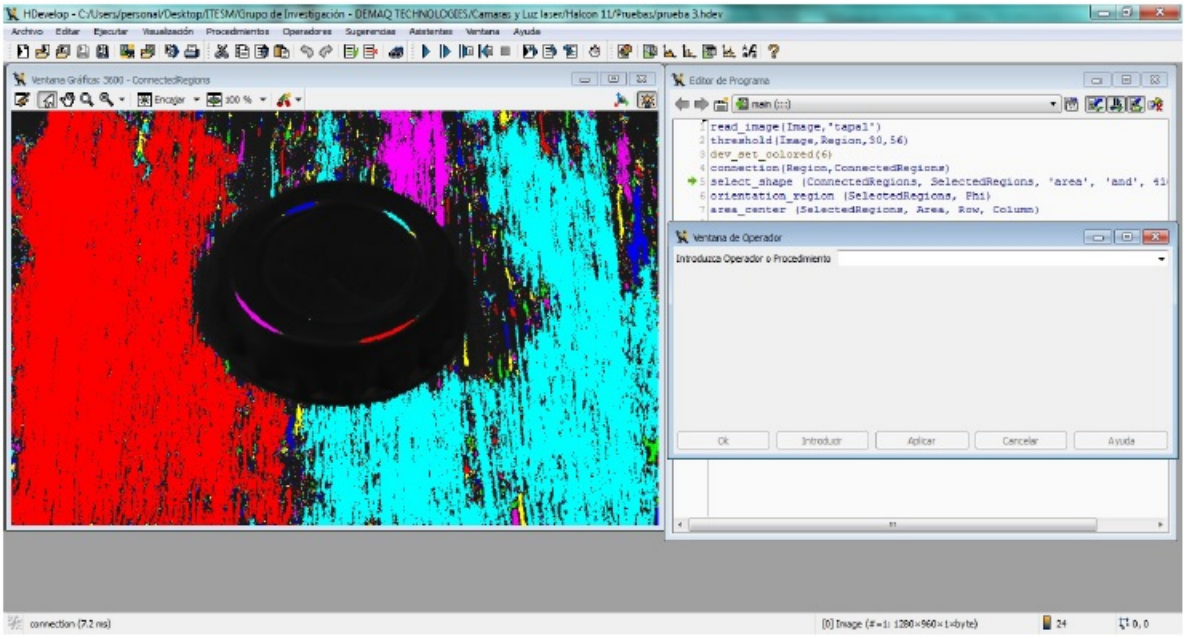


Figure 4.12: Applying conection regions to an image

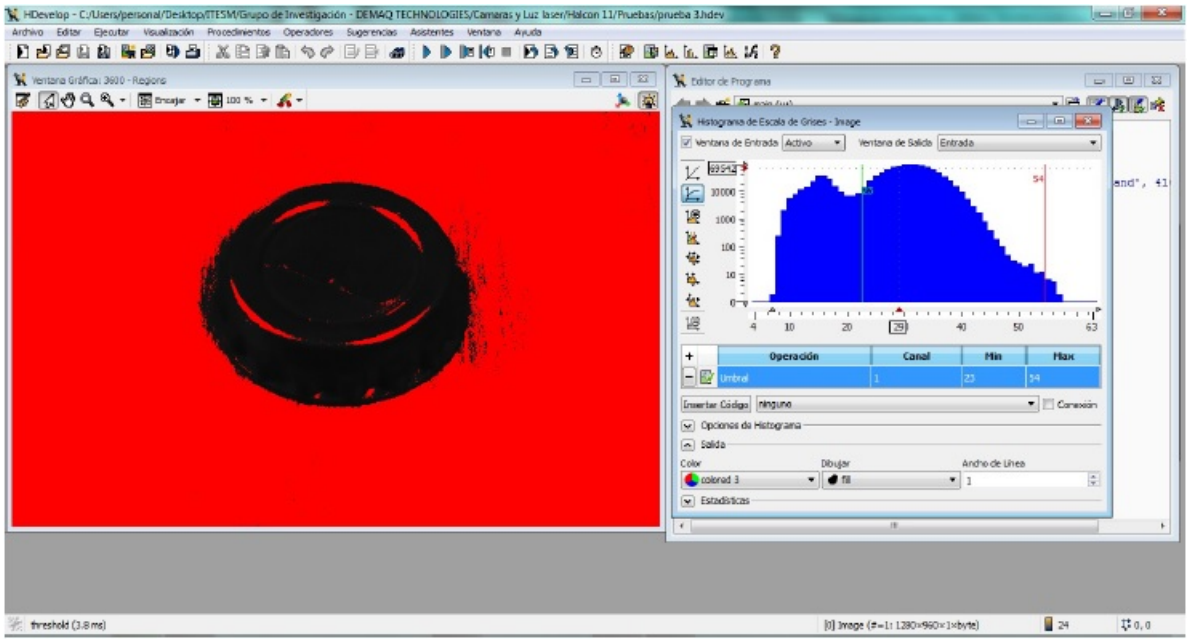


Figure 4.13: Applying grayscale histogram using region 1

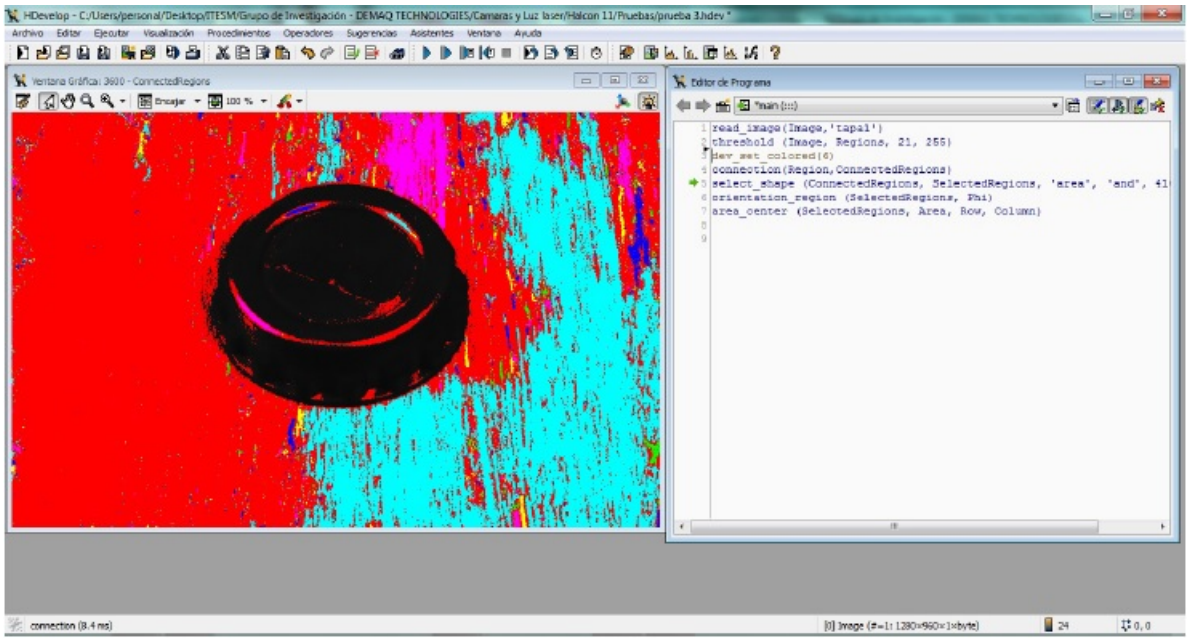


Figure 4.14: Applying region and shape

Chapter 5

Calibration

5.1 Introduction

The calibration of a camera [4] is the first step to solving applications where it is necessary to obtain quantitative data of the image. Although it is possible to obtain information of the scene from images taken with uncalibrated cameras, the calibration process is essential when it comes to obtaining measurements of the same. Accurate camera calibration allows for real-world distances from images taken of it. With this information it is possible to solve industrial parts assembly applications, avoid obstacles on the navigation of a robot, control a robot arm or perform path planning. If instead we focus on the 3D reconstruction of objects, each image point determines an optical beam passing through the optical center of camera to the scene. The management of multiple images of the same scene in which there is no movement, can relate the two optical beams for the position of the 3D point in the scene. In this case it is necessary to solve the previous step of correspondences of an object in the different images. Once it has been possible to perform the 3D reconstruction of the object, it can be compared with a stored model to determine the result of imperfections in the same manufacturing process, which involves a significant improvement over human inspection.

5.2 Theory

In the calibration process, it is considered that the transition from the coordinates of a point in the world system to the coordinates on the camera system.

5.2.1 Camera Model

There are several models [8] that define the geometry that indicates how a real world object is projected onto a plane (ex: CCD sensor), becoming a 2D image. The simplest is based on the concept of camera obscura (Pin-Hole).

As can be seen in the figure below, a pinhole camera is a box that allows light rays enter inside

through a small hole and influence on a photosensitive surface. Therefore, an object located a distance Z relative to the perforated wall is inverted projected on the image plane, located a distance f from said opening.

The size of this hole only allows the passage of a very small amount of light, that is why wearing glasses is necessary in practice. This allows increasing the number of rays that pass through, so that the generation of the image in poor lighting conditions is provided. In this case, slightly varies the manner in which the rays strike to the image plane, but pinhole camera model is still considered valid, even when used one or more lenses.

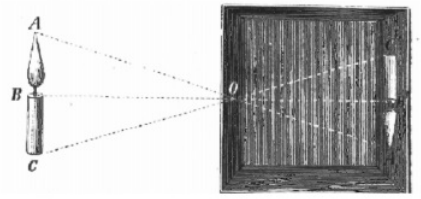


Figure 5.1: Optical principle of camera obscura

In the figure 55.2 it can see a geometric representation equivalent pinhole camera model, which facilitate the mathematics used from now on.

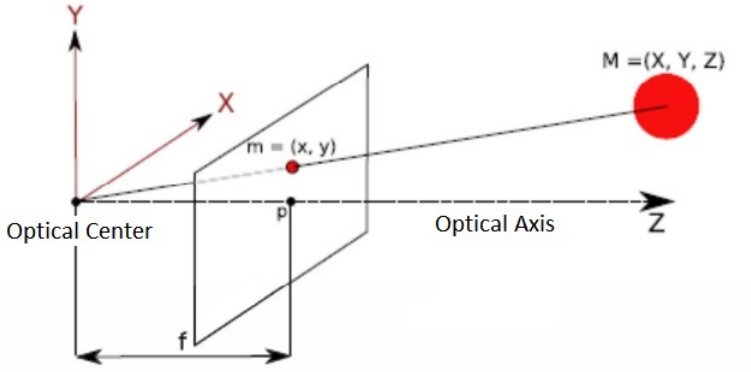


Figure 5.2: Geometric representation of the mathematical model based on the camera obscura.

Here are some considerations to keep in mind:

- The C is the optical center of the lens point such that any ray of light passing through it suffers no deviation. Nothing shall be considered as the origin of coordinates system.
- The optical axis is the imaginary part of the optical line and cut perpendicularly to the image plane center.
- The focal length f is the distance from the optical center to the focal plane.
- The focal plane or image plane is located at $Z = f$. Is virtual plane where the image is formed without any investment.
- The main point p, is the intersection of the optical axis with the image plane. Typically, it is that is not aligned with the center of the camera's sensor.

- If the above geometry viewed from the direction of the axis X is the figure (a). Similarly when viewed from the Y axis direction, is the figure (b). Because in both cases, a similarity of triangles is met, we can say the following.

$$\frac{y}{f} = \frac{Y}{Z} \vee \frac{x}{f} = \frac{X}{Z} \tag{5.1}$$

$$(X, Y, Z) \Rightarrow \left(\frac{fX}{Z}, \frac{fY}{Z}\right) \tag{5.2}$$

According to this model, every point M of the real world, will become a point m of an image according to the relationship:

$$m \approx PM \tag{5.3}$$

Where P, is the projection matrix, which will be described in detail along the next section.

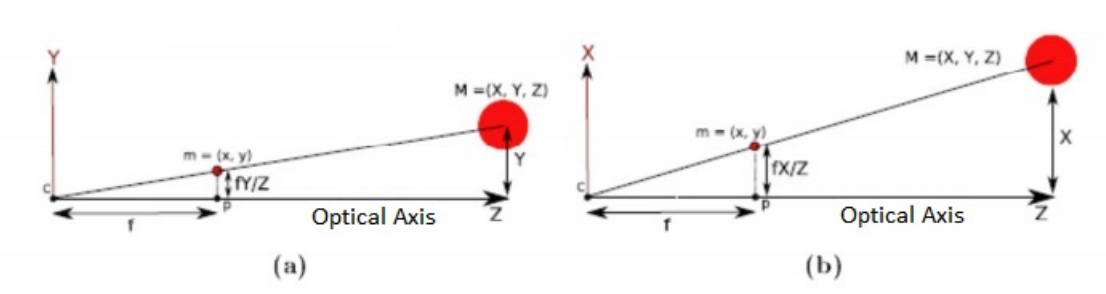


Figure 5.3: Model of obscura camera seen from the direction of the X axis and Y axis respectively

In others words, two different types of lenses are relevant for machine vision tasks. The first type of lens effects a perspective projection of the world coordinates into the image, just like the human eye does. With this type of lens, objects become smaller in the image the farther they are away from the camera. This combination of camera and lens is called a pinhole camera model because the perspective projection can also be achieved if a small hole is drilled in a thin planar object and this plane is held parallel in front of another plane (the image plane).

The second type of lens that is relevant for machine vision is called a telecentric lens. Its major difference is that it effects a parallel projection of the world coordinates onto the image plane (for a certain range of distances of the object from the camera). This means that objects have the same size in the image independent of their distance to the camera. This combination of camera and lens is called a telecentric camera model.

5.2.2 Intrinsic Parameters

[12] The projection matrix P of equation (5.3), transforms the coordinates of a 3D point in the real world in pixels of an image. It is constructed from a matrix K and a vector of null values:

$$P = [K|0] \tag{5.4}$$

Where K is the calibration matrix, which is formed by a set of parameters called intrinsic parameters:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_x \\ 0 & 0 & 1 \end{pmatrix} \tag{5.5}$$

- c_x and c_y indicate the shift of the image plane coordinates, relative to the main point. Is void only if the optical axis coincides with the center of the camera sensor, the optical axis is not always through the center of the image generated.
- The factor s (skew factor) determines the degree of perpendicularity of the walls of the pixel sensor. It is inversely proportional to the tangent of the angle between the X and Y axes, so that s will have a null value if the pixels are rectangular. This is usually the case in nearly all sensors used today.
- f_x and f_y are two focal distances in pixels. These are proportional to the focal length f considered in equations 1 and 2 as:

$$f_x = fS_x \tag{5.6}$$

$$f_y = fS_y \tag{5.7}$$

Where:

- f is the natural lens focal length, in units of length (millimeters, micrometers, etc).

- S_x and S_y are the number of pixels per unit length of the sensor, along the x axis to the y axis respectively. Obviously, if the sensor has the same number of pixels per unit length in all dimensions, the two focal f_x and f_y have the same value.

5.2.3 Distortions

For the relationship (3) is valid in practice, it is necessary to take into account distortions that occur during the formation of images.

5.2.3.1 Radial Distortion

[12] Using lens facilitates the entry of light, an appropriate approach and versatility, but also introduces distortions in the images formed on the sensor. One such effect is the radial distortion or barrel distortion, and is shown schematically in the figure below. This distortion is due to some

lenses cause rays farther from the optical center, curl much more than those that directly affect near the center of the lens.

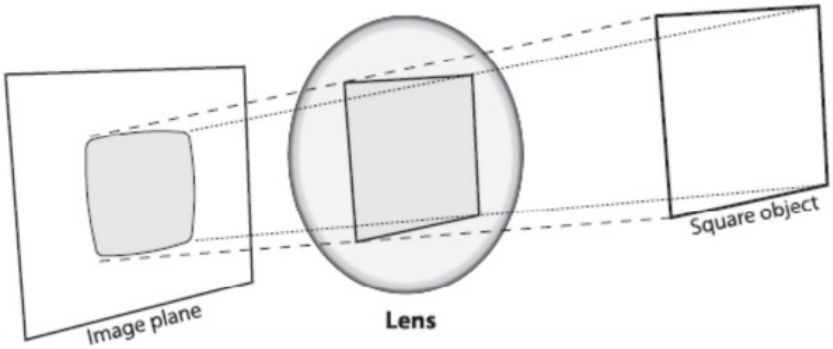


Figure 5.4: Outline of the radial distortion in a camera

This type of distortion is most pronounced in the area close to the limits of the images, as can be seen in the example below. It is important to note that also grows with decreasing focal length of the lens used or when poor quality optics are used.

The radial distortion can be modeled by the Taylor series around $r = 0$ Thus the coordinates of an image distortion are as follows:

- $(x_{dist}; y_{dist})$ is the position of a pixel in the distorted image.
- $(\tilde{x}; \tilde{y})$ is the position of a pixel in the corrected image.
- $\tilde{r}$ is the distance from the center which is expressed as $\sqrt{\tilde{x}^2 + \tilde{y}^2}$.
- $L(\tilde{r})$ is the distortion factor and is equal to:

$$L(\tilde{r}) = \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...\right) \tag{5.8}$$

  Where the k_i are the coefficients of distortion.

In practice only the use of the coefficients k_1 and k_2 is usually required. Conversely, for those lenses that introduce a lot of distortion (ex: cheap lenses, fisheye, etc.) must be taken into account k_3 for the correction is done properly.

5.2.3.2 Tangential Distortion

[12] Another type of distortion that must take into consideration when working with images is the tangential distortion. In this case, the deformations are due to a problem of the camera, no lens.

It may be that during the manufacturing process, the sensor is securely attached to the wall on which it rests. This situation occurs in the following figure. In this way the lens will not be used parallel to the plane where the image is formed.

Figure 5.5: Radial distortion corrected

The tangential distortion can be described simply with two parameters p_1 and p_2, which transforms the coordinates of the image.

$$\tilde{x} = x_{dist} + (2p_1 y_{dist} + p_2(r^2 + 2x_{dist}^2)) \tag{5.9}$$

$$\tilde{y} = y_{dist} + (2p_1 x_{dist} + p_2(r^2 + 2y_{dist}^2)) \tag{5.10}$$

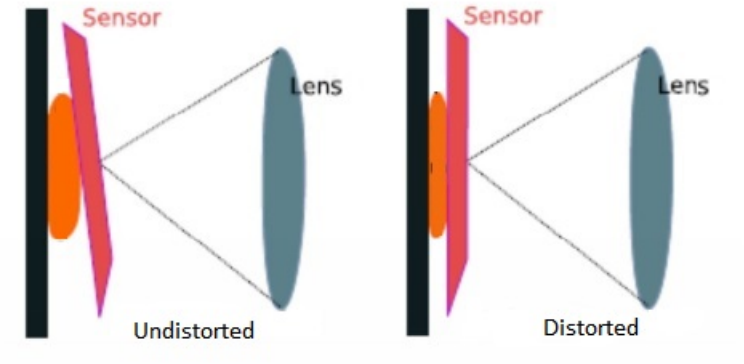


Figure 5.6: Effect that causes tangentially distortion in the camera

5.2.4 Extrinsic Parameters

[12] Furthermore, in most practical applications it is necessary that the camera shake or rotate, to properly capture the scene.

$$m \approx PWM = P'M \tag{5.11}$$

$$W = [Rt] \tag{5.12}$$

Where:

- R is a rotation matrix which represents a rotation of the camera (or an object with respect to it). Have a different shape depending respect to axis (X, Y, Z) the rotation is made:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Psi & cos\Psi \\ 0 & -sen\Psi & cos\Psi \end{pmatrix} \tag{5.13}$$

$$R_y = \begin{pmatrix} cos\varphi & 0 & -sen\varphi \\ 0 & 1 & 0 \\ sen\varphi & 0 & cos\varphi \end{pmatrix} \tag{5.14}$$

$$R_z = \begin{pmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.15}$$

  If the rotation is relative to the Z axis, as drawn in the figure below, the new coordinates will be as follows:

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} cos\theta & sen\theta & 0 \\ -sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \Leftrightarrow \begin{matrix} X' = Xcos\theta + Ysen\theta \\ Y' = -Xsen\theta + Ycos\theta \\ Z' = Z \end{matrix} \tag{5.16}$$

If the rotation is made about the other two axes (X or Y), the operation would analogously, using the appropriate rotation matrix R. addition be observed that corresponding to said rotation axis component would not altered.

- t is a translation vector representing a displacement of the coordinate system. It indicates a change in position of the camera or an object about it. Thus $M_{final} = M_{inicial} - t$.

  1. A three-dimensional rotation is defined by three angles varying (Ψ,φ,θ).
  2. A translation in space is specified by three parameters (x, y, z).
  3. The own intrinsic parameters of the camera are five (f_x, f_y, c_x, c_y, s).

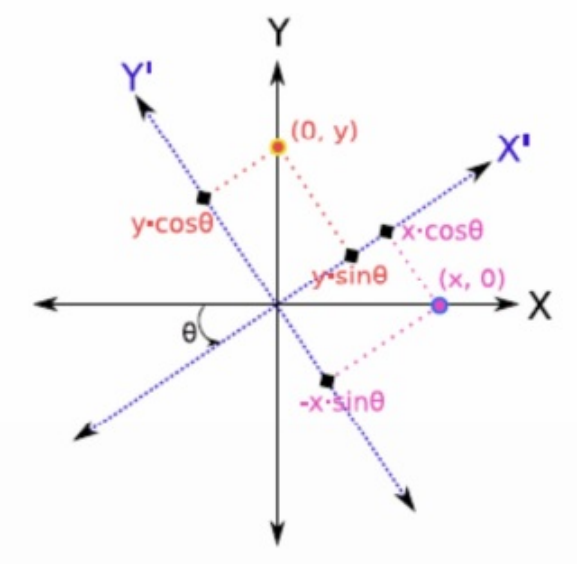


Figure 5.7: Camera rotation about the axis Z

The conclusion is that it is necessary to know these eleven parameters for each image it have generated the same camera, if the expression (5.11) is used. What you will see in the next section, is the method that has been followed in this project to obtain these parameters.

Besides, the calibration of the camera is the estimation of the intrinsic parameters of the same which model the internal geometry of the camera and the optical characteristics of the sensor. The extrinsic parameters measure the position and orientation of the camera relative to the coordinate system established for the scene. These give the relationship with respect to the coordinate system of the user instead of the coordinate system of the camera.

5.2.5 Techniques for camera calibration

5.2.5.1 Photogrammetric Calibration

The photogrammetric calibration is performed by the observation of patterns whose geometry in 3D space is known with a good level of accuracy. The patterns of calibration standards are typically positioned in two or three orthogonal planes between them. In some cases, only a single plane, whose translation is well known. This type of calibration requires an elaborate setup, but its results are efficient.

5.2.5.2 Autocalibration

This method is based on the movement of the camera observing a static scene from traveling and using only the image information. The rigidity of the scene generally imposes restrictions on the parameters of camera. Three images taken by a single camera with fixed intrinsic parameters are sufficient for the extrinsic and intrinsic parameters. Although this technique is very flexible, it is not mature yet.

5.2.5.3 Pin-Hole Model

The camera model [65] traditionally used to pass real 3D coordinates to 2D coordinates belonging to the captured image, usually the perspective projection model called pin-hole model. In this model all rays from a certain object pass through a fine hole to impact the image sensor. Since the lenses do not have this linear behavior, the pin-hole model must be corrected with a value of distortion, ie, must be supplemented with parameters that correct your ideal behavior and approach, as far as possible, the actual behavior of the target .

The reference system of the camera is placed in the center of the projection, the z axis coinciding with said system with the optical axis, also called axial axis. In this arrangement of axes, the image plane coordinate u, v, is situated at an equal to the focal length of the lens distance perpendicular to the optical axis. The intersection of the optical axis with the image plane is called the principal point.

The projection center C of the camera is assumed constant but is a priori unknown. The image plane is usually placed in front of the projection center C to take a picture without investment. The following figure shows a schematic pin-hole model shown.

5.3 Calibration

The problem is to find and solve a mathematical model of how the camera sees the scene, solve understood the process of finding a set of values called parameters, so that then allow to obtain three-dimensional information from images.

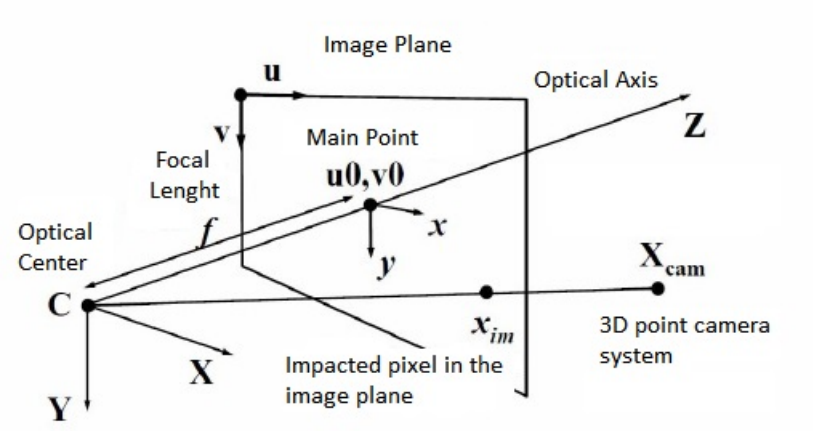


Figure 5.8: Diagram of the pin-hole model.

Because the calibration parameters involved in the imaging process are obtained, then the calibration is the process by which the relationship between three-dimensional coordinates of objects in the vicinity with corresponding two-dimensional image projections set.

5.3.1 Techniques according to the calibration target

Depending on the object used to calibrate, you can highlight two types of calibration, coplanar and non-coplanar.

5.3.1.1 Coplanar

The object used is flat, with a printed pattern which is called calibration grid, for example the one shown in the figure below.

Figure 5.9: Coplanar

5.3.1.2 Non-coplanar

The object used is typically a cube, wherein the different faces are the same printed pattern, or a grid composed of two planes, each with the same pattern as shown in the following figure. A object points of known three-dimensional positions are known as calibration points.

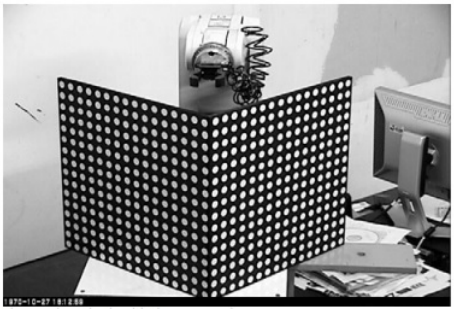


Figure 5.10: Non-coplanar

Another technique is not coplanar in a field where several calibration points distributed on the plane and a set of points are raised to have spatial information as shown in the following figure.

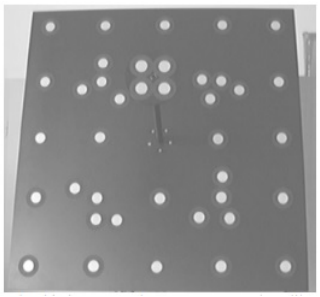


Figure 5.11: Non-coplanar with calibration points outside the plane

5.3.2 Calibration methods

The cameras or image sensors are the elements responsible for capturing light scene information and transmit it to the computer as an analog or digital signal. Once the image has been taken up by the video camera, acquisition card and image processing receives the analog signal sent by it, to turn it into a digital signal to be processed.

The calibration of a camera is a need for measures of the scene from images in the same step. The accuracy of the calibration subsequently determine the accuracy of the measurements that are made from images. It is for this reason that it is essential to calibrate the camera with full guarantees that the parameters obtained are closest to the real. This commitment involves the choice of calibration method and the proper use thereof.

There are different methods of camera calibration as well as different ways of classifying; in this section some of the currently used methods among these are referenced:

5.3.2.1 Tsai method

Tsai method [4] represents a classic calibration process based on measurements of coordinates of points of a 3D template relative to a fixed reference point. This shows excellent results, a single image of the calibration being required, although a significant accuracy of data input is needed to achieve them.

Tsai method obtains an accurate camera parameters if the input data are slightly contaminated with noise estimate. Given that it takes at least a hundred points in the template and coordinates must be referred to a fixed coordinate origin, plus the orientation and position of the camera with reference to the coordinate axes, these parameters are: Rx, Ry and Rz representing rotation angles between the axes transform the world and Tx, Ty and Tz camera are the components of the translation vector for the transformation between the camera axis and the world. Additionally, five intrinsic parameters related to camera constants that are known are needed, because only enough to look at the information the manufacturer provides the camera must supply. These other constant parameters are: Ncx, number of sensor elements in the horizontal direction; Nfx, number of pixels in the horizontal direction of the camera; dx, width of each part of the sensor of the camera (mm/el); dy, high of each element of the sensor of the camera (mm/el); dpx, effective width of a pixel in the camera (mm/pix); dpy, high cash from a pixel in the camera (mm/pix).

3D template consist two or three orthogonal planes to each other, which leads to a laborious and expensive to carry out, taking into account that it is essential for proper design of the calibration well as an accurate measurement of the coordinates of the points, for get good results at the end of the procedure. Nevertheless, the possibility of errors in the measurements are high.

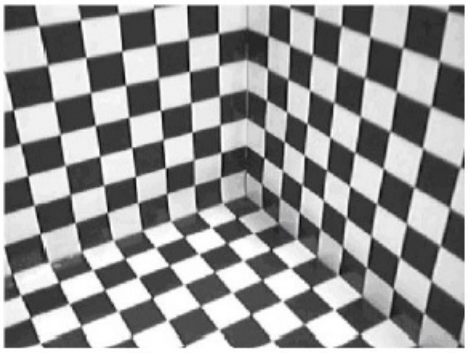


Figure 5.12: 3D Calibrating template

5.3.2.2 Zhang method

Zhang [4] proposes a calibration technique based on the observation of a flat blank from various positions. This method uses the coordinates of the points within a flat 2D template taking different pictures of it from different positions and orientations. Thus the advantages of the calibration methods based on measurements of the coordinates of the template with the advantages of self-calibration in which it is not necessary to use combined template.

This calibration mode is very flexible from the point of view of both the camera and the template can be moved freely and it can take as many pictures as you want without having to perform measurements on the template.

The 2D template does not require a special design template, nor as accurate measurement of its points. Furthermore, the sensitivity calibration algorithm to errors in measurements can be improved by increasing the number of points in the template, simply printing a chess board with more corners.

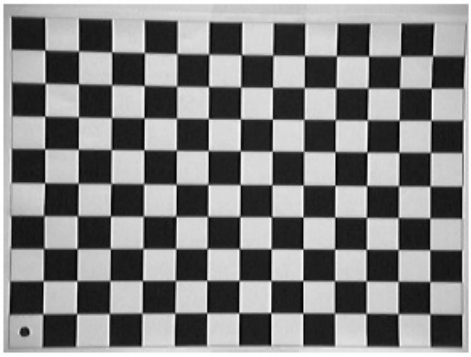


Figure 5.13: 2D Calibration template

5.3.2.3 Auto Calibration

These techniques use no calibration object because it is only necessary to relate a point in different images. Only by moving the camera in a static scene, the rigidity of the overall scene causes two restrictions dento intrinsic camera parameters. Therefore several images of the same scene taken at the same intrinsic parameters, the correspondence between three images are sufficient to calculate both the intrinsic and extrinsic parameters. In these cases even a template is not necessary, is necesrio calculate a large number of parameters, resulting in a rather complex mathematical problem. Due to the difficulty to start searching self-calibration methods tend to be unstable. You also need to consider the family of algorithms that calculate the parameters that model the distortion caused by the lens in the image without using calibration objects and therefore without knowing any 3D structure. These methods rely on that in a perspective projection ideal, the camera transforms straight lines in 3D space in straight lines within the corresponding 2D image space. Therefore reinforcing the linearity of the parts of the image curves are due to the distortion of camera lens, it is possible to estimate the deformation it produces. There are methods using epipolar and trilinear constraints among pairs and triplets of images respectively to estimate the radial distortion.

It has chosen to follow the guidelines shown based on the algorithm of Zhang Zhengyou calibration and subsequent improvements. The choice of this method is due to several reasons:

- The Halcon platform comes with libraries for calibration.
- Simplicity of implementation: it is only necessary to perform multiple captures a familiar object, so that it varies its position over a series of photographs.
- Validity of results: the data obtained after calibration are as good as those obtained with other methods, without a high computational cost or investment in expensive equipment required

5.4 Implementation

For the implementation, the diagram for calibration was performed. According to this, it can follow an order and get a correct calibration of the camera.

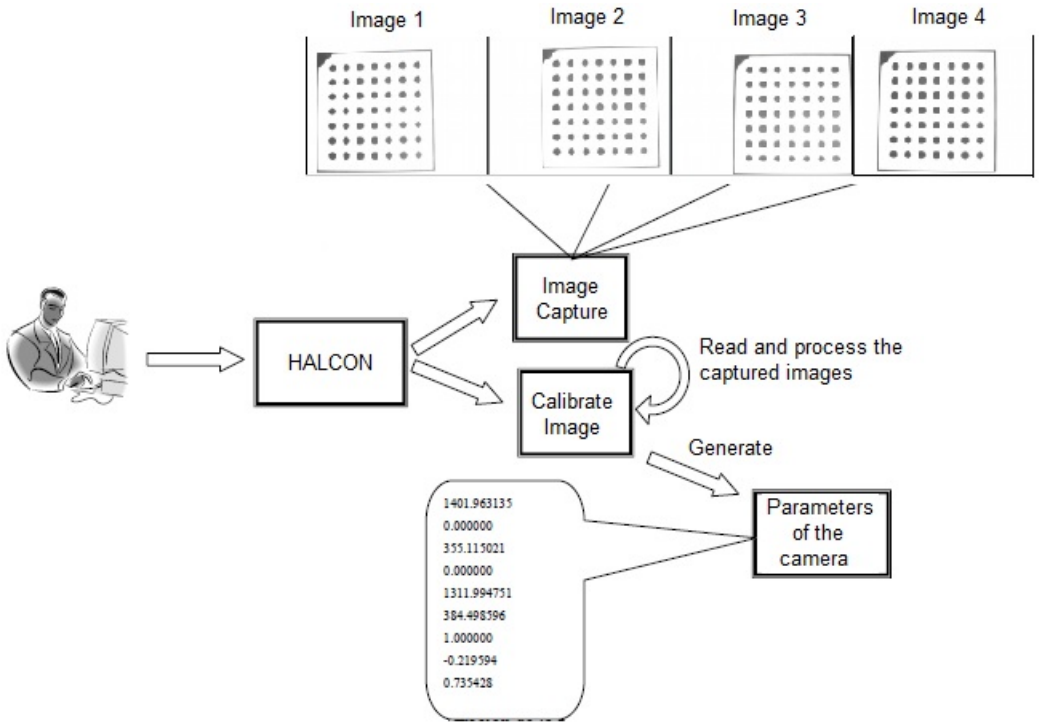


Figure 5.14: Calibration Diagram

5.4.1 Calibration

5.4.1.1 Procedure

Essential step in 3D triangulation with SL systems is their calibration, i.e. the determination of the interior orientation (focal length, principal point position, lens distortion parameters) of video projector and digital camera as well as their (scaled) relative position in space. Typically, a camera-projector calibration is carried out in two separate steps. First, the camera interior orientation is estimated, and next projector interior and relative orientations are found. In this context, use a planar surface and a combination of printed circular control points and projected targets to perform plane-based calibration. Then, using the epipolar constraint, homologies between projector and camera pixels are established and the projector is calibrated. In the camera is calibrated and then a full SL scanning of a planar object containing targets is performed to obtain correspondences between the projector and the camera pixels. This procedure is repeated with different orientations of the planar surface, and synthetic images of what the projector would capture as a virtual camera are computed and used for its calibration. Finally, adopts the same technique of "virtual" projector images, while an alternative approach is also proposed, in which a calibrated stereo camera configuration is used to compute the 3D coordinates of a projected pattern on different planar object orientations; the acquired 3D coordinates are used in a subsequent step of projector calibration.

Then, prior to the implementation which allows for the calibration step has required the use of a board points. This calibration object follows a simple pattern of black ellipses, which allows the algorithm to work effectively and accurately identify regions using Halcon.

Here are the steps that have been carried out to calculate the parameters of the camera:

1. The camera is implemented in the metal support to calibrate using a grid.

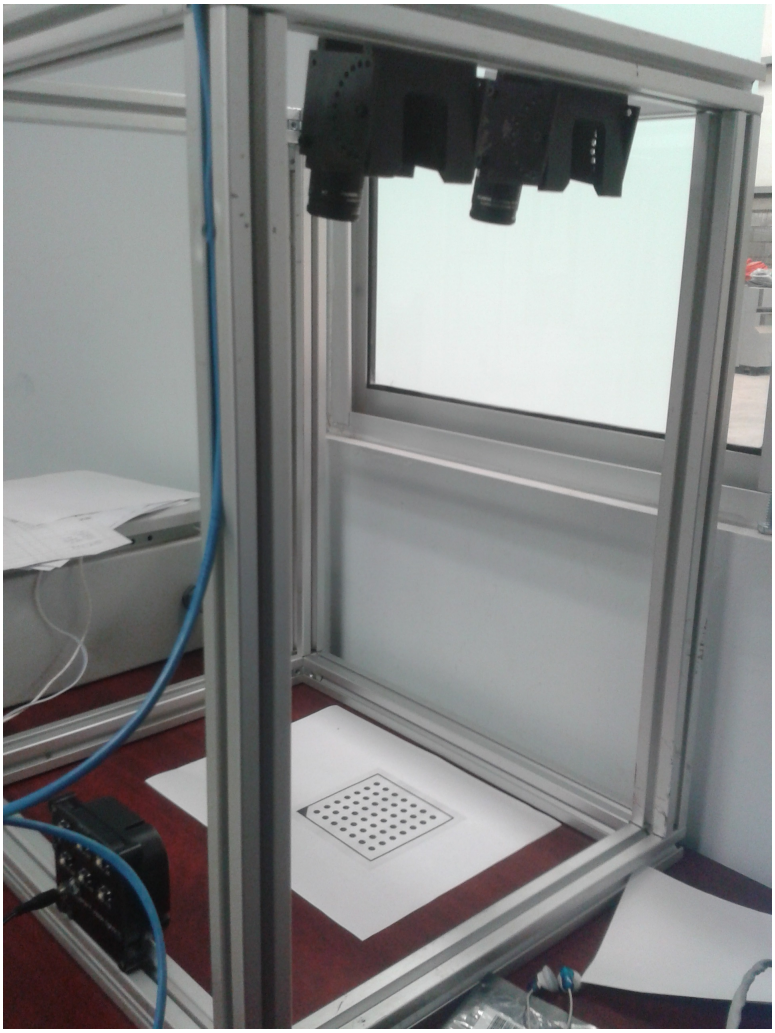


Figure 5.15: Camera in metal support prepared for calibration

2. Board of points for camera calibration.

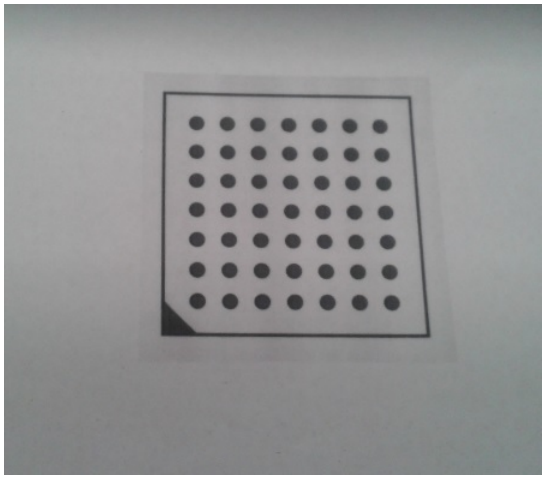


Figure 5.16: Board of points

3. Realization of a series of screenshots as shown in figure below.

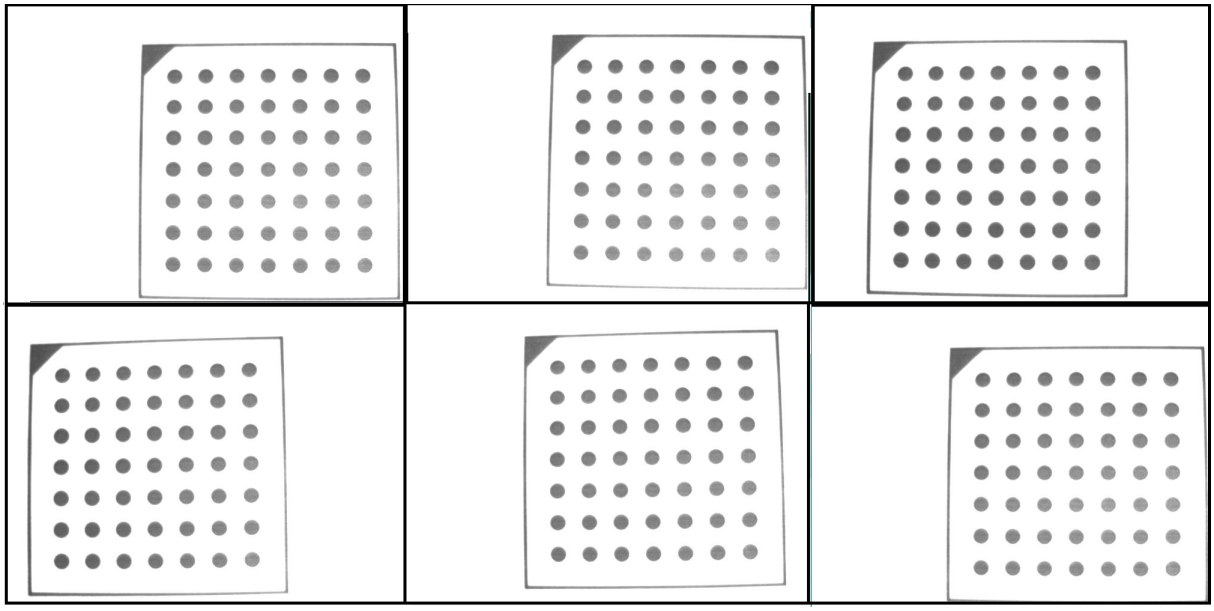


Figure 5.17: Procedure of Calibration

4. Automatic detection of ellipses inside the board. For this algorithm to be used command of Halcon. After the search process all ellipses successfully are detected, similar to the figure shown. Finally, the parameters of the camera are obtained

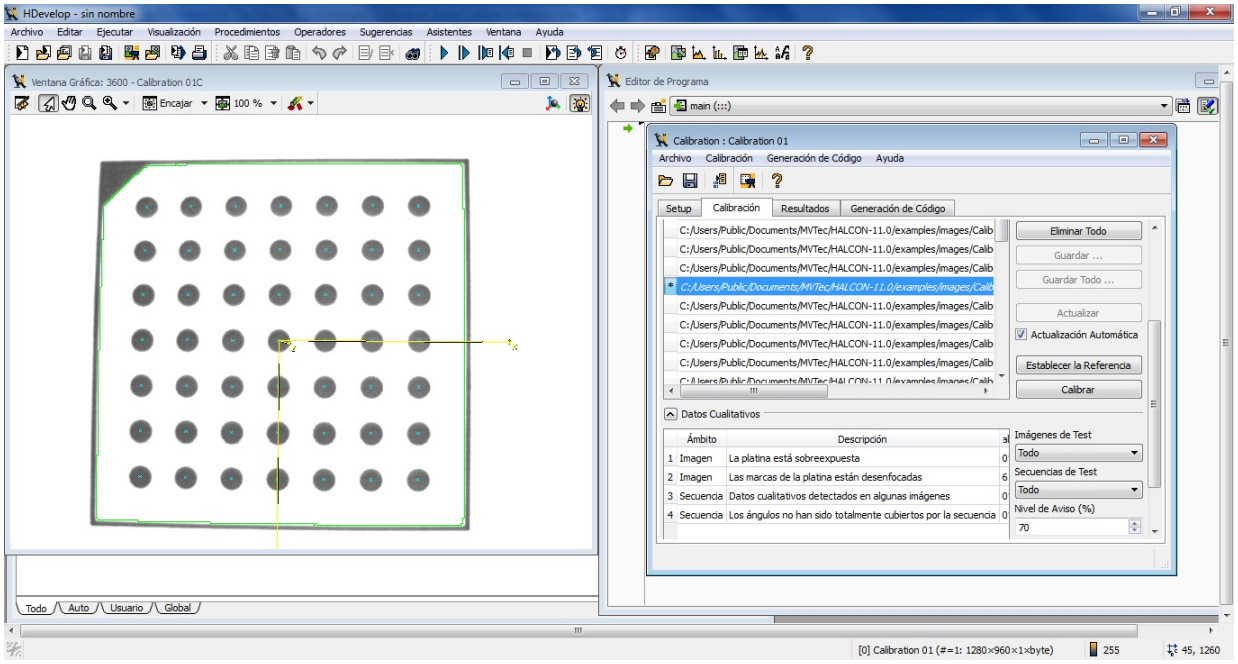


Figure 5.18: Automatic detection of ellipses inside the board

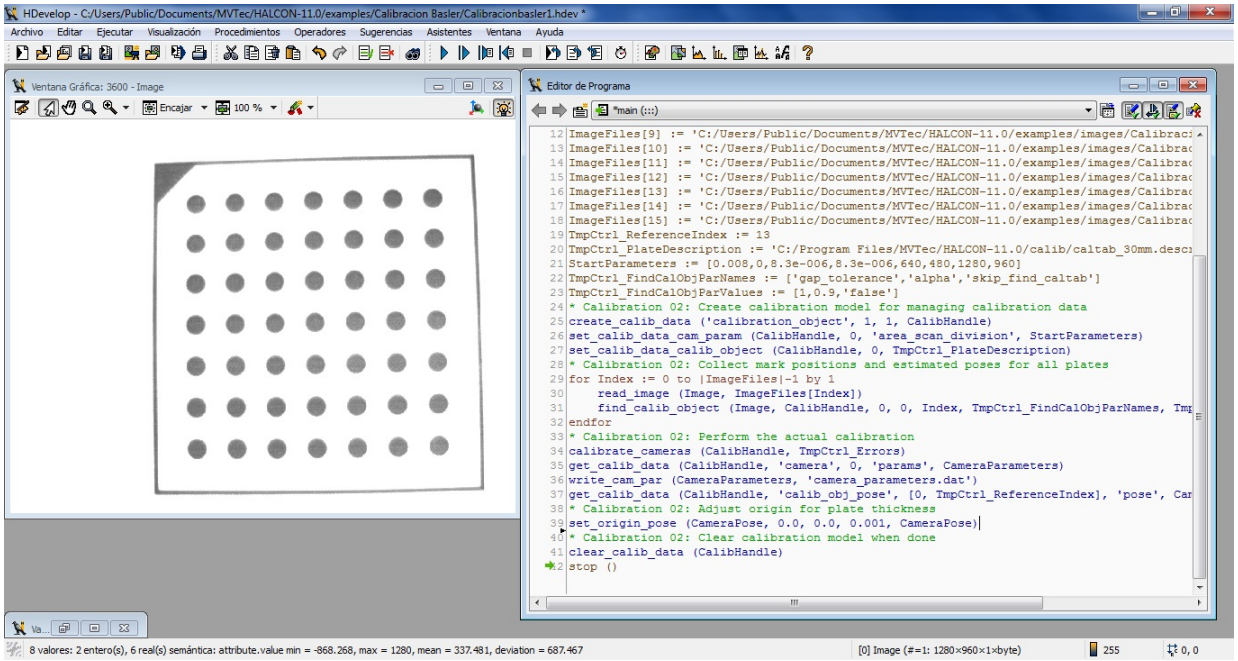


Figure 5.19: Algorithm for an automatic detection of ellipses inside the board

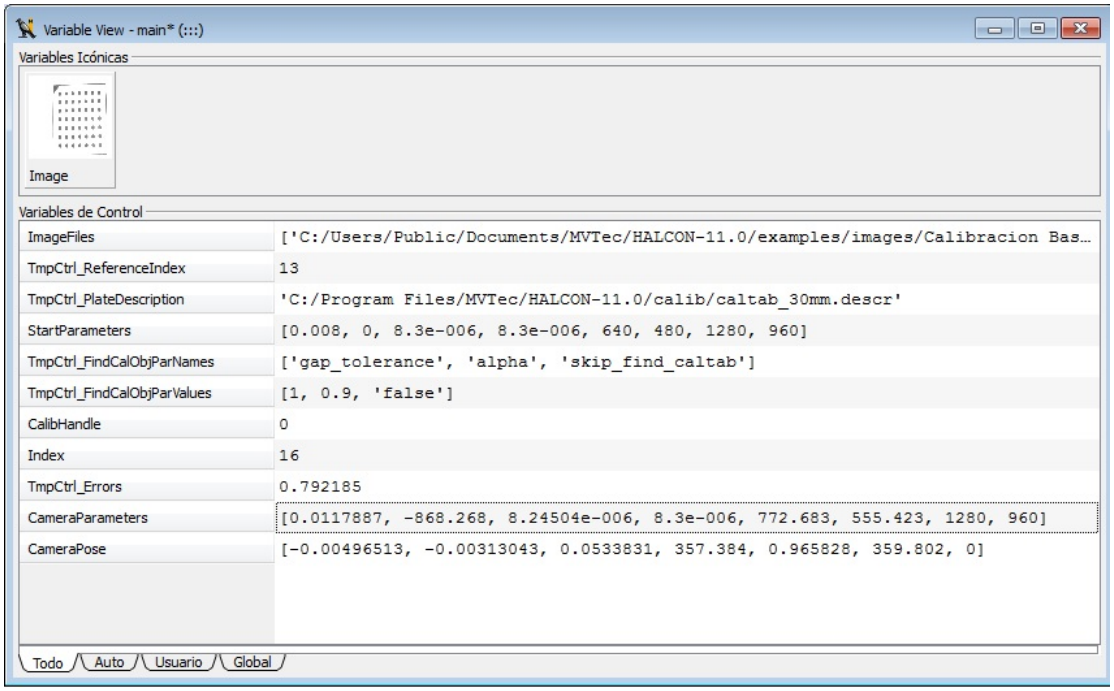


Figure 5.20: Getting the parameter of the right camera

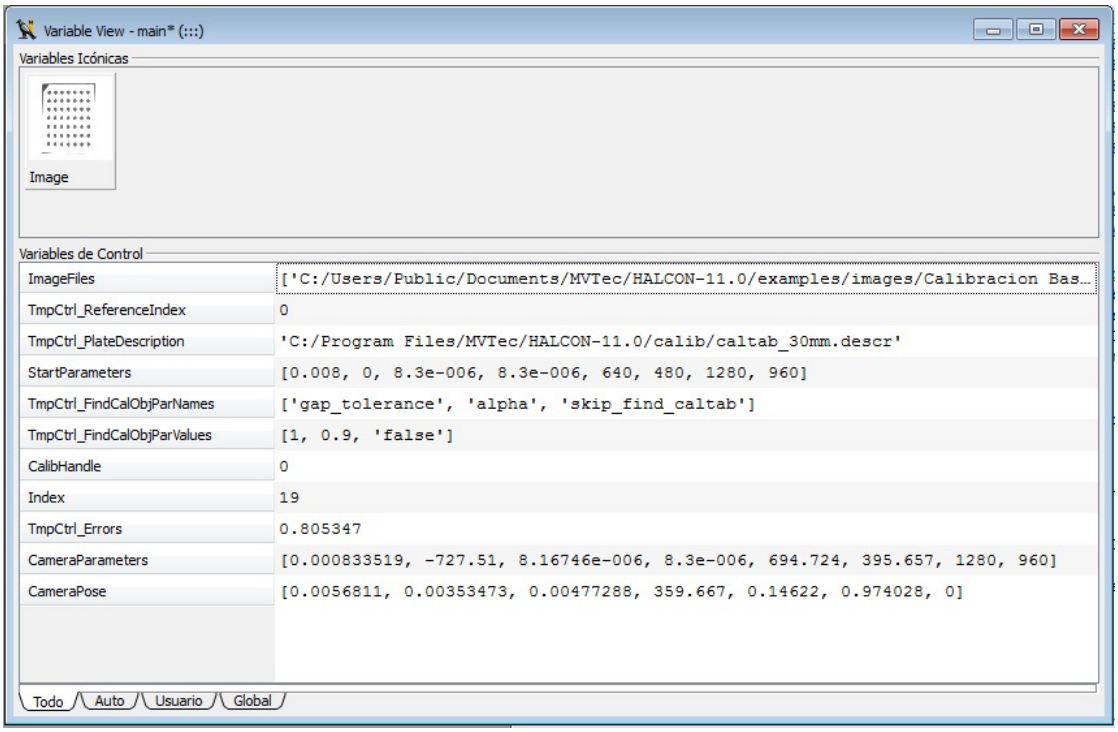


Figure 5.21: Getting the parameter of the left camera

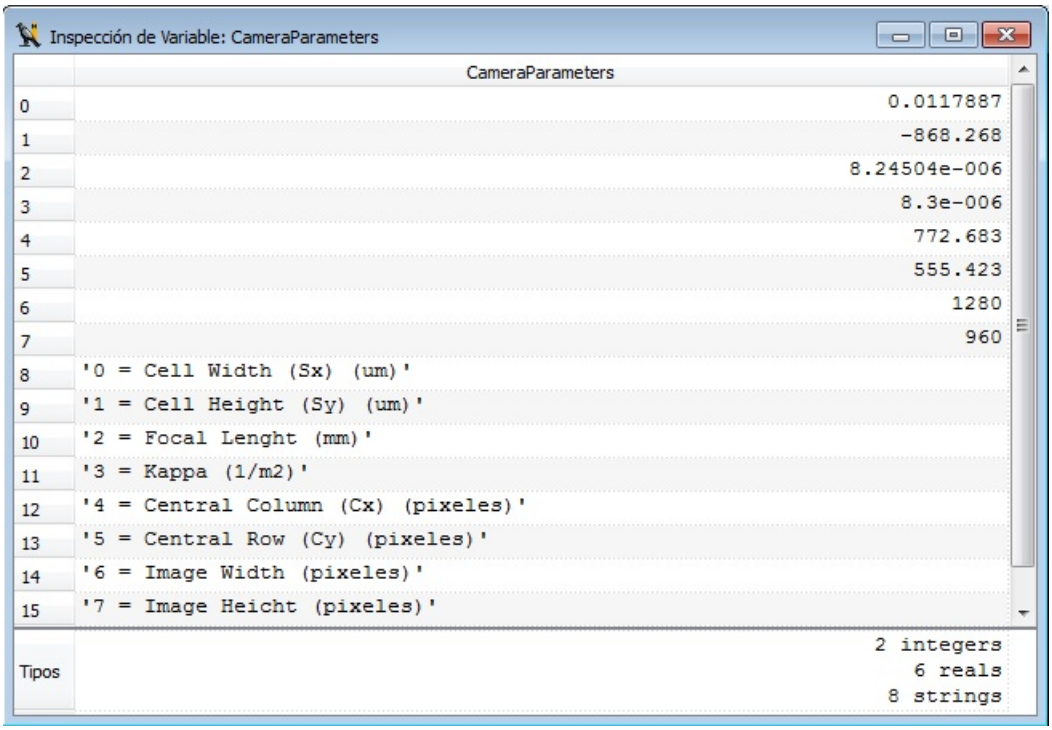


Figure 5.22: Right Camera Parameter

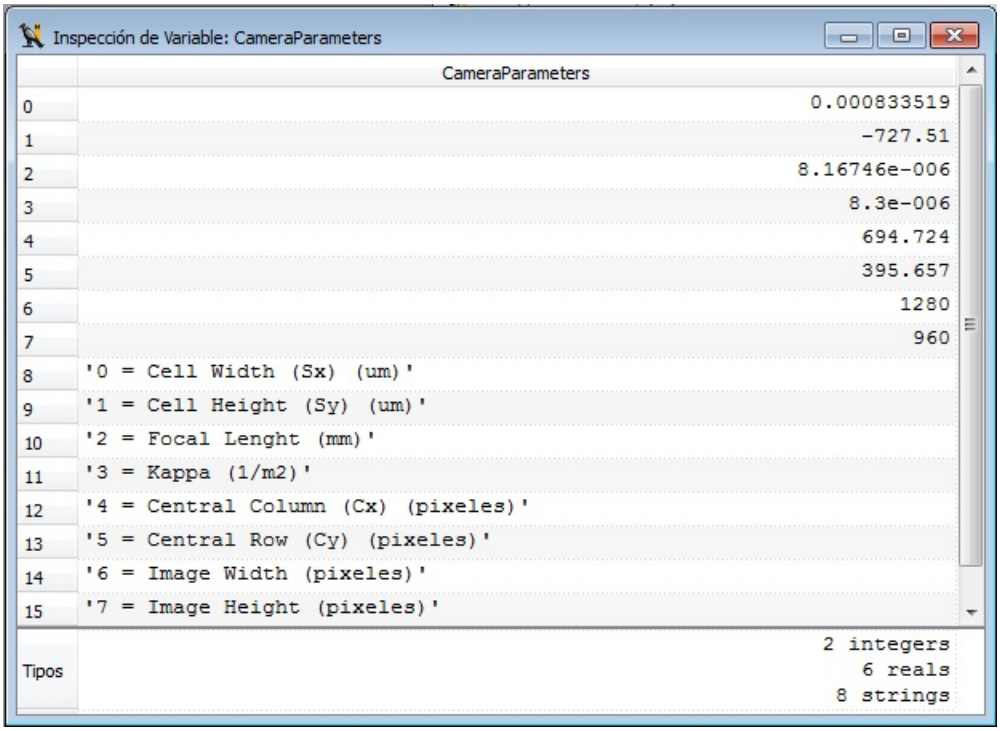


Figure 5.23: Left Camera Parameter

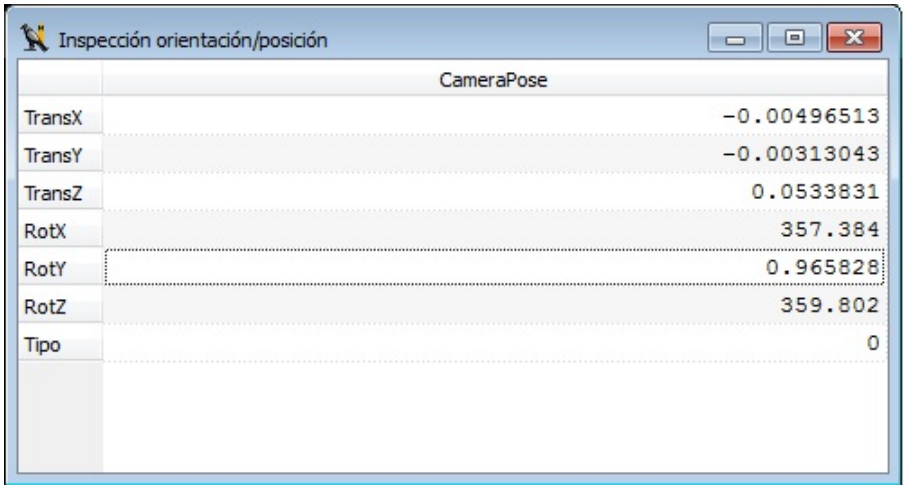


Figure 5.24: Right Camera Pose

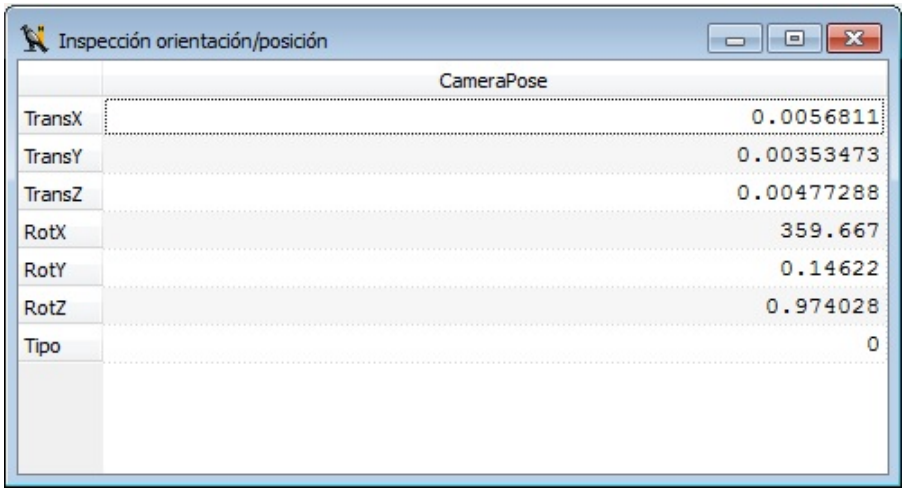


Figure 5.25: Left Camera Pose

Chapter 6

3D Reconstruction

6.1 Introduction

The 3D reconstruction is the process by which objects are scanned in a coordinate system X, Y, Z, using an encoding method of surveying, maintaining its physical characteristics (size, volume and shape). Thus, a calibration procedure or system parameters, converts the sequence of images acquired and processed digitally, full scale of the object. For this reason, 3D reconstruction methods are a useful tool in applications where the topographic information plays an important role in decision-making; as for quality control, 3D modeling of industrial objects and functional explorations strain on the back or elsewhere in the body, among others.

In this chapter basic methods where structured light technique for 3D surface measurement information to apply are presented. Encoding methods are used to solve the problem of correspondence between projected points on a surface and point in the image plane in the plane of the sensor are emphasized.

6.2 Mathematical basis for obtaining 3D information

The general principle on which the systems are based structured light to obtain three-dimensional information in measuring surfaces relies on the triangulation method.

6.2.1 Triangulation

A general mathematical [5] description of the process is given before a brief description of some methods that use this system. For this development is considered as the origin of the reference system O_1 center plane where an image is captured as shown in the following figure. The focal point of the lens coordinates $F_1 = (0, 0, f_1)^t$ and $P_0 = (x_0, y_0, z_0)^t$ point of the object in the scene the line containing the respective point $P_1 = (x_1, y_1, 0)^t$, the super index t indicates transposed in image plane given by:

$$P_1 = F_1 + \alpha(P_0 - F_1) \tag{6.1}$$

Expressed in matrix form is:

$$\begin{bmatrix} x_{p1} \\ y_{p1} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f_1 \end{bmatrix} = \alpha \begin{bmatrix} x_{p0} \\ y_{p0} \\ z_{p0} - f_1 \end{bmatrix} \tag{6.2}$$

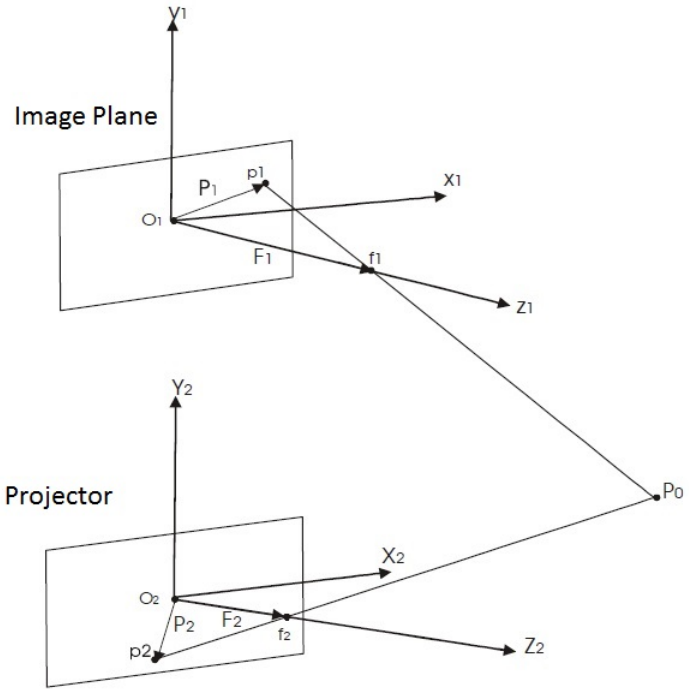


Figure 6.1: Geometry for measurement system structured light

In the plane of the projector (which generates the regulating system to project on the surface pattern) analogously to the above, an origin of coordinates is considered in the same orientation and centered with the image plane, whose origin is located in $O_2 = (x', y', z')^t$ and represent the focal point in this plane located in $F_2 = (0, 0, f_2)^t$ that with respect to the reference plane of the system is located considering O_1 translation as $F_2 = (x_2, y_2, z_2 + f_2)$ so that it get the coordinates of $P_2 = (x_2, y_2, 0)^t$ it have:

$$P_2 = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \\ 0 \end{bmatrix} = \alpha \begin{bmatrix} x' + x_2 \\ y' + y_2 \\ z_2 \end{bmatrix} \tag{6.3}$$

Therefore, considering P_0 and F_2 it has for this case:

$$P_2 = F_2 + \beta(P_0 - F_2)$$

Then:

$$\begin{bmatrix} x' + x_2 \\ y' + y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' + f_2 \end{bmatrix} = \beta \begin{bmatrix} x_0 - x' \\ y_0 + y_2 \\ z_0 - z' - f_2 \end{bmatrix} \tag{6.4}$$

As solving for equations (6.1) and (6.4) and simplifying, it have:

$$z_0 = x' + \frac{f_1 f_2}{f_1 x_2 - f_2 x_1}(x' + x_2 - x_1 + \frac{z' x_2}{f_2}) \tag{6.5}$$

$$z_0 = x' + \frac{f_1 f_2}{f_1 y_2 - f_2 y_1}(y' + x_2 - y_1 + \frac{z' y_2}{f_2}) \tag{6.6}$$

Analogously the coordinates for x_0 and y_0 are uniquely determined by knowing the coordinates of the image point (x_1, y_1) and the coordinates of the projector (x_2, y_2) plus the respective point f_1, f_2, x_2, y_2 and z_2. The translation and orientation of the image plane are determined to calibrate the system homogeneous coordinates on the basis of which it can be expressed in a single matrix translation, rotation, scaling, and perspective. Note that z_0 is determined either by knowing the coordinates x_2 or y_2 coordinates.

To relate the projection of a point on the object in the scene and the captured image are as follows: the calibration matrix that relates the position of the object points to the coordinate system of the camera is given using homogeneous coordinates, by the matrix:

$$\begin{bmatrix} x_1 \\ y_1 \\ 0 \end{bmatrix} = \begin{bmatrix} A_{111} & A_{112} & A_{113} & A_{114} \\ A_{121} & A_{122} & A_{123} & A_{124} \\ A_{131} & A_{132} & A_{133} & A_{134} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{6.7}$$

Similarly to the coordinate system of the projector:

$$\begin{bmatrix} x_2 \\ y_2 \\ 0 \end{bmatrix} = \begin{bmatrix} A_{211} & A_{212} & A_{213} & A_{214} \\ A_{221} & A_{222} & A_{223} & A_{224} \\ A_{231} & A_{232} & A_{233} & A_{234} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{6.8}$$

By performing operations and ordering terms they have:

$$(A_{111} - A_{131}x_1)x_0 + (A_{112} - A_{132}x_1)y_0 + (A_{113} - A_{133}x_1)z_0 = A_{134}x_1 - A_{114}x_1 \tag{6.9}$$

$$(A_{121} - A_{131}y_1)x_0 + (A_{122} - A_{132}y_1)y_0 + (A_{123} - A_{133}y_1)z_0 = A_{134}y_1 - A_{124}x_1 \tag{6.10}$$

$$(A_{211} - A_{231}x_2)x_0 + (A_{212} - A_{232}x_2)y_0 + (A_{213} - A_{233}x_1)z_0 = A_{234}x_2 - A_{214}x_1 \tag{6.11}$$

$$(A_{221} - A_{231}y_2)x_0 + (A_{222} - A_{232}y_2)y_0 + (A_{223} - A_{233}y_2)z_0 = A_{234}y_2 - A_{224}x_1 \tag{6.12}$$

Sorting coordinates as a matrix:

$$PV = F \tag{6.13}$$

Where each matrix is expresed:

$$\mathrm{P} = \begin{bmatrix} A_{111} - A_{131}x_1 & A_{112} - A_{132}x_1 & A_{113} - A_{133}x_1 \\ A_{121} - A_{131}y_1 & A_{122} - A_{132}y_1 & A_{123} - A_{133}y_1 \\ A_{211} - A_{231}x_2 & A_{212} - A_{232}x_2 & A_{213} - A_{233}x_2 \\ A_{221} - A_{231}y_2 & A_{222} - A_{232}y_2 & A_{223} - A_{233}y_2 \end{bmatrix} \; V = \begin{bmatrix} x_{p0} \\ y_{p0} \\ z_{p0} \end{bmatrix}$$

$$F = \begin{bmatrix} A_{134}x_{p1} - A_{114} \\ A_{134}y_{p1} - A_{124} \\ A_{234}x_{p2} - A_{214} \\ A_{234}y_{p2} - A_{224} \end{bmatrix}$$

For determination of object coordinates, that is, the matrix V according to linear algebra (6.13) is a normal equation satisfies $P^tPV = P^tF$ at which it can be determined that:

$$V = (P^tP)^{-1}P^tF \tag{6.14}$$

Before looking in detail some methods of structured light should emphasize that this process must be precise correspondence between the image point and the corresponding projected point. This can be seen in the idea of the equations (6.5), (6.6) and (6.14) in which the coordinates of the object P_0, can be determined from knowledge of the overall variables (which for all the projected points are equal) f_1, f_2 focal lengths, x_2, y_2, z_2 projector position relative to the camera, but in addition to the specific variables of each point (x_1, y_1), (x_2, y_2).

The correspondence problem is that with a single point there is no doubt how specific variables must be introduced into the equations. When someone have for example two points a and b is possible to confuse the image point by point with vice versa. This would cause the amounts are entered into the equations would be incorrect. That is, it must use the set of points:

$$P_{1a} = (x_{p1a}, y_{p1a}, 0), P_{2a} = (x_{p2a}, y_{p2a}, 0), P_{1b} = (x_{p1b}, y_{p1b}, 0), P_{2a} = (x_{p2b}, y_{p2b}, 0) \tag{6.15}$$

But if through carelessness or confusion, it evaluate z_{p0} using the pairs P_{1a} with P_{2b} and P_{1b} with P_{2a}, it is obvious that a major error. This apparently trivial, but it is likely to happen if one is not aware of the problem before. This is the problem of correspondence; it must always ensure that the expressions, pairs of image points P_{1j} correspond to the respective object point P_{2j}. This

is not merely a matter of numbers, but correct identification of each point to each point object image.

When a large number of points may be handled from a few dozen and can complicate the problem; is very common, however, have hundreds and few thousand points. In such a case it becomes relevant design methods to identify object points and their corresponding image points. This is the correspondence problem and is particularly difficult to do when the array object and image points are very symmetrical.

This is also important when the projection of the points of them to the hidden camera up or lost by the finite size of the object being measured. In the following figure, it see that the projector has seven points a, b, c, d, e, f and g. However, c and b points are hidden from the camera and the point g is lost in the distance as the object fails to be intersected by this ray. In the picture the dots appear a, d, e and f, however the lack of other information one would tend to assign the letters a, b, c and d or b, c, d, e also c, d, e, f to the points being all incorrect.

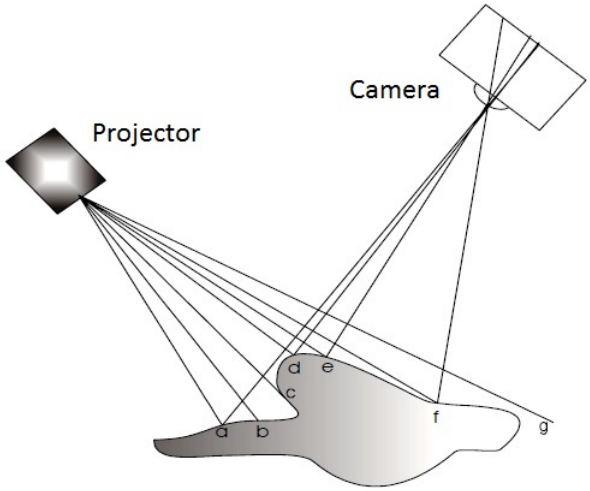


Figure 6.2: Concealments of points for a camera projected points on a rough surface

In the figure 6.3, it should be clear that in the absence of other indications, it is not possible to say which comes from each image point object.

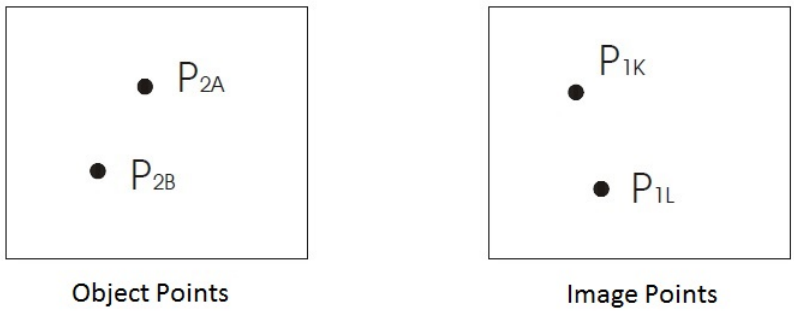


Figure 6.3: Diagram showing the difficulty of associating an object point to its corresponding image point without any indication for it

6.3 Image Acquisition Techniques 3D

The development of a system for 3D Vision requires solving a number of aspects or stages: recovery of three-dimensional structure of the scene, modeling and object representation, recognition and localization, and interpretation of the scene.

The different techniques for recovering the three-dimensional structure of the scene have specific characteristics at all levels of the visual interpretation process, from initial imaging, analysis and interpretation of it, every method requires both equipment specific algorithms.

6.3.1 Structured light

[28] Within the field of three-dimensional view, there are a number of techniques that are now used successfully in numerous industrial applications. Among them, is what is known as the Structured Light.

This type of system is characterized by a direct and active method. A direct method is characterized in that conclusions can be obtained by analyzing the data obtained directly from the images. As for active, keep in mind that this system uses a generator light system, which introduces a type of energy to the environment where the study was performed.

Structured light systems are based on studying the deformation of a light pattern to be intercepted by any object. This is the main problem with such tools as a type of concentrated light is needed at one point. As it would not be illuminated, any of the usual systems that are currently used, such as bulbs, fluorescent, etc., as are composed of waves of different frequencies causing the beam is diffused throughout the environment.

One of the best solutions is to use a laser beam. Due to its characteristics of coherence, divergence, and addressability, it behaves in a perfect light such systems. This is because the consistency causes all beam waves have the same frequency, and with his little divergence and high directivity allow it to be directing a laser beam at any point you want. Once you and the type of light to be used is known, it is necessary to choose a suitable pattern. The different solutions ranging from the use of points to grids with lasers of different colors. Hiring points means having to traverse the object across the surface taking a lot of points and some areas may lose it.

The use of a plane seems a better solution than the previous one. The plane illuminate a set of points with the same characteristics and meet the condition of a plane in space. Using a grid with different colored dots involves having the entire surface of the illuminated object and once the problem is to find the different points and situation. Of course, besides the light pattern, you need a camera that collects all the images of the deformation of the laser plane. The camera position in the set should be the one, which allows to obtain both the best resolution as prevent dark areas exist, ie, there are no areas of the object which are not illuminated by the laser.

Once it have decided on one or another pattern and corresponding camera will need to calibrate the different parameters. This step is the most important as for the coordinates of different points in the object is to employ the method of triangulation. Triangulation is to obtain the position of a point relative to the position of the camera and the plane. The figure below explains the method

of triangulation.

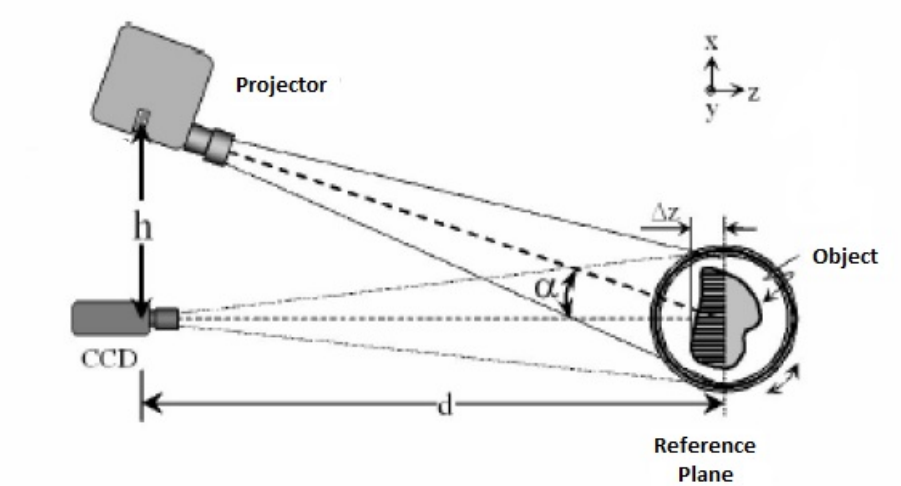


Figure 6.4: 3D Triangulation Scheme.

If the distance from the camera to an object point is known, that is the basis triangle, the distance between the camera and the laser, one side, and the angle of plane of the laser, it can see the three-dimensional coordinates of that point.

But for this it is necessary to know the positions in space from both the camera and the laser plane, so it will be necessary, a calibration process oth systems.

The projected patterns are designed so that each image pixel may be assigned a well-defined code. The projection techniques such patterns can be classified according to the coding strategy used, divided into three groups:

- Time multiplexing technique.
- Neighborhood spatial techniques.
- Direct encoding.

In our case, it consider only the technical calls time multiplexed one of the most widespread in commercial systems for three-dimensional structured light scanning.

Among the multiplexing techniques can be found:

6.3.1.1 Gray binary patterns and Coding Technique

Binary encoding [40] uses black and white stripes to form a sequence of projection patterns, so that each point on the surface of the object has a unique binary code which differs from any other code in different points. In general, patterns of stripes can encode 2^N. The following figure shows a pattern of 5 bits simplified projection displays. Once this sequence patterns are projected onto a static scene, there are 32 (2^5) single zones coded unique stripes. 3D (x, y, z) coordinates can be computed (based on the principle of triangulation) for all 32 points along each horizontal line, thus forming a complete picture of the 3D image.

The binary coding technique is very reliable and less sensitive to the characteristics of surfaces, as there are only binary values at each pixel. However, to achieve high spatial resolution, a large number of sequential patterns need to be projected. All objects in the scene must remain static.

In this case, two illumination levels are used only, coded 0 or 1, each pixel has its own coding pattern comprising a sequence of 0s or 1s. Such patterns are simple to encode and process.

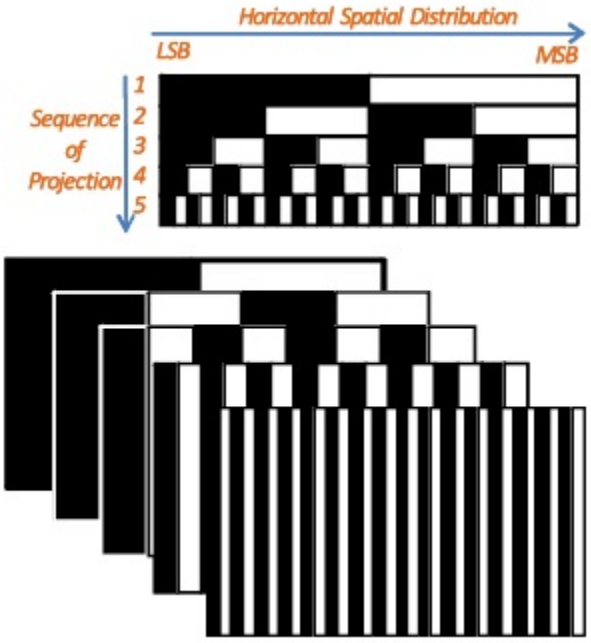


Figure 6.5: Projections of binary coded sequential patterns for 3D images

6.3.1.2 Gray level patterns

[41] To effectively reduce the number of patterns is necessary to obtain a high-resolution 3D patterns gray level are developed. For example, one can use different levels of intensity M (instead of only two in the binary code) to produce unique coding for projecting patterns. In this case, N may code patterns M^N slots. Each strip of code can be viewed as a point in a space on the basis of N, and each dimension has M different values. For example, if $N = 3$, and M is 4, then the total number of bands unique code is 64 (4^3). Compared to 64 stripes with a binary code, six patterns are needed. There is an optimization of the design of the patterns of binary and gray coding. The objective is to maximize some measure of the distance between all code words. For practical applications of 3D imaging, to distinguish adjacent stripes is important. An example of the coding patterns optimized gray level in the Hilbert space is shown in the following figure.

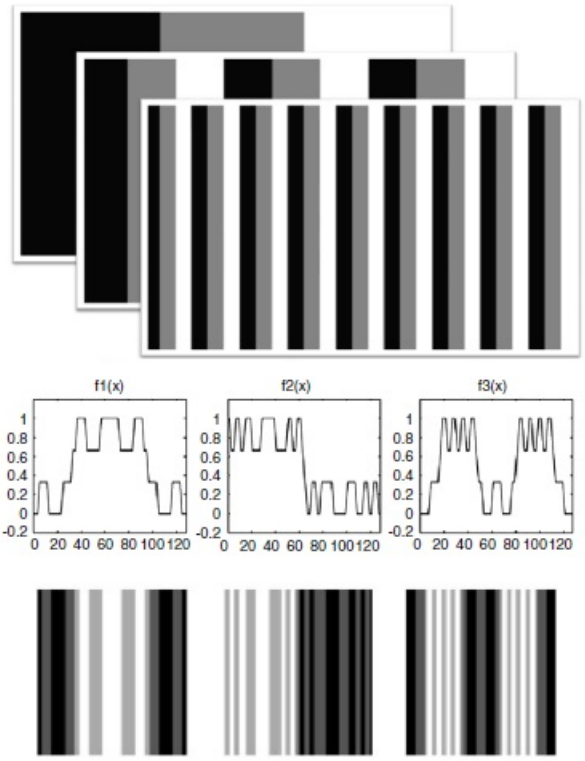


Figure 6.6: Coding for 3D gray level images and patterns gray level codes optimized

6.3.1.3 Phase Shift

[42] The phase shift method is a known fringe projection for 3D images of the surface. A set of sinusoidal patterns projected onto the object surface. The intensities for each pixel (x, y) of the three projected fringe patterns are described as::

$$I_1(x,y) = I_0(x,y) + I_{mod}(x,y)cos(\phi(x,y) - \theta)$$
$$I_2(x,y) = I_0(x,y) + I_{mod}(x,y)cos(\phi(x,y))$$
$$I_3(x,y) = I_0(x,y) + I_{mod}(x,y)cos(\phi(x,y) + \theta)$$

Where $I_1(x,y)$, $I_2(x,y)$ and $I_3(x,y)$ are the intensities of three patterns of stripesson, $I_0(x,y)$ is the DC component, $I_{mod}(x,y)$ is the signal amplitude modulation, $\phi(x,y)$ is the phase, and θ is the constant displacement angle.

Phase unwrapping is the process that turns the envelope to the absolute phase. The phase information (x, y) can be recovered (i.e., unwrapped) from the currents in the three fringe patterns:

$$\phi' = arctan\left[\sqrt{3}(\frac{I_1(x,y) - I_3(x,y)}{f_1x_2 - f_2x_1})\right] \tag{6.16}$$

The discontinuity of the arc tangent function in 2π can be eliminated by adding or subtracting multiples of 2π in $\phi(x,y)$ values:

$$\phi(x,y) = \phi'(x,y) + 2k\pi \tag{6.17}$$

Where k is an integer representing the projection period. Note that development methods provide only relative and development do not resolve to the absolute phase. 3D (x, y, z) coordinates can be calculated based on the difference between the measured phase $\phi(x,y)$ and the phase value from a reference plane. The following figure illustrates a simple case where:

$$\frac{Z}{L-Z} = \frac{d}{B} \tag{6.18}$$

$$Z = \frac{L-Z}{B}d \tag{6.19}$$

Simplifying the relationship leads to:

$$Z \approx \frac{L}{B}d\alpha\frac{L}{B}(\phi - \phi_0)$$

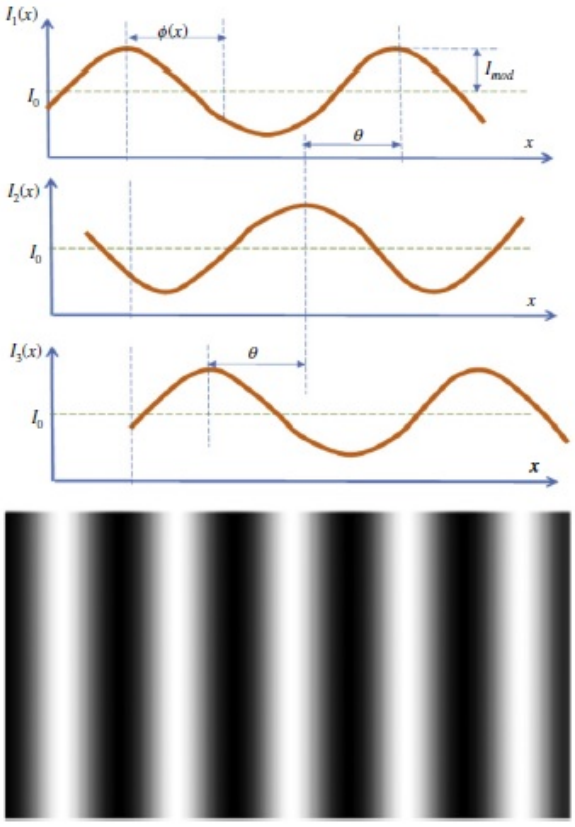


Figure 6.7: Phase shift with three projection patterns and an example of an image strip

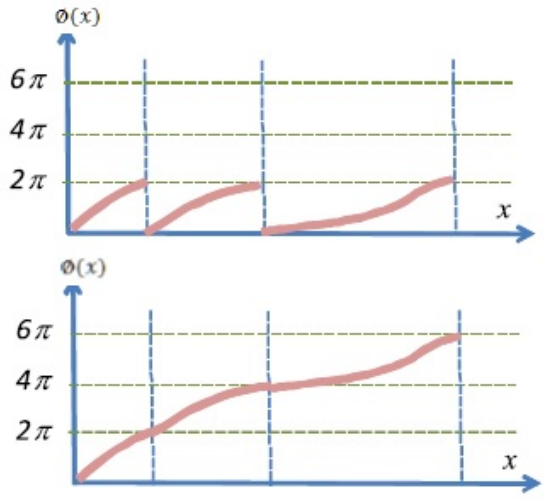


Figure 6.8: Illustration of the process of elimination of unwrapping

6.3.1.4 Hybrid Method: Phase Shift + Gray encoding types

[46] As it discussed above, there are two big problems with phase change techniques: develop methods provide only relative and do not resolve development for absolute phase. If both surfaces have a discontinuity of above 2π, then any method based on the unwinding will unfold properly in these two surfaces relative to each other. These problems, often called "ambiguity" can be solved by using a combination of the projection of gray code and phase shift techniques.

The following figure shows an example of the combination of gray code projection to a phase shift of 32 coding sequence stripes. The gray code determines the absolute range of phase unambiguously, while the phase shift provides subpixel resolution beyond the number of bands provided by the gray code. However, hybrid methods require a larger number of projections and are not well suited for 3D imaging of dynamic objects.

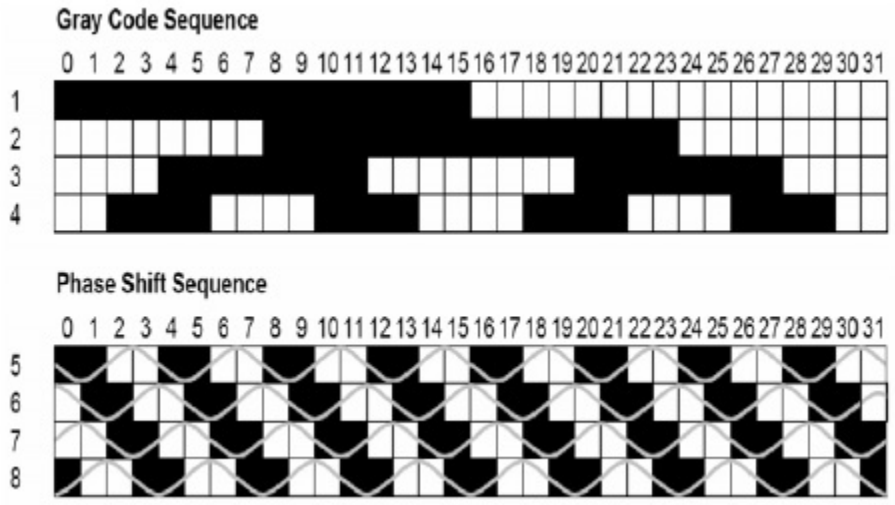


Figure 6.9: Combination Gray Code and Phase Shift

6.4 Implementation

For the implementation, an schemes for reconstruction was performed. According to this, it can follow an order and get a correct reconstruction of the object.

In the figure 6.11, it can be seen from the generation and projection patterns made by the user. Moreover, capturing the projected image using the HALCON pattern is appreciated. After the acquisition of the images, suitable algorithms is performed using HALCON commands for 3D object reconstruction.

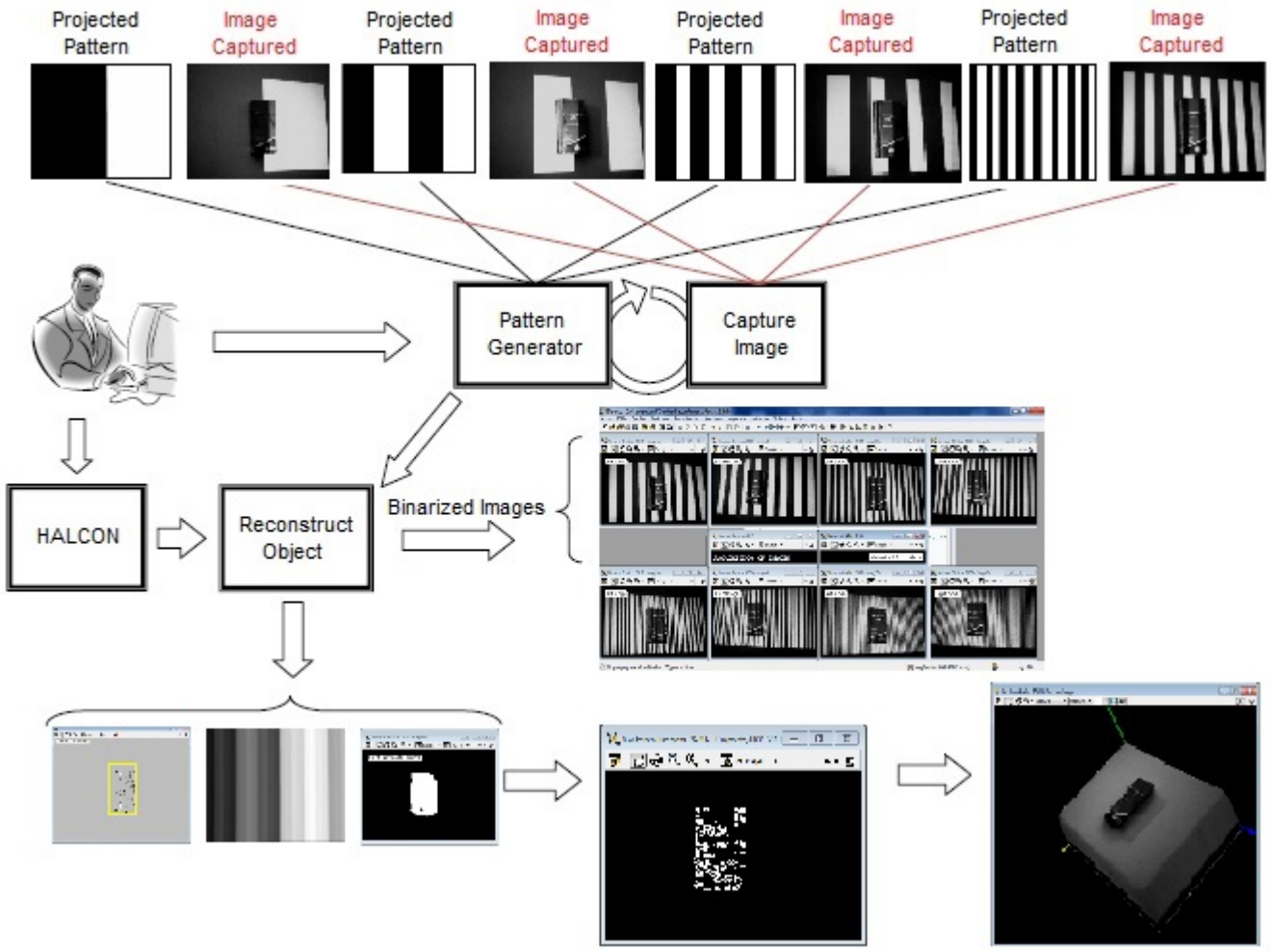


Figure 6.10: Reconstruction 3D Diagram

For programming in HALCON, the following figure 6.12 was used as a guide of the different ways how to derive a 3D object model that can be used:

6.4.1 Selection of 3D Reconstruction Method

Traditional optical 3D reconstruction methods have attracted special interest in industrial and medical applications due to its non-invasive characteristics, low implementation cost and excellent

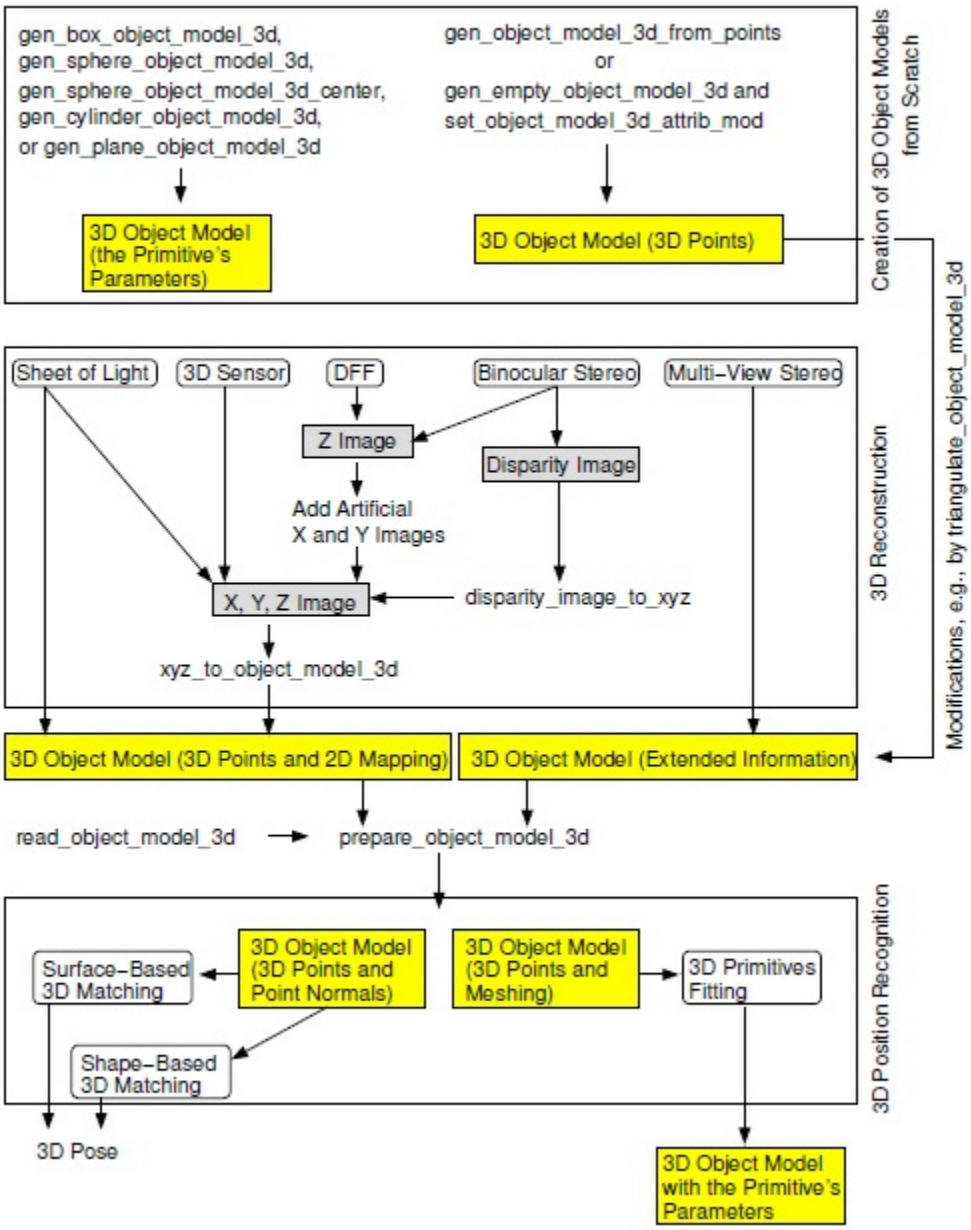


Figure 6.11: Overview on the 3D object model

resolutions. Are usually classified into passive or active. With passive methods is not necessary to control the light source, but the digital treatment for depth information is very delicate and requires high computational effort. Active methods using a radiation pattern simplifies the problem of measuring the topography.

When thinking about applying some reconstruction technique must take into account the actual characteristics of the objects to be studied, such as degree of light absorption and size. Also, there are objects that have abrupt changes in height and strong variations of the slope of the surface, these features are a problem in the rebuilding process for some optical techniques, from this point of view can be called complicated or discontinuous objects. For complicated discontinuous objects is common to use conventional fringe projection methods and triangulation.

Knowing that 3D reconstruction systems used in engineering are typically discontinuous macroscopic objects as defined above, is necessary to adapt or implement 3D reconstruction techniques for this type of objects. An alternative solution is to use a method that groups the important features of triangulation (resolve discontinuities) and fringe projection (two-dimensional analysis without scanning and easily implemented), the Gray code method meets these characteristics.

The gray code method is based on the projection of n patterns of different spatial frequency bands, which can describe the grouping 2^n different projection directions, where in space the reference plane corresponds to a binary code which can be identified in image digitized by the CCD. Thus, the topography of the object enters a lateral shift of words with respect to the reference plane. A digital method that can identify each code before and after placing the order, allow calculating the lateral shift and determine the appropriate height. Thus, when projecting 2D patterns is not required to calculate a sweep and shifts due to the topography of the object resolve discontinuities.

The gray code technique is easy to implement and flexible to the object size. However, drawbacks with low contrast images and is limited to static scenes because it is needed more than an outlet for reconstruction.

6.4.2 3D Reconstruction using code gray method

6.4.2.1 Optical System

The optical assembly is required to develop technique gray code comprises a projection system generates a sequence of binary patterns on the object surface, and an acquisition system for recording images of the patterns projected on reference plane before and after placing the order. This experimental setup is illustrated in the figure below, where P and C points represent the exit pupils projector and camera respectively, d is the distance between them, L is the distance to the reference plane measured in Z direction orthogonal coordinate system (X, Y,Z). The camera optical axis is perpendicular to the reference plane and the axis of the projector forms an angle to this, these axes intersect the R plane at point O, FW is the observation field width along the coordinate X on the reference plane.

The method performs a gray code coding the measurement space based on the projection, in different times n, n-dimensional binary patterns, formed by black and white stripes of different width for each pattern.

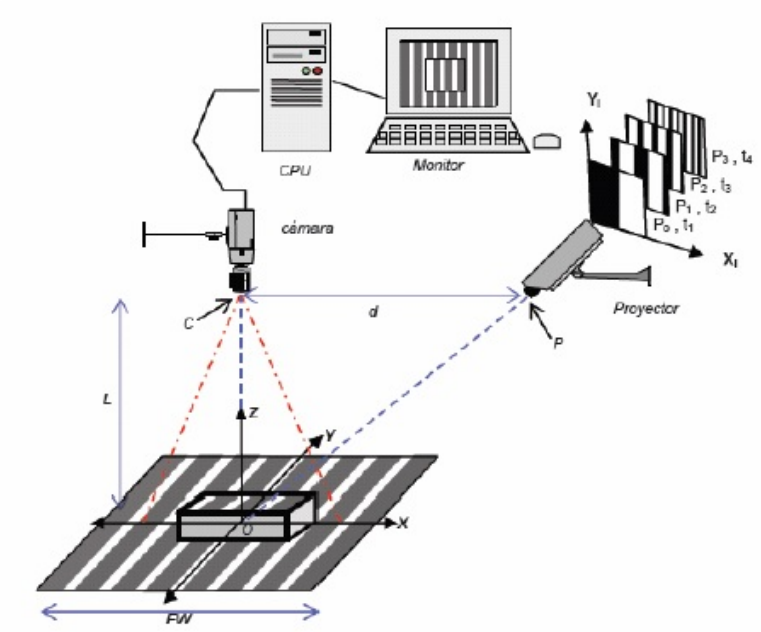


Figure 6.12: Experimental set of gray code method

To better understand the coding of space, the following is an example for 4 patterns, located in a system of coordinates(Xi, Yi, t) where Xi, Yi coordinates correspond to the object plane of the optical system shown projection and t is the time coordinate in which each pattern is projected. A white stripes are assigned the logic "1" and the black the logic "0". Thus, by choosing the value corresponding to one position (a, b) plane Xi, Yi for each pattern, a binary number is formed in the direction te, also called "binary word", which is repeated throughout of the coordinate Y. By plane (Yi, t) having the same word is called "projection direction". Thus, each position Xi has a different projection direction. Thus, for the position $Xi = a$ in the next figure, the projection direction can be identified by the decimal value (10) of the binary word (0 1 0 1). In the example, four patterns are projected obtaining $2^4 = 16$ binary words and they produce 16 coding projection directions around the space Xi. The binary words are easily displayed in gray levels as shown in the top of the figure.

In general, if n patterns are projected by grouping in the direction t can be described 2^n different projection directions, which direction is associated with a binary word. The patterns are constructed using the structure of mathematical logic where from the screening of n patterns "bits" and thus 2^n different combinations of "binary words of n bits" with logical values 0 and 1 are obtained .

		C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
$R=2^3$	P_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$R=2^2$	P_1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
$R=2^1$	P_2	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
$R=2^0$	P_3	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
Valor Decimal		0	1	3	2	6	7	5	4	12	13	15	14	10	11	9	8

Figure 6.13: Sequence of gray code for $n = 4$

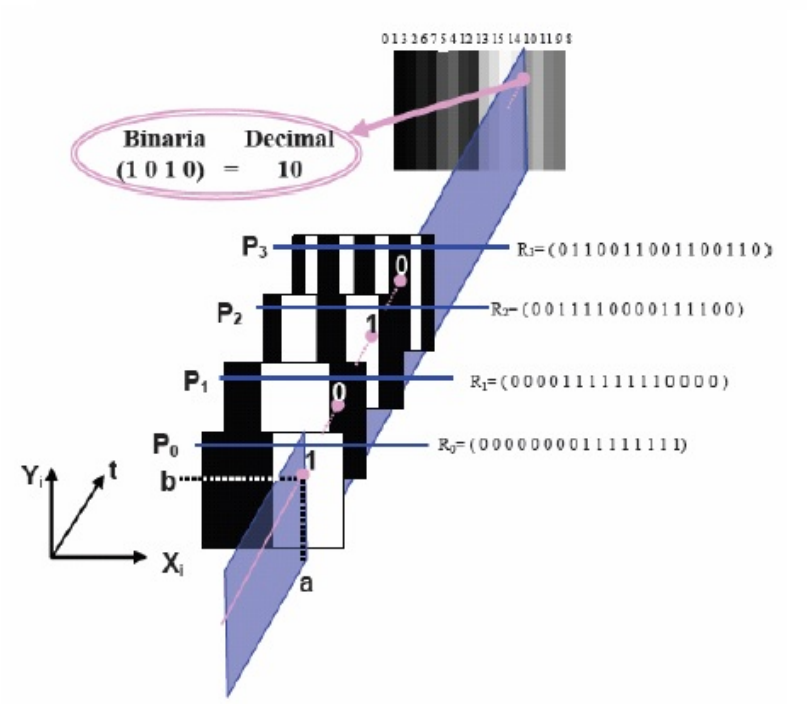


Figure 6.14: Combining patterns

$R0$, $R1$, $R2$, $R3$; code words of the most significant to the least significant(2^3,2^2,2^1,2^0). These values correspond to the sequence of values in binary representation of a row of the $P0$, $P1$, $P2$ and $P3$ patterns shown in the previous figure.

The columns $C0$, $C1$, ..., $C15$, resulting from the combination at time $P0$, $P1$, $P2$ and $P3$, are code words, each defining a different projection direction. In the last row of the image of the chart is the corresponding decimal value to each word in binary code.

In general, mathematical logic to define the figure of the previous table for a value of n patterns being obtained the 2^n projection directions. Rows allow designing different patterns of projection projection funció instantly. Likewise, columns allow to create different binary identify words reconstruction procedure.

6.4.3 Binarization and Management Shadows

The procedure of binarizacion allows to convert the image gray levels to binary values that correspond to significant levels in a function of the word binary. There are different strategies of binarizacion, the choice of the most appropriate strategy depends only on the characteristics of the acquired image in terms of the level of lighting, contrast and the presence of discontinuities in the surface of the object.

Traditionally in the binarizacion chooses a single threshold value, which is the same for all pixels of the image. The value of the most appropriate threshold is fixed, depending on the histogram of each image. This threshold value must be between the gray level of the larger image and the level of gray smaller. Considering the reflectivity of the surface becomes difficult to choose the

threshold when the density of stripes is high or when the highest gray level and the lowest gray level of the threshold range are close to each other.

The following figure shows the binarization for 4 different threshold values of an image of the pattern of stripes. Also in this figure, you can see the effect of choosing a threshold only depending on the contrast of the strips while for very high thresholds appear black regions also lost information. This shows that images of patterns of stripes with a global threshold value cannot be binarize correctly.

For this reason, it is necessary to implement a binarization method that acted locally. This way you can binarize the image independent of changes in ambient lighting conditions, the variation of the properties of surface reflectivity and shape of the object.

Although there are several techniques, the binarization method chosen for the development of the technique of gray code is by its robustness. That is, you must find where it ends and begins a black stripe white or vice versa. This method requires 2 images per pattern, an image of the direct pattern projection and a image of the of the reverse pattern projection for the location of the strips.Though an appropriate procedure binarizaión is chosen, there are in the process of reconstruction problems in regions where there arent stripes due to the "shadow" caused by the topography of the object and the type of lighting.

6.4.4 Implementation of project - Hardware

The implementation of the cameras and the projector on a metal structure was performed.

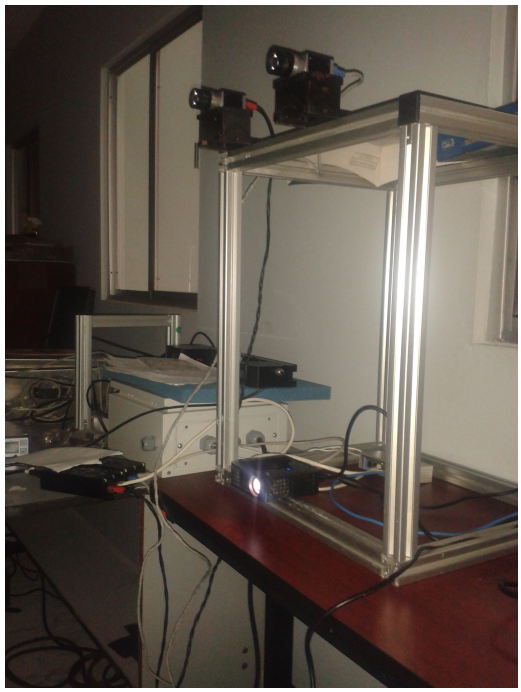


Figure 6.15: Implementation of cameras and projector-left side

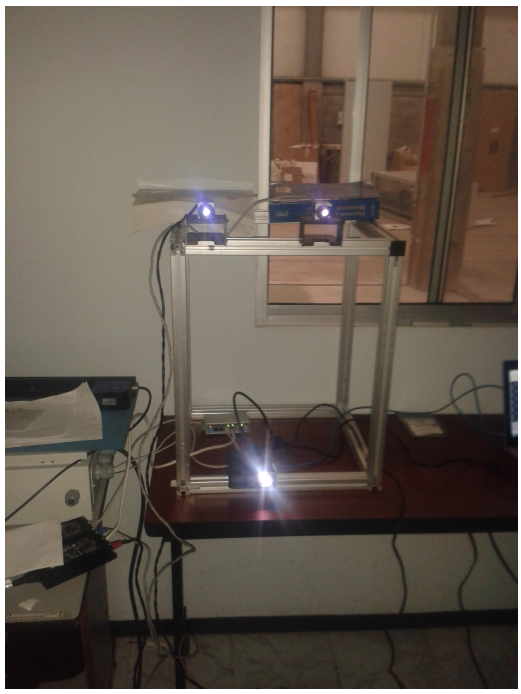


Figure 6.16: Implementation of cameras and projector-frontal side

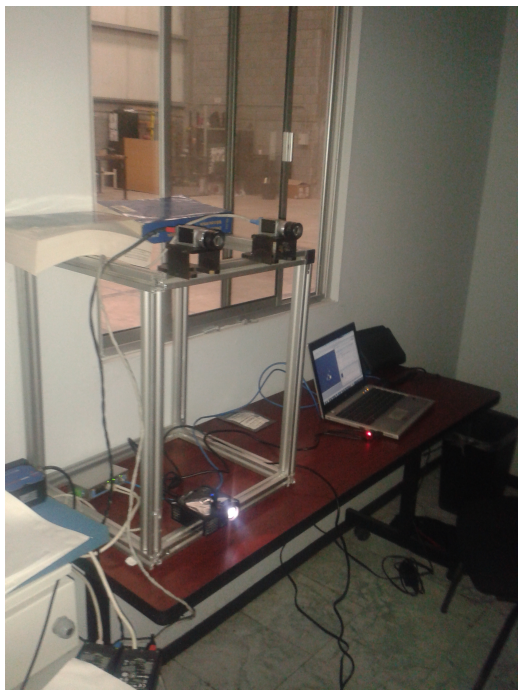


Figure 6.17: Implementation of cameras and projector-right side

Figure 6.18: Object with the projected pattern

Another implementation of the cameras and the projector.

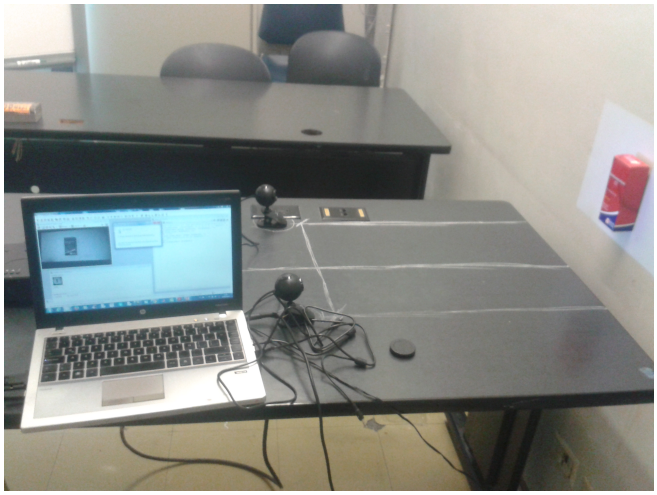


Figure 6.19: Implementation of the cameras and the projector-left side

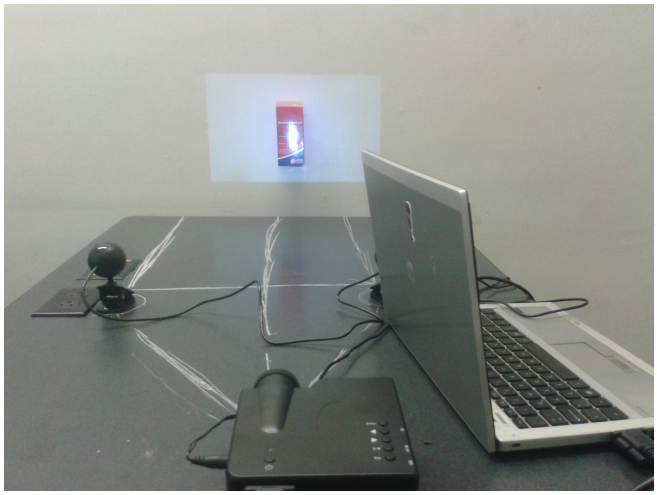


Figure 6.20: Implementation of the cameras and the projector-back side

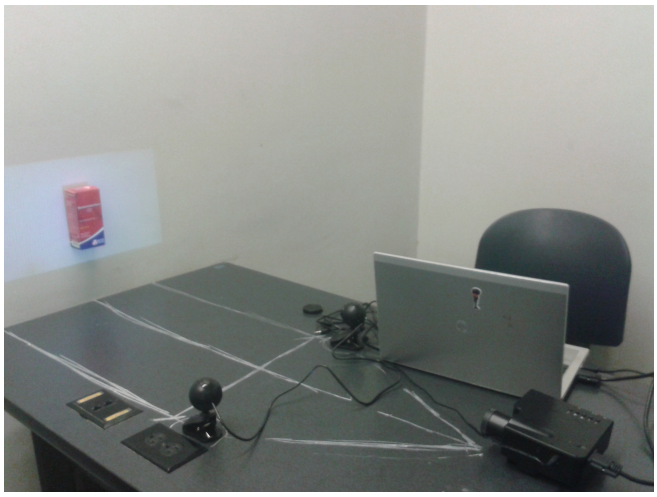


Figure 6.21: Implementation of the cameras and the projector-right side

6.4.4.1 Acquisition of images

It proceeded to the acquisition of the images for the cameras (right camera and left camera). Pictures of objects without pattern and with pattern projected on the object were obtained.

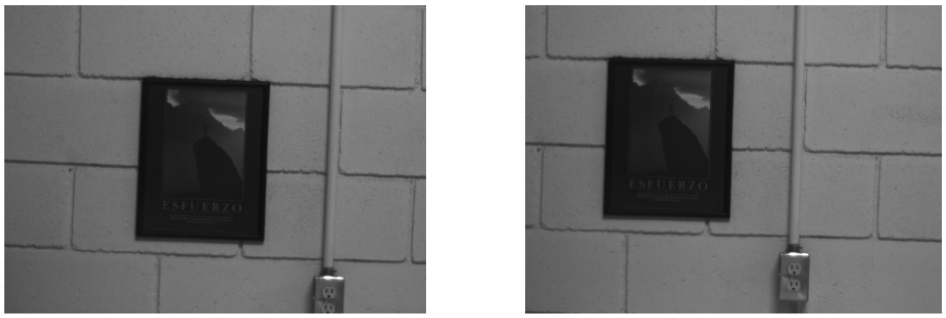


Figure 6.22: Acquisition of images without pattern of line - First object

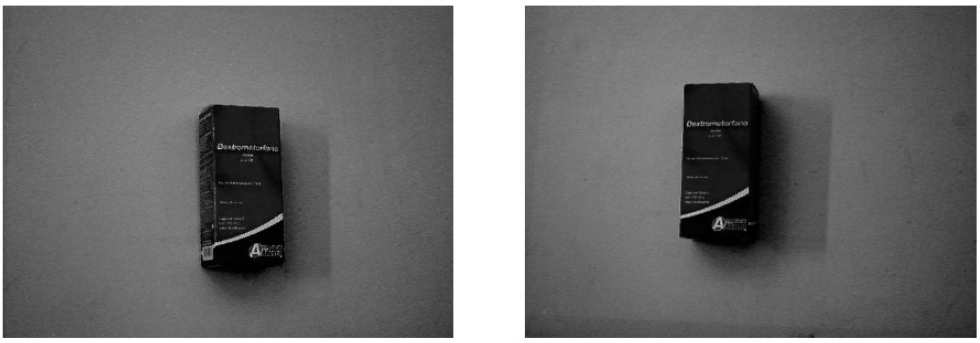


Figure 6.23: Acquisition of images without pattern of line - Second object

The patterns used in the project are:

6.4.4.1.1 Pattern of Lines

Patterns made on the basis of a lines, allowing to detect the corners. Furthermore, it was from lowest to highest number of quadrilaterals in black and white color. The patterns were performed in the Power Point that comes in the Office.

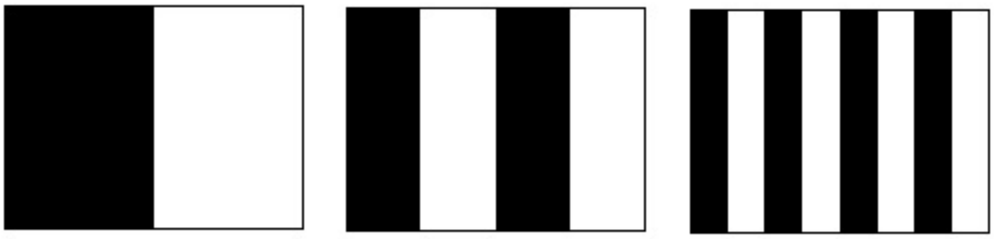


Figure 6.24: Pattern of line R=2^0 R=2^1 R=2^2

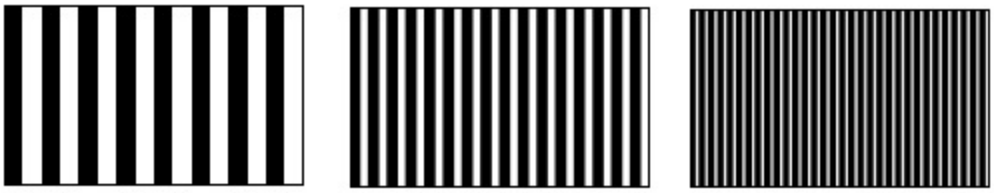


Figure 6.25: Pattern of line R=2^3 R=2^4 R=2^5

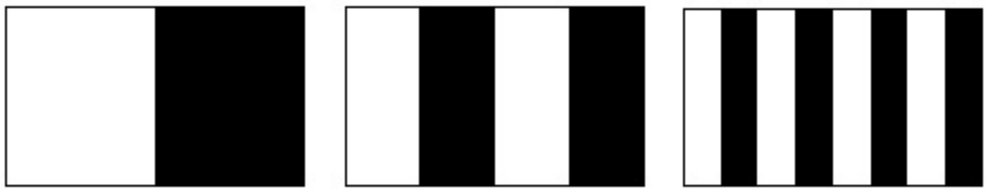


Figure 6.26: Reverse pattern of line R=2^0 R=2^1 R=2^2

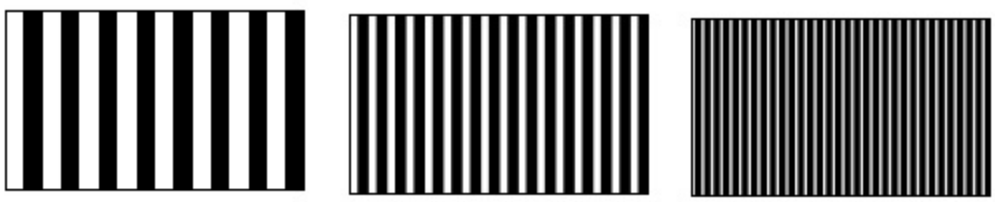
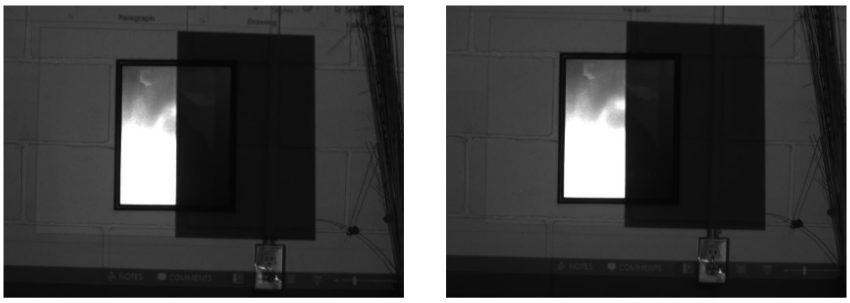


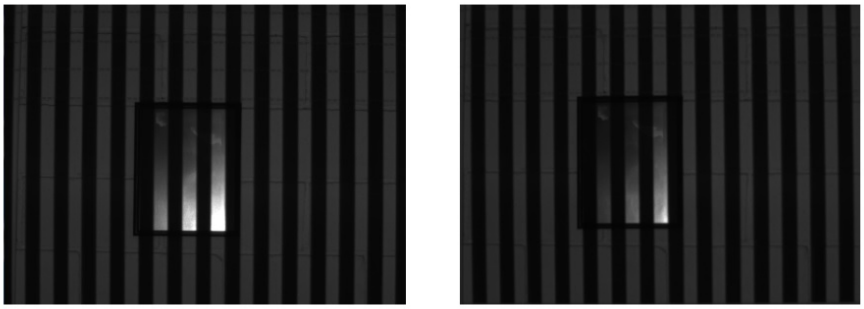
Figure 6.27: Reverse pattern of line R=2^3 R=2^4 R=2^5

6.4.4.1.2 Acquisition of images using pattern of line



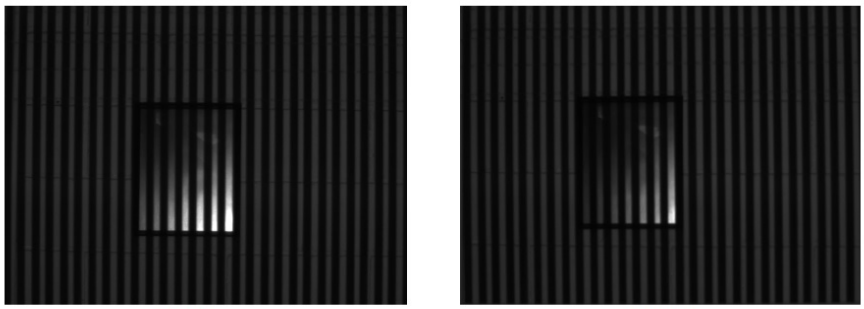
a) Left b) Right

Figure 6.28: Acquisition of images using pattern of line R=2^0



a)Left b) Right

Figure 6.29: Acquisition of images using pattern of line R=2^0



a) Left b) Right

Figure 6.30: Acquisition of images using pattern of line R=2^4

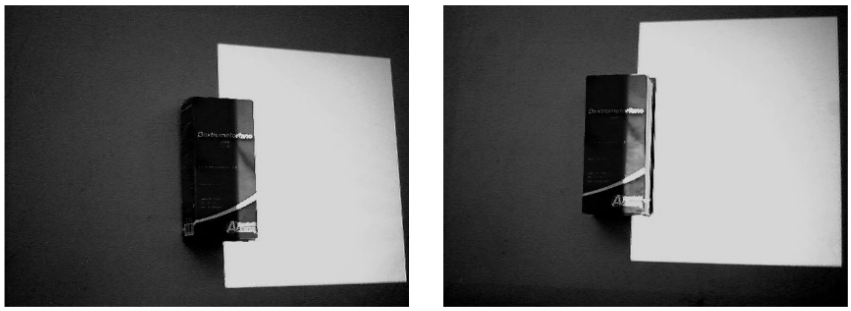


Figure 6.31: Acquisition of images using pattern of line R=2^5

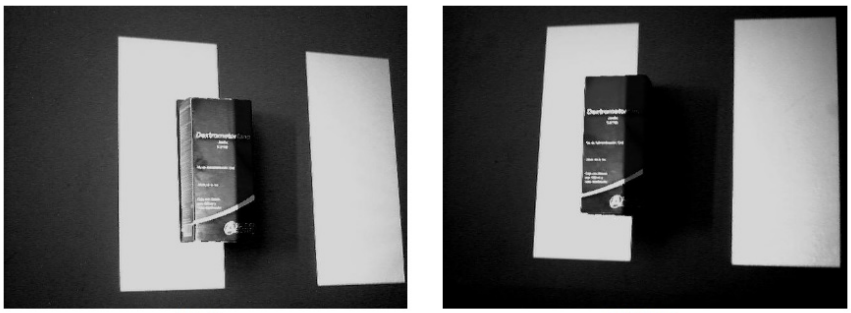


Figure 6.32: Acquisition of images using pattern of line R=2^1

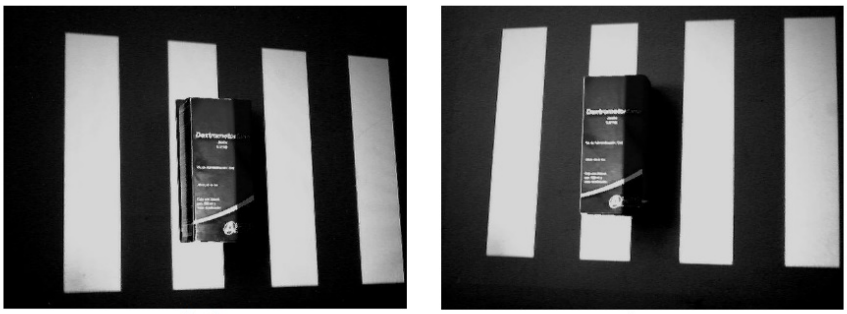


Figure 6.33: Acquisition of images using pattern of line R=2^2

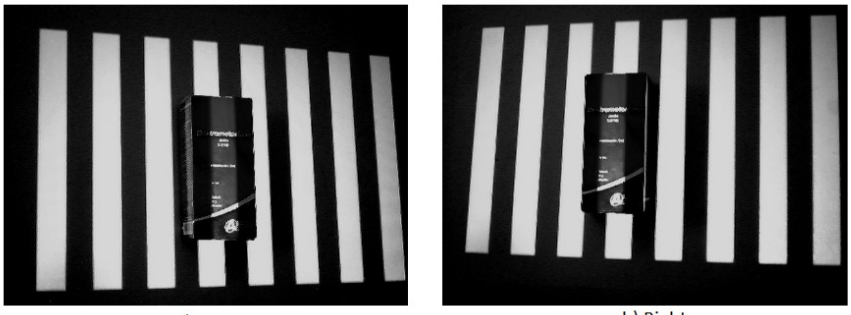


Figure 6.34: Acquisition of images using pattern of line R=2^3

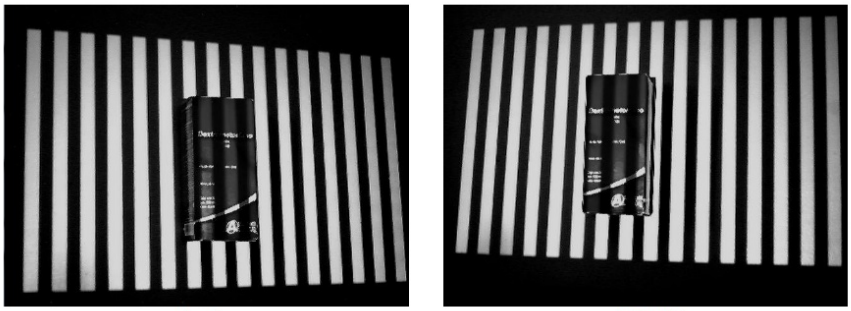


Figure 6.35: Acquisition of images using pattern of line R=2^4

6.4.5 Implementation of project - Software

It begins with the presentation of the project.

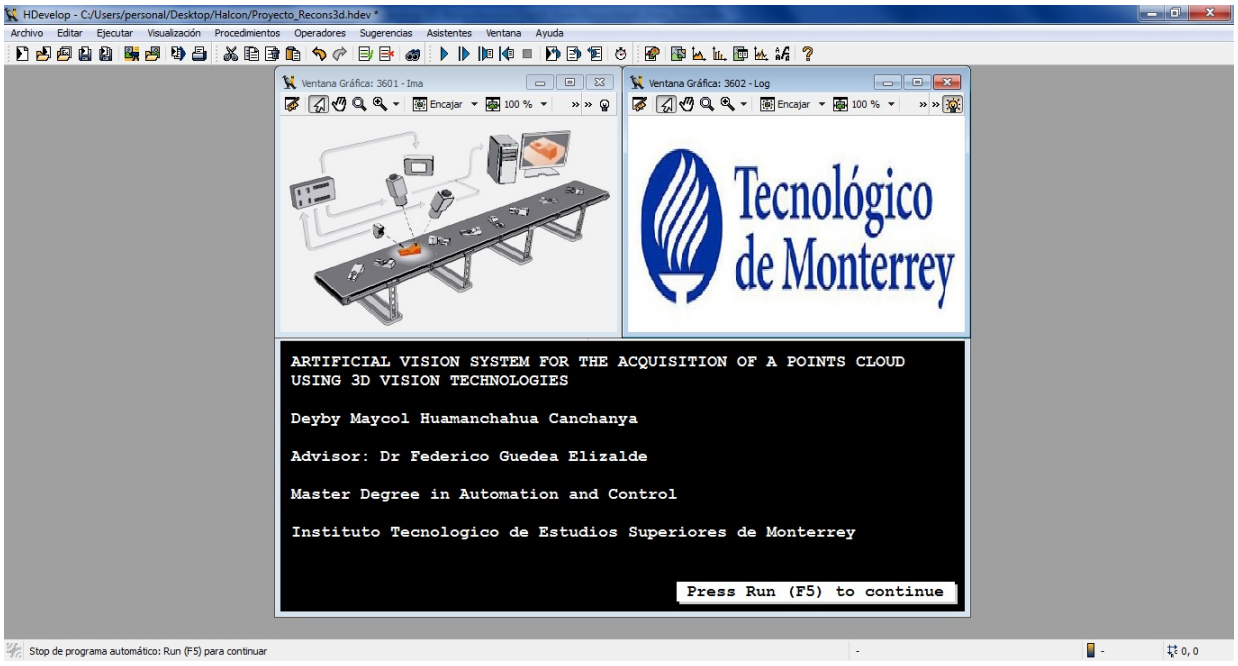


Figure 6.36: Presentation of the project

6.4.5.1 Detection of points of interest

In image processing, the concept of feature detection means obtaining image information. The resulting features are subsets of the domain of the image, often in the form of isolated points, continuous curves or connected regions.

Although there is no exact definition of what constitutes a feature, can be defined as an interesting part of an image (interest point). Are often used as a starting point for many computer vision algorithms.

Because the algorithm is based on these characteristics, the algorithm will be as good as its

detector. Another thing to keep in mind is that a good feature detector should detect same feature in two or more different images of the same scene, ie, a property must have a feature detector is the repeatability or recurrence.

Other properties that should have a feature detector is accurate (because it must detect feature in the correct pixel) and stability (this should detect the feature after the image has suffered some kind of geometric transformation such as rotation or scale).

Several features detectors already developed, that vary in the type of characteristic to be detected, the computational complexity and repeatability. These detectors can be divided into several groups being the most important detectors of corners, edges and straight lines.

6.4.5.2 Detector of corners

Corners in images represent useful information and are very important to describe objects for recognition and identification.

A corner can be defined as the intersection of two edges. A corner can also be defined as a point at which two edges with different main directions and the area near the point. Another way to define one corner is like an area where the intensity variations in the x and y directions are large or in other words a region where the intensity varies in both directions.

A corner detector requires certain conditions. First, all true corners must be detected and no false corner must be detected . Second, the corners must be detected properly located. In addition, the detector must have repeatability (stability), it must be robust to noise and be computationally efficient.

For detection of corners in pictures exist numerous detectors, of which here are going to try a few: Sojka, Lepetit, Foerstner, Harris and Harris Binomial.

6.4.5.2.1 Sojka Detector of corners

Sojka [48] defines a corner as the point of intersection of two straight, non-collinear gray value edges. To decide whether a point of the input image $Image$ is a corner or not, a neighborhood of $MaskSize$(Filter Size) x $MaskSize$ points is inspected. Only those image regions that are relevant for the decision are considered. Pixels with a magnitude of the gradient of less than $MinGrad$(Threshold for the magnitude of the gradient) are ignored from the outset.

Furthermore, only those of the remaining points are used that belong to one of the two gray value edges that form the corner. For this, the so called Apparentness is calculated, which is an indicator of the probability that the examined point actually is a corner point. Essentially, it is determined by the number of relevant points and their gradients. A point can only be accepted as a corner when its Apparentness is at least $MinApparentness$(Threshold for Apparentness). Typical values of $MinApparentness$ should range in the region of a few multiples of $MinGrad$.

To calculate the Apparentness, each mask point is weighted according to two criteria: First, the influence of a mask point is weighted with a Gaussian of size $SigmaW$(Sigma of the weight

function according to the distance to the corner candidate) according to its distance from the possible corner point. $SigmaW$ should be roughly a quarter to the half of $MaskSize$ to obtain a reasonable proportion of the size of the weighting function to the mask size. Secondly, the distance of the point from the (assumed) ideal gray value edge is estimated and the point is weighted with a Gaussian of size $SigmaD$ according to that distance. I.e., pixels that (due to the discretization of the input image) lie farther from the ideal gray value edge have less influence on the result than pixels with a smaller distance. Typically, it is not necessary to modify the default value 0.75 of $SigmaD$.

As a further criterion, the angle is calculated, by which the gray value edges change their direction in the corner point. A point can only be accepted as a corner when this angle is greater than $MinAngle$(Threshold for the direction change in a corner point (radians)).

The position of the detected corner points is returned in (Row, $Column$). Row(Row coordinates of the detected corner points) and $Column$(Column coordinates of the detected corner points) are calculated with subpixel accuracy if $Subpix$(Subpixel precise calculation of the corner points) is 'true'. They are calculated only with pixel accuracy if $Subpix$ is 'false'.

6.4.5.2.2 Lepetit Detector of corners

Lepetit extracts points of interest like corners or blob-like structures from $Image$(Input Image). The $Image$ is first smoothed with a median of size 3x3. Then, all the gray values on a circle with radius $Radius$(Radius of the circle) around an interest point candidate (m) are examined. The absolute differences of two diagonally opposed gray values ($m1$,$m2$) on the circle to the central pixel m is computed. At least one of these differences has to be larger than $MinCheckNeighborDiff$(Threshold of grayvalue difference to each circle point). All diagonally opposed pixels on the circle must fulfill that condition. To suppress detection of points at edges that have a small curvature (aliasing), it is possible to compute $CheckNeighbor$(Number of checked neighbors on the circle) further differences of circle point neighbors of $m1$ and $m2$ to the center, that as well fulfill the above criteria. By computing all gray value differences of the circle points to the center, a mean gray value difference is determined. That value has to be larger than $MinScore$(Threshold of grayvalue difference to all circle points) and allows to restrict the results to points with high contrast. By computing the score of all eight neighbors of m, it is possible to fit a quadratic equation to that. The Lepetit detector can especially be used for very fast interest point extraction.

6.4.5.2.3 Foerstner Detector of corners

Foerstner [66] [67] extracts significant points from an image. Significant points are points that differ from their neighborhood, i.e., points where the image function changes in two dimensions. These changes occur on the one hand at the intersection of image edges (called junction points), and on the other hand at places where color or brightness differs from the surrounding neighborhood (called area points).

The point extraction takes place in two steps:

In the first step the point regions, i.e., the inhomogeneous, isotropic regions, are extracted from

the image.

In the second step, two optimization functions are calculated for the resulting points. Essentially, these optimization functions average for each point the distances to the edge directions (for junction points) and the gradient directions (for area points) within an observation window around the point.

6.4.5.2.4 Harris Detector of corners

Harris detector is one of the most popular corners detectors due to its invariance to changes in rotation, scaling, noise and lighting. Harris method is based on matrix calculation of estimated by the first-order derivatives correlation for each pixel.

Harris [68] extracts points of interest from an image. The Harris operator is based upon the smoothed matrix.

$$M = G * \begin{bmatrix} \sum_{c=1}^{n} I_x, c^2 & \sum_{c=1}^{n} I_x, c; I_y, c \\ \sum_{c=1}^{n} I_x, c; I_y, c & \sum_{c=1}^{n} I_y, c^2 \end{bmatrix} \tag{6.20}$$

where G stands for a Gaussian smoothing of size $SigmaSmooth$(Amount of smoothing used for the integration of the gradients) and I_x,c and I_y,c are the first derivatives of each image channel, computed with Gaussian derivatives of size $SigmaGrad$(Amount of smoothing used for the calculation of the gradient).

If necessary, they can be restricted to points with a minimum filter response of $Threshold$(Minimum filter response for the points). The coordinates of the points are calculated with subpixel accuracy.

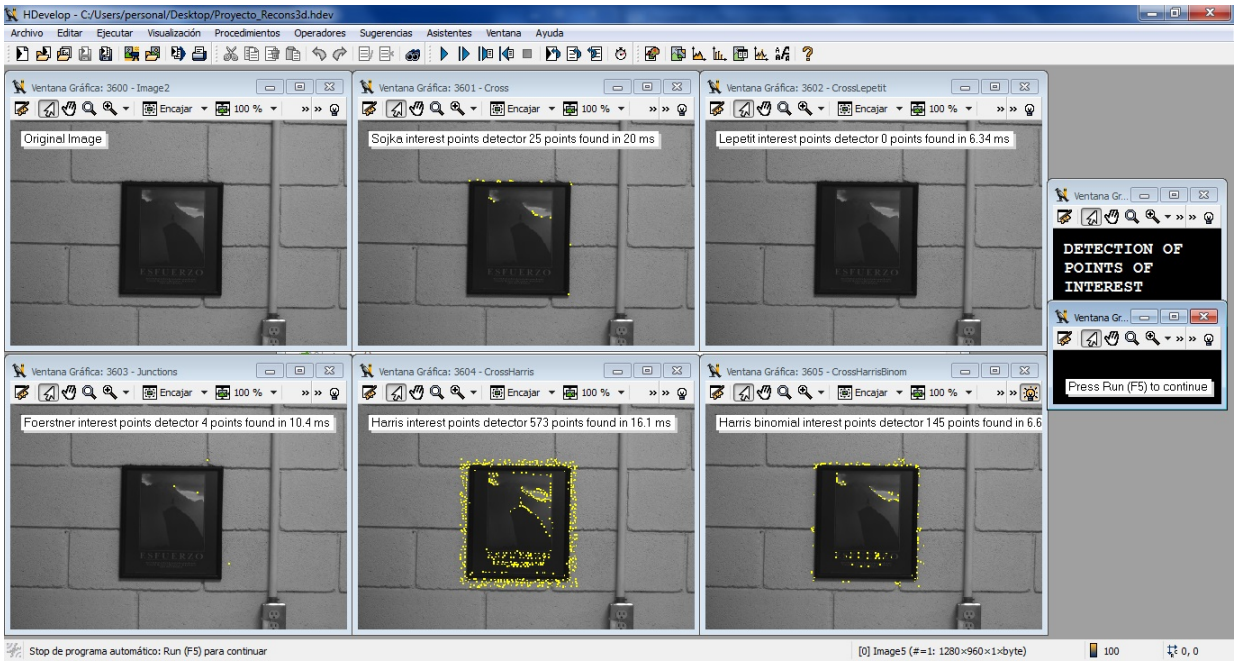


Figure 6.37: Detection of points of interest

6.4.5.3 Detector of edges

Edge detection is key in the three-dimensional reconstruction because it can extract important image information, such as the shapes of the objects that compose it. The edges indicate where the objects, their shape, their size, are and also provide information on its texture.

The edges are the points where there is a border between two regions of the image. In general, these can be of any shape; and may include linkages, discontinuities and extreme. In practice, the border can be defined as sets of points in the image that have a high gradient (large variations in intensity). For detection of edges in images there are many methods which will be defined here only some: Robert, Sobel, Prewitt, Frei-Chen, Robinson, Kirsch and Canny.

6.4.5.3.1 Prewitt Detector of edges

Prewitt calculates an approximation of the first derivative of the image data and is used as an edge detector. Convolution masks used for edge detection by Prewitt are:

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

Figure 6.38: Convolution masks operator of Prewitt

6.4.5.3.2 Robert Detector of edges

Robert operator makes a simple and quick measure of the gradient of the image. This operator consists of two 2x2 convolution masks. One of the masks is the 90 rotation of the other.

+1	0
0	1

Gx

0	+1
1	0

Gy

Figure 6.39: Convolution masks operator of Robert

6.4.5.3.3 Sobel Detector of edges

The Sobel operator uses two 3x3 convolution masks, one being the other rotated 90.

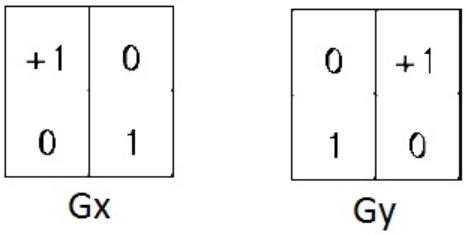


Figure 6.40: Convolution masks operator of Sobel

6.4.5.3.4 Canny Detector of edges

The Canny algorithm is also known as the optimal edge detector. Canny followed a series of criteria [69] that all edges must be detected and false edges should not be detected. Furthermore the edge points should be well located, ie the distance between the edge pixels as detected by the detector and the actual edge should be minimal. Another criterion was that there was only one answer to the same edge.

Using these criteria for the Canny detector: first, the image is smoothed to remove noise. Then the gradient of the image to determine areas where there are rapid changes in intensity is calculated. If the pixel is not a local maximum deletes it.

A process of eliminating hysteresis is used for the remaining pixels. With two thresholds, so that if it is less than the first threshold, the pixel is not considered edge and if greater than the second threshold is marked as edge. If the pixel is between two values, it is not marked as edge unless there is a path between this pixel and a pixel more than the second threshold gradient.

6.4.5.3.5 Robinson Detector of edges

Robinson calculates an approximation of the first derivative of the image data and is used as an edge detector. In Robinson operator, the following four of the originally proposed eight 3x3 filter masks are convolved with the image. The other four masks are obtained by a multiplication by -1.

Figure 6.41: Convolution masks operator of Robinson

6.4.5.3.6 Frei-Chen Detector of edges

Frei calculates an approximation of the first derivative of the image data and is used as an edge detector. The filter is based on the following filter masks:

```
A =
        1        sqrt(2)         1
        0           0            0
       -1       -sqrt(2)        -1

B =
        1           0           -1
     sqrt(2)        0        -sqrt(2)
        1           0           -1
```

Figure 6.42: Convolution masks operator of Frei-Chen

6.4.5.3.7 Kirsch Detector of edges

Kirsch calculates an approximation of the first derivative of the image data and is used as an edge detector. The filter is based on the following filter masks:

```
-3  -3   5      -3   5   5       5   5   5       5   5  -3
-3   0   5      -3   0   5      -3   0  -3       5   0  -3
-3  -3   5      -3  -3  -3      -3  -3  -3      -3  -3  -3

 5  -3  -3      -3  -3  -3      -3  -3  -3      -3  -3  -3
 5   0  -3       5   0  -3      -3   0  -3      -3   0   5
 5  -3  -3       5   5  -3       5   5   5      -3   5   5
```

Figure 6.43: Convolution masks operator of Kirsch

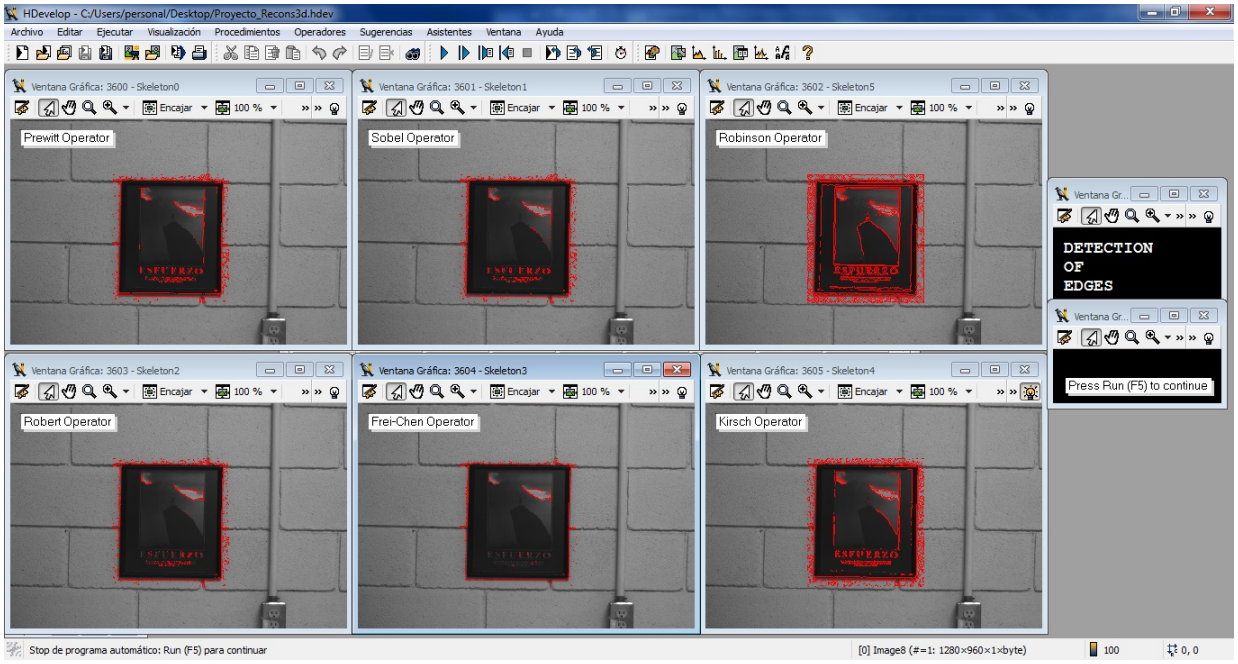


Figure 6.44: Detection of edges

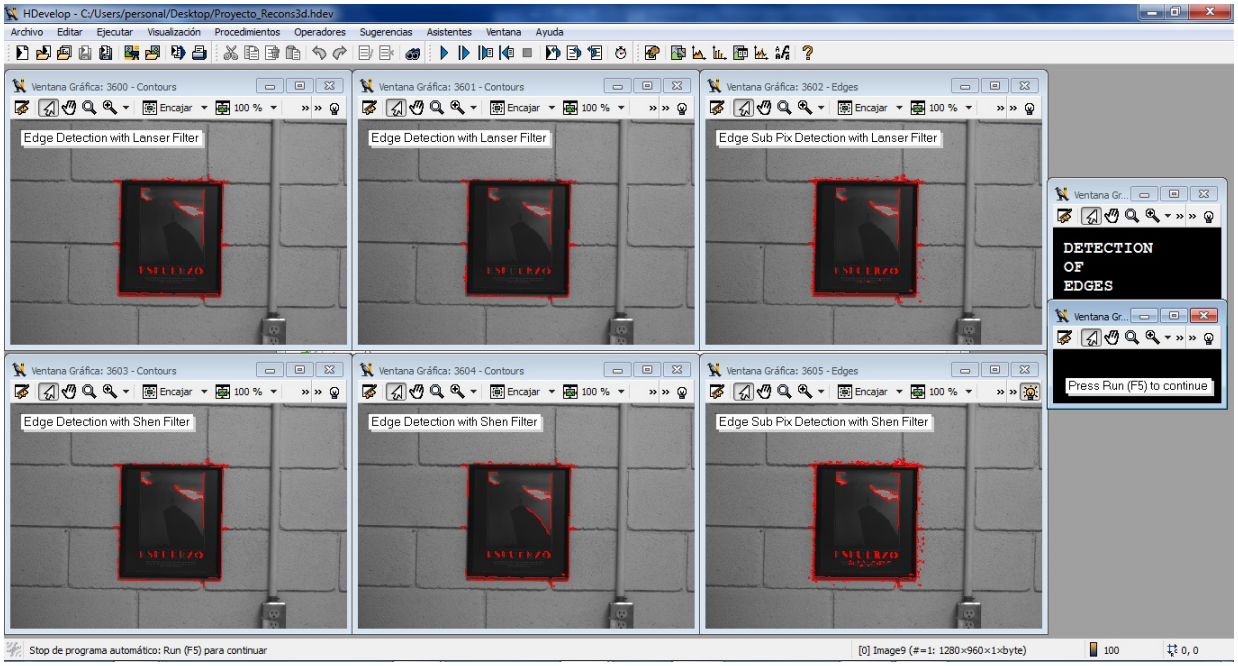


Figure 6.45: Detection of edges

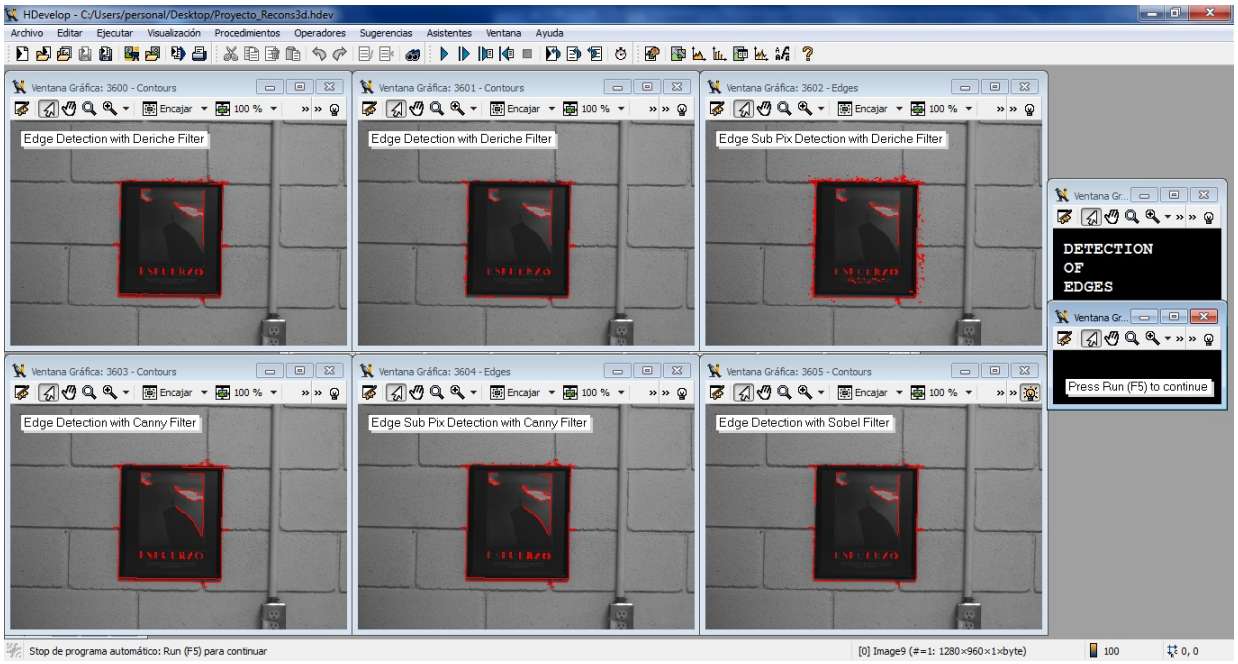


Figure 6.46: Detection of edges

6.4.5.4 Detection of straight lines

To detect straight lines in images, there arent variety of algorithms such for edges and lines [70]. The Hough transform to detect straight lines in a simple manner, so that it has proceeded to the implementation of this algorithm.

6.4.5.4.1 Hough transform

The Hough Transform is an algorithm used in pattern recognition in images to find certain forms within an image, such as lines, circles, etc. The simplest version is to find straight lines. Hough Transform exposed in [71] was proposed by Paul Hough in 1962 and patented by IBM. The algorithm uses the full set of the image, making it robust to the incomplete and contaminated data.

For execution requires a binary image in which edges have previously selected. If you consider a pixel of coordinates (x_i, y_i), this being an edge pixel image, there will be endless lines of the form:

$$y_i = ax_i + b \tag{6.21}$$

All these lines are defined by infinite values of a and b. By varying the parameter from $-\infty$ to $+\infty$ infinite values of b is obtained. Therefore, if two pixels belong to the same line representation is the intersection of two lines.

Hough transform applies this concept to the location of straight lines in the image. Cartesian space is discretized into intervals $[a_{min}, a_{max}]$ and $[b_{min}, b_{max}]$, creating a grid cell accumulation. For each pixel, considered edge, is walking range to obtain the values of b. For each value of b will

put a vote in the cell. This operation is done with all the pixels labeled as edges. At the end, those cells with the most votes will indicate the presence of lines in the image, whose models correspond to the coordinates of the cell.

However, Cartesian space is not ideal, because the ranges of a and b are not limited. Therefore, a representation is done in polar coordinates:

$$\rho = xcos\theta + ysin\theta \tag{6.22}$$

Where ρ is the length of a normal from the origin to the line and θ is the angle of ρ with respect to the x axis, θ is limited to the range of [0,π]. Note that the representation of a point in the parameter space is not a straight but a sinusoidal.

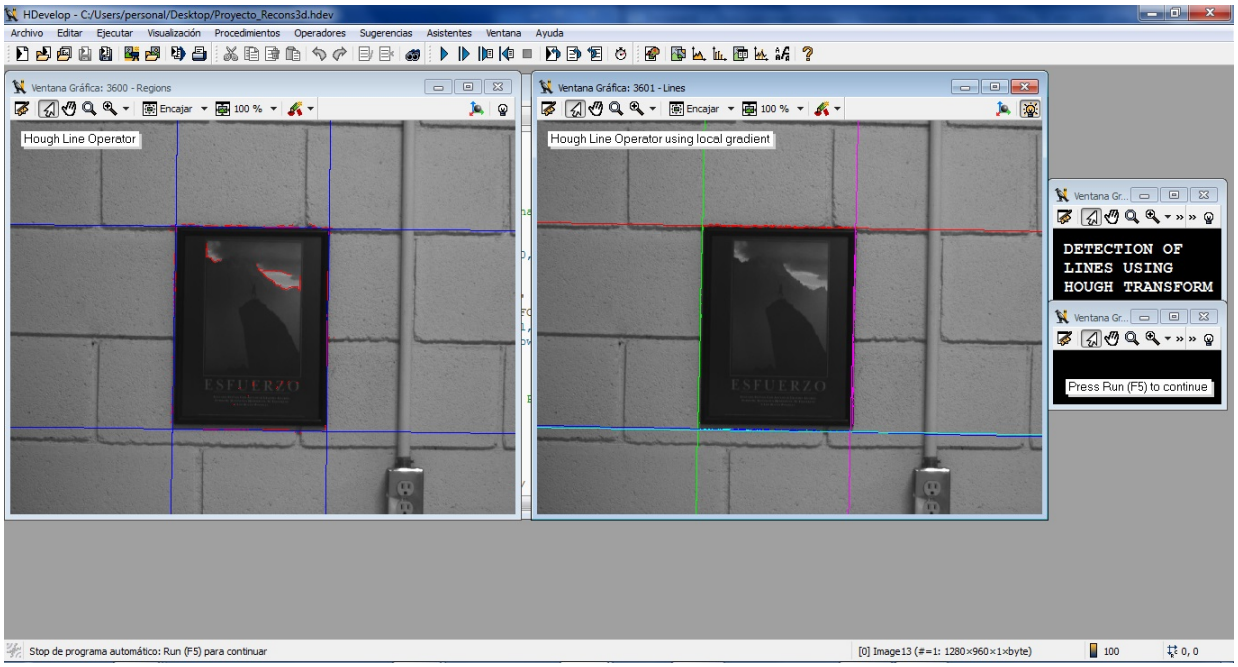


Figure 6.47: Detection of Straight Lines

6.4.5.5 Segmentation

Segmenting images is one of the most used and useful in processing and image analysis techniques. It is used to distinguish an object from the background or to divide an image into regions with some unrelated.

In general, image interpretation systems operate on data from a previous segmentation step so that the correct interpretation of information depends largely on the results of the segmentation.

As mentioned, this is one of the first steps in recognizing objects but is also one of the most problematic of machine vision. Depending on the desired target image and there are many segmentation techniques. A classification of these techniques could be: based on thresholds, contours, regions and groups (clustering) techniques as mentioned in [72].

6.4.5.5.1 Segmentation using Region

Segments images into regions of the same intensity, rastered into rectangles of size *Row* * *Column*. In order to decide whether two adjacent rectangles belong to the same region only the gray value of their center points is used. If the gray value difference is less then or equal to *Tolerance* (Points with a gray value difference less then or equal to tolerance are accumulated into the same object) the rectangles are merged into one region.

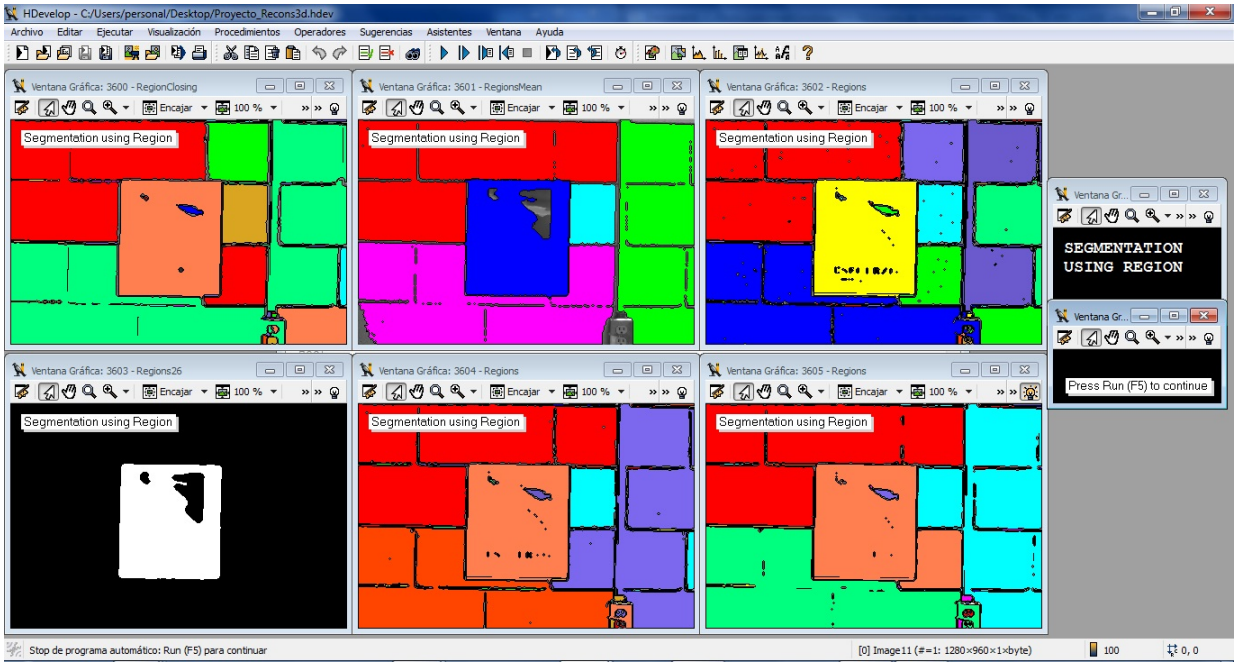


Figure 6.48: Segmentation using Region

6.4.5.5.2 Segmentation using Contour

Segments the input contours *Contours* (Segmented contours) into lines. First, approximates the input contours by polygons. With this, the contours are oversegmented in curved areas. After this, neighboring line segments are substituted by circular or elliptic arcs, respectively, if the contour can be approximated better by an arc. The input contours are first smoothed. This can be necessary to prevent very short segments in the polygonal approximation and to achieve a more robust fit with circular or elliptic arcs, because the smoothing suppresses outliers on the contours.

After this, circular or elliptic arcs are fit into neighboring line segments. If the maximum distance of the resulting arc to the contour is smaller than the maximum distance of the two line segments, the two line segments are replaced with the arc. This procedure is iterated until no more changes occur.

Later, the parts of the contour that are still approximated by line segments are again segmented with a polygonal approximation with maximum distance and the newly created line segments are merged to circular or elliptical arcs where possible.

Therefore, parts of the input contours that can be approximated by long arcs can be found more efficiently. In the second step, parts of the input contours that can be approximated by short arcs

are found and the end parts of long arcs are refined.

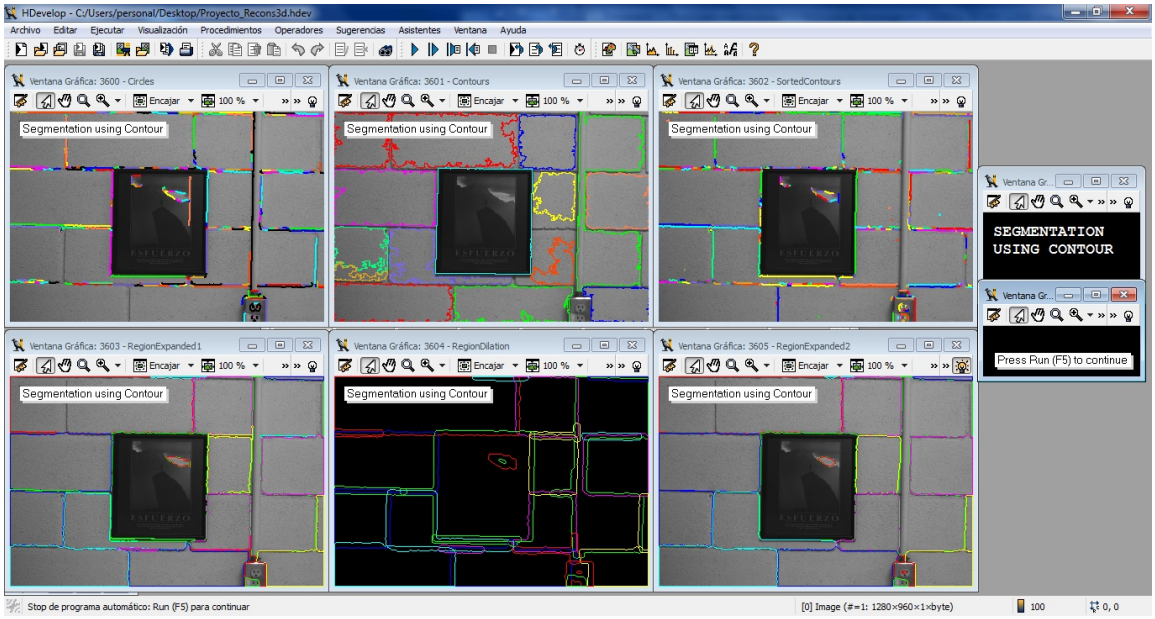


Figure 6.49: Segmentation using Contour

6.4.5.6 Step by step the 3D Reconstruction

The images obtained by the right and left camera are segmented by region technique and then its characteristic points were obtained by the technique of Sojka.

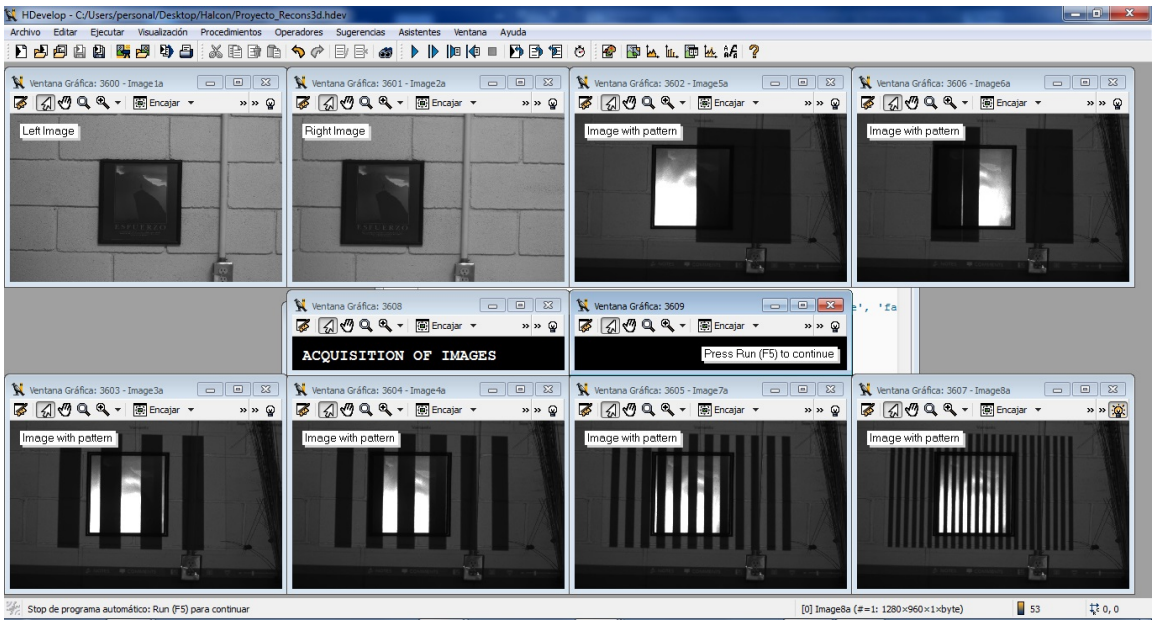


Figure 6.50: Acquisition of images with pattern of the first object

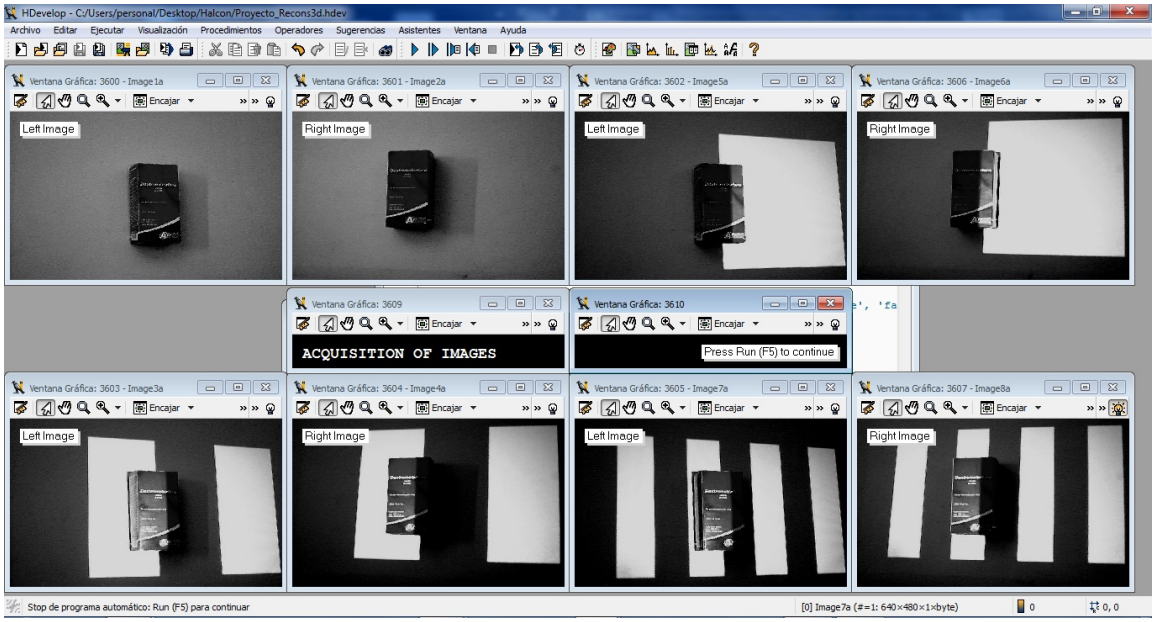


Figure 6.51: Acquisition of images with pattern of the second object

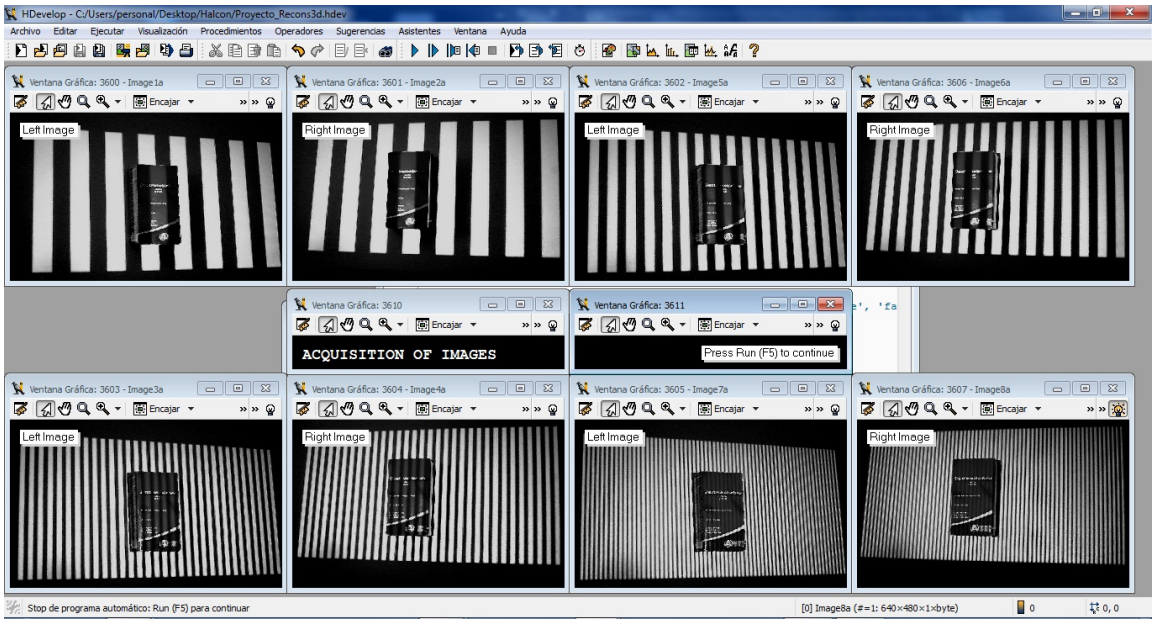


Figure 6.52: Acquisition of images with pattern of the second object

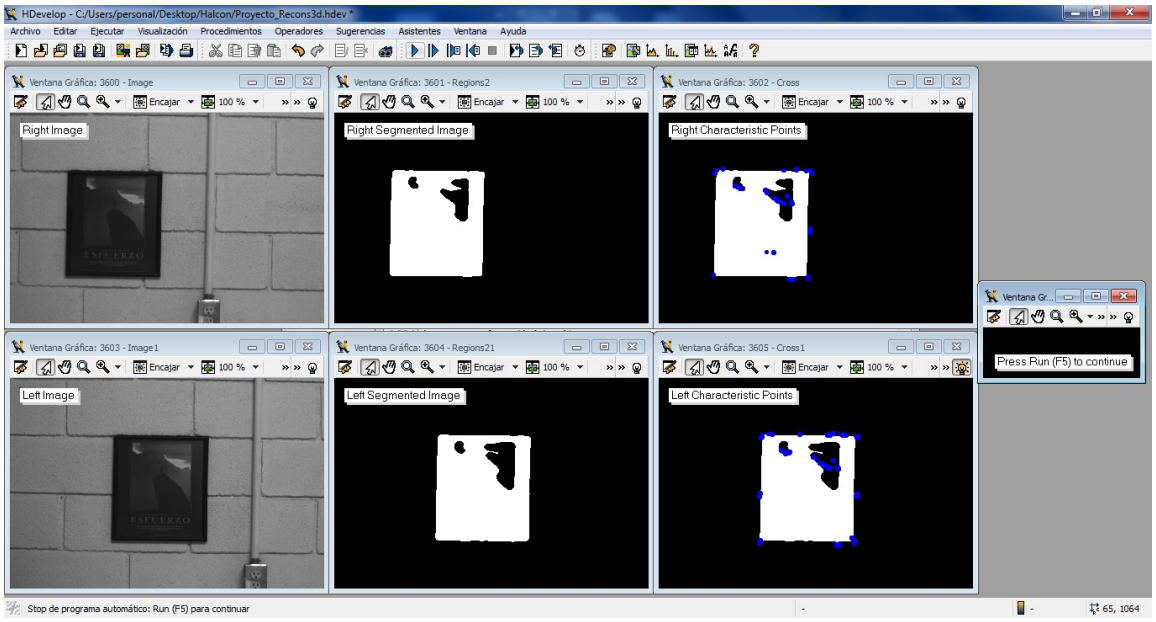


Figure 6.53: Management techniques to the left and right images for the first object

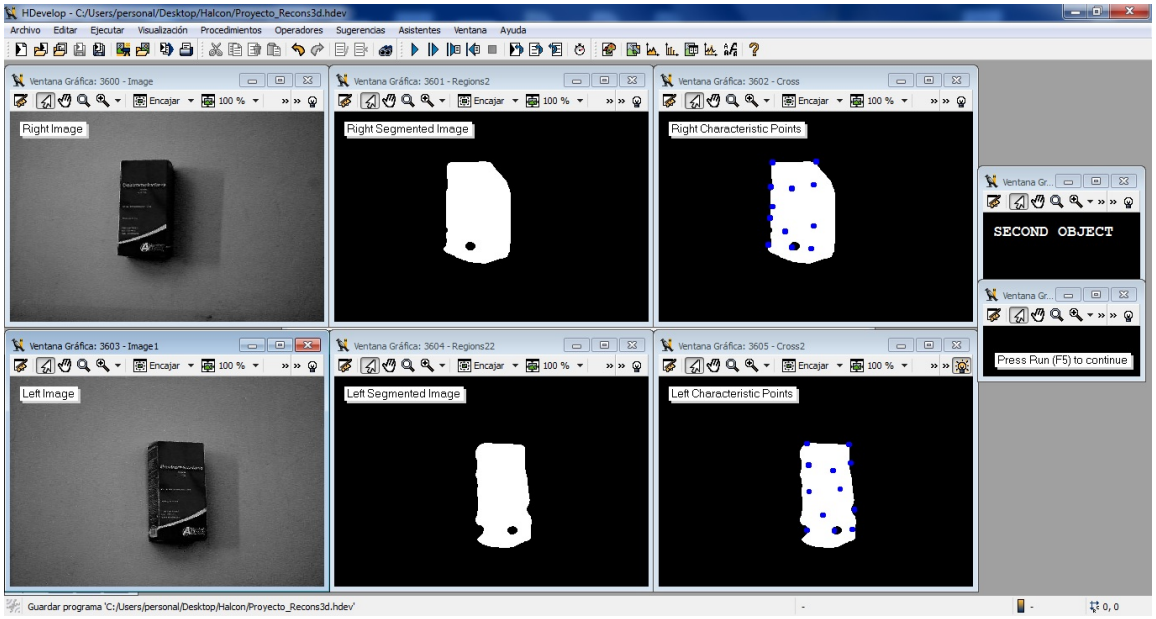


Figure 6.54: Management techniques to the left and right images for the second object

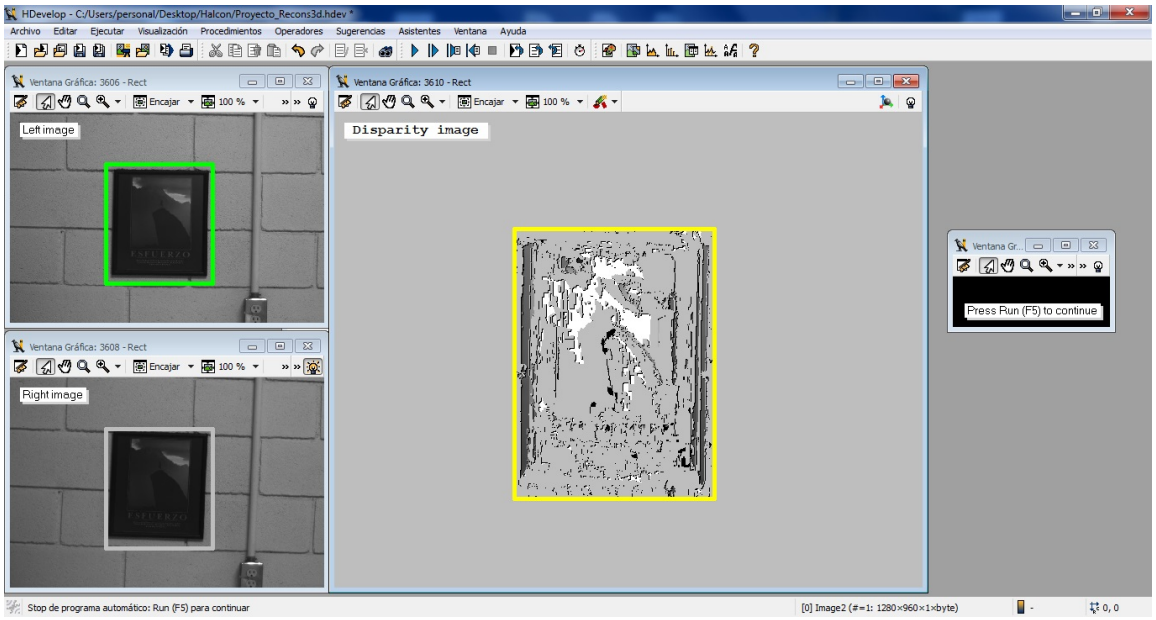


Figure 6.55: Disparity Map

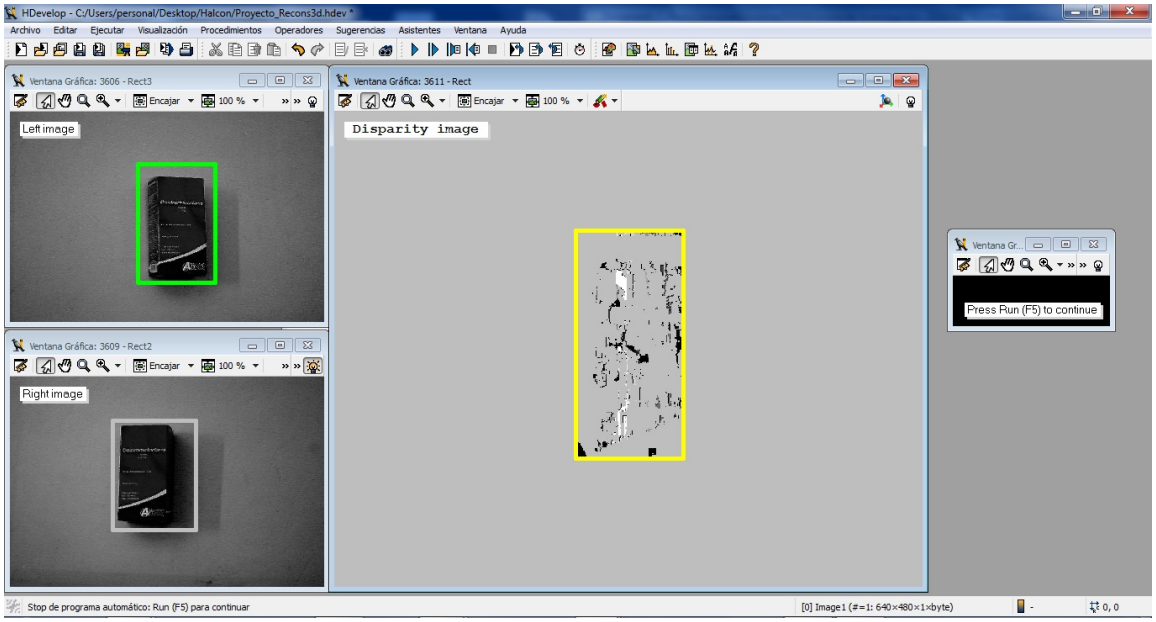


Figure 6.56: Points Cloud

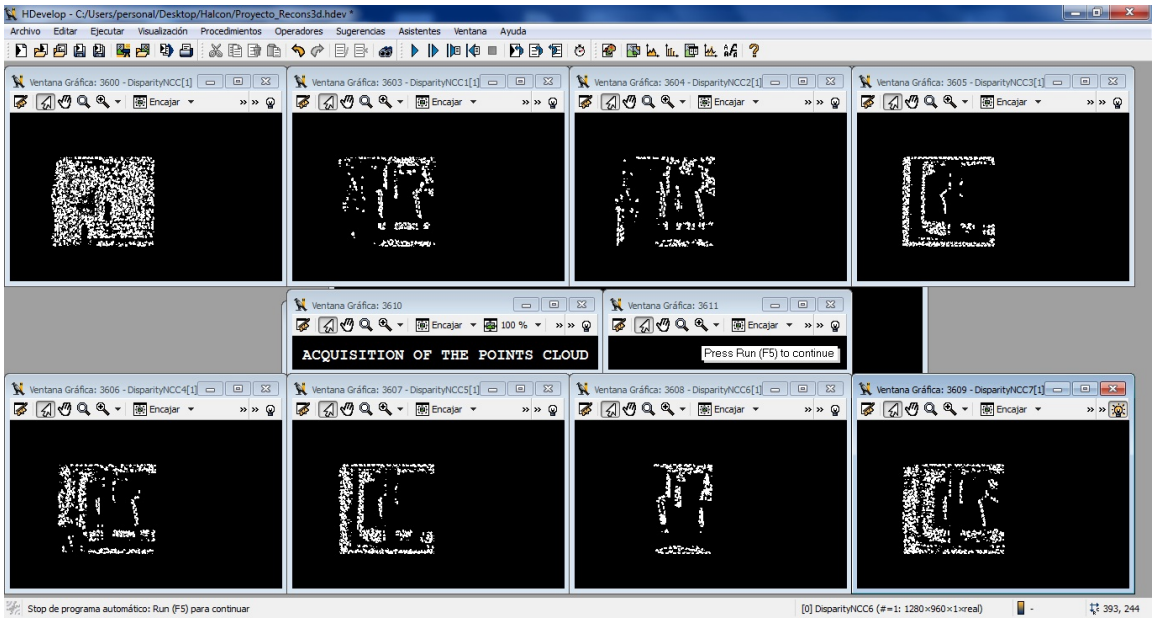


Figure 6.57: Points Cloud

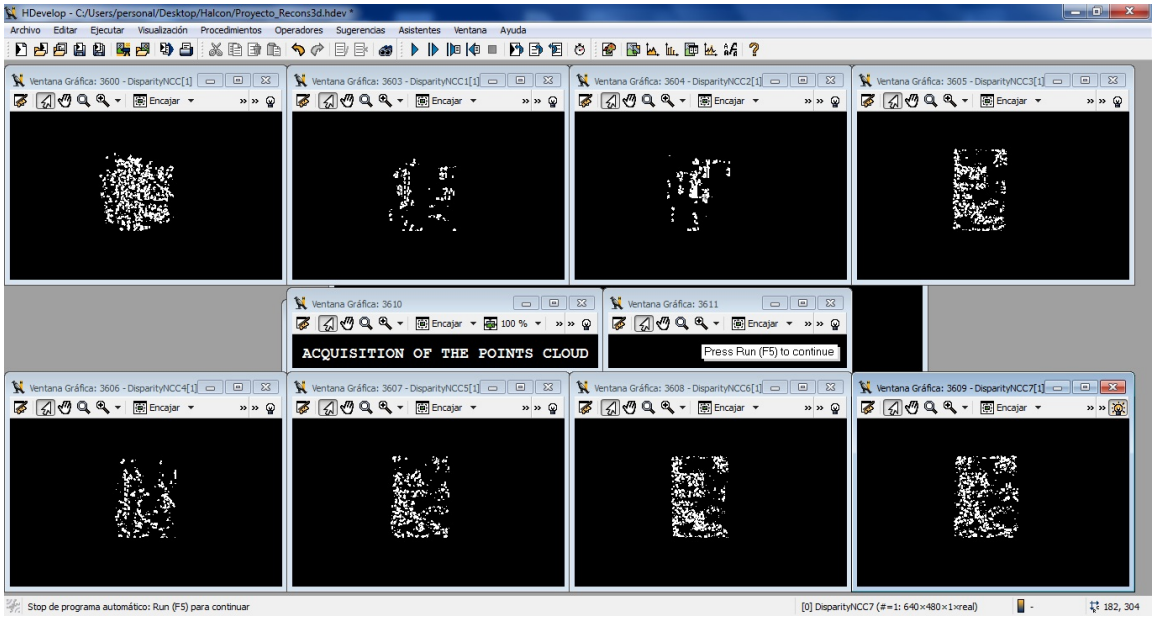


Figure 6.58: Disparity Map

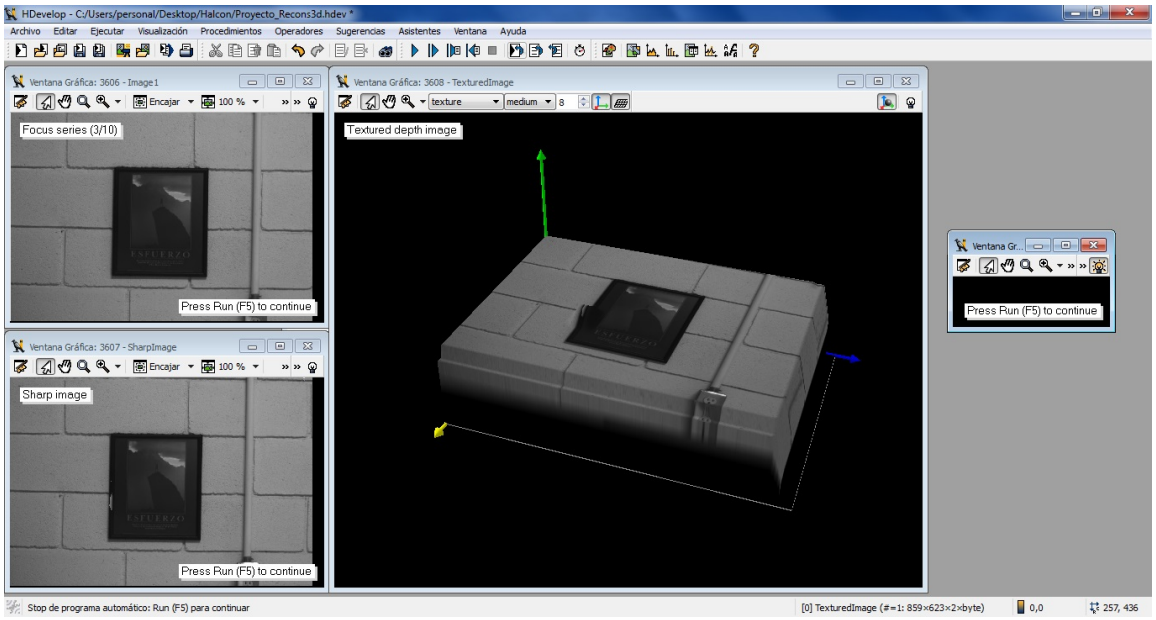


Figure 6.59: Textured to the left and right images for the first object

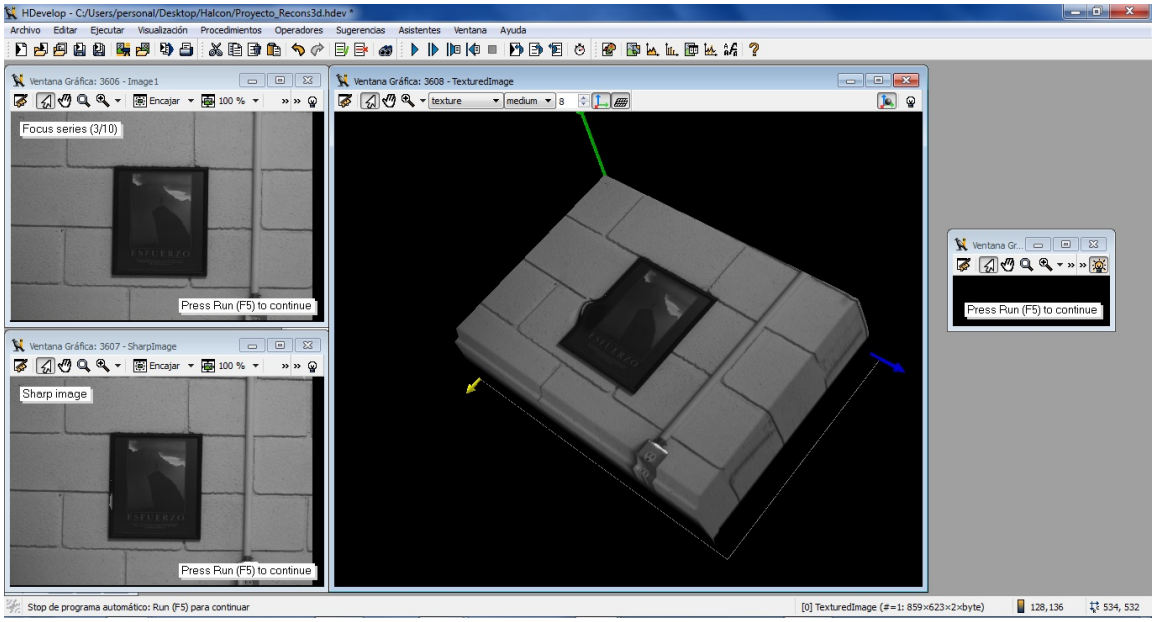


Figure 6.60: Management techniques rotating of the Textured Image

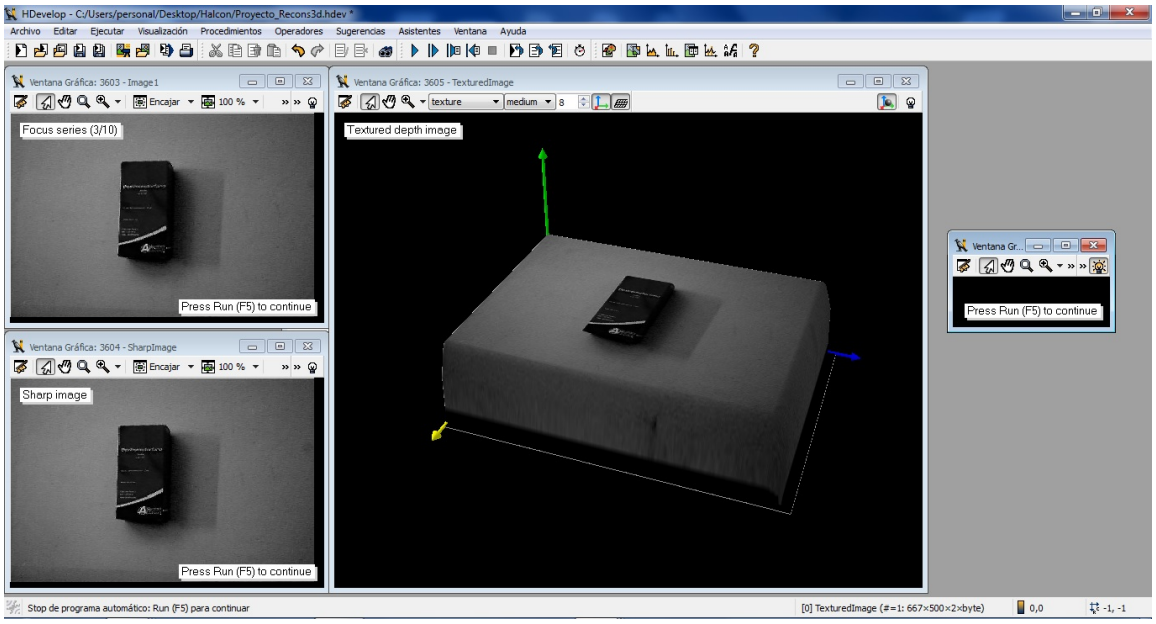


Figure 6.61: Textured to the left and right images for the second object

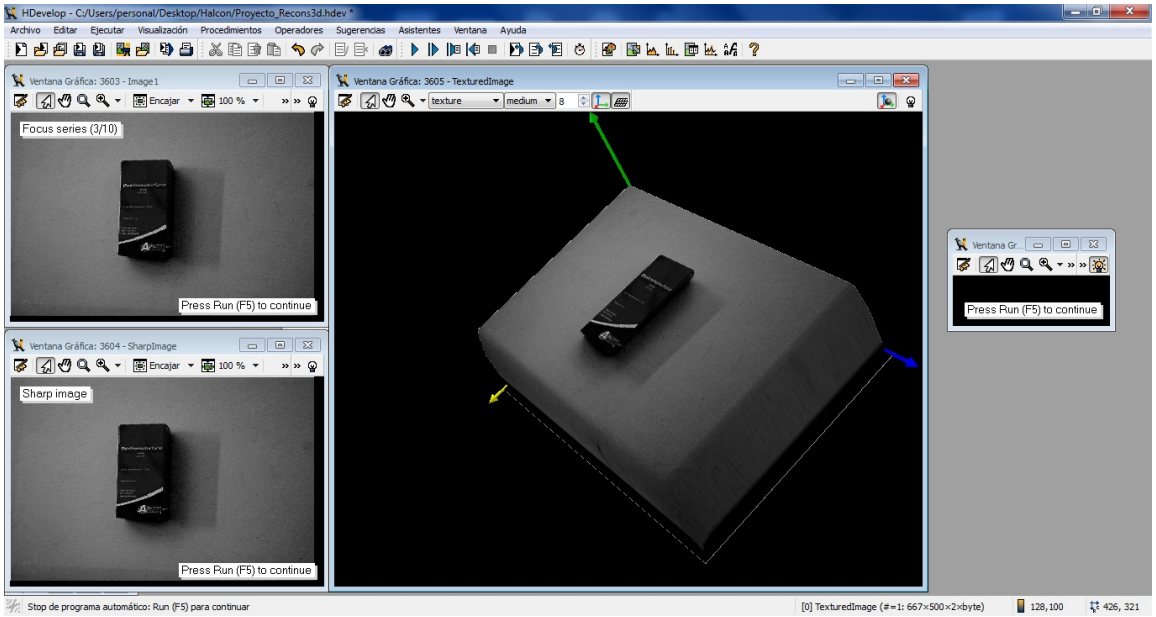


Figure 6.62: Management techniques rotating of the Textured Image

Chapter 7

Conclusion

7.1 Conclusions

The conclusion is that the project has been successfully completed, because it is designed and implemented the algorithm that allows to obtain a point cloud based on two cameras and a projector. Moreover, it has successfully completed all the objectives proposed at the beginning of the project. Although it required a prior knowledge of the software (HALCON), it was achieved to implement algorithms for obtaining the point cloud using interest points detectors, edge detectors and segmentation then use the binarization technique and to obtain the required result.

In this project, it has made a comprehensive analysis of the 3D Vision Techonlogies such as points of interest detectors, analyzing its main features. They have also been implemented and proven effective. Among the implanted detectors, the Sojka is the best result. Also, it has studied existing edge detectors and then implement them. Finally, a comparative study of the different methods was performed. The best performing detector is the Canny detector, detecting all the edges of objects. All other detectors based on gradient calculations are not as effective as the Canny detector. It has managed to implement a detector straight lines that detect correctly straight lines in the image. This detector is known as Hough detector and obtains optimal results is straight objects. For image segmentation have been studied various methods and algorithms programmed divide the image into several areas of interest, separating the object from the background such as Regions and Contour. The algorithm of Regions segmented images in color and allow to segment an image into several zones depending on the color. This algorithm has been used in the final algorithm because it got to separate the object from the background, while the contour only divided the image into zones. It is required to perform these stages of image processing to correctly handle the software (HALCON). Then, it proceeded with programming for obtaining the point cloud of objects that were used as evidence.

About the results of reconstruction, it can verify if the system is presented with an acquisition of images at random and is not done carefully, it is very poor. Therefore, one can not obtain an acceptable result, so that the fragility of this system is demonstrated. Also, if it is not done in a proper environment (completely dark room) could not get adequate for the reconstruction images. On the other hand, the part of the calibration is performed successfully obtaining search criteria. In the case of obtaining the point cloud, it can be concluded that a crude point cloud was obtained

because it was not taken into account factors affecting the reconstruction and some devices did not reach some characteristics that were needed. About the software (HALCON), it can conclude that it is a powerful vision software but to use completely and properly is necessary a preparation to avoid inconvenience when programming starts.

Finally, it is expected that the work is of interest, as a reference for later work and is the starting point to fill a need that is currently unmet.

7.2 Recommendations

- It is recommended to have a basic knowledge of the software (HALCON) to avoid difficulties in programming.
- Using equipment that provide fast processing speed, because the image processing is carried out through a large number of mathematical algorithms.
- Use cameras that have a good resolution and projectors with a good video resolution and good contrast.
- An environment with low light It is recommended to obtain good quality reconstructions.
- For calibration of the camera, it is recommended to be printed in a good quality printer for points of the template are well defined. These should be printed on white paper and points in black color. Furthermore, it should print on the recommended size.
- It is recommended that the number of images taken from the calibration of cameras should not be less than ten (10) images, in this way guarantee better results.

7.3 Future Work

This project provides the basis for future improvements and extensions. In an upcoming project extensions could be:

- Optimize algorithms points of interest for more accurate detection, ie, eliminate points that do not add information leaving only those significant image points. Besides, improve the image segmentation and so avoid that the algorithm detects the shadows and brightness of objects. This step is key to a successful obtaining a 3D object.
- Try other reconstruction techniques to see if better or faster results.
- The applications of 3D reconstruction are vast and expect continued progress enables new applications that will test the validity of the results presented. These new developments must be capable of being applicable to the system.
- 3D reconstruction algorithm used in this project presents results that could be improvable through the use of new techniques and better management of software (HALCON).

Bibliography

[1] EMVA, *EUROPEAN MACHINE VISION ASSOCIATION (EMVA): EMVA Standard 1288 - Standard for Characterization and Presentation of Specification Data for Image Sensors and Cameras*, 2006.

[2] K. Weber, *CMOS: Listos para el Broadcast de hoy.* Grass Valley, February 2012.

[3] M. Schwar and D. Toth, *Camera Selection-How can I find the right camera for my image processing system?* Basler AG, April 2014.

[4] F. DePiero and M. Trivedi, *3D computer vision using structured light: Design, Calibration and Implementation issues.* Electrical and Computer Engineering Department.

[5] I. Kalisperakis and G. Karras, *A structured light approach for the reconstruction of complex objects.* Department of surveying technological educational Institute of Athens.

[6] F. Pernkopf, *3D surface acquisition and reconstruction for inspection of raw steel products.* Computers in Industry, Vol. 56, 2005, pp. 876885.

[7] P. Besl, *Active, optical range imaging sensors.* Machine Vision and Applications, Vol. 1, 1988.

[8] O. Faugeras, *Three-Dimensional computer vision: A geometric viewpoint.* The MIT Press, 1993.

[9] M. Smith and R. Stamp, *Automated inspection of textured ceramic tiles.* Comput. Ind., Vol. 43, 2000.

[10] R. McKeon and P. Flynn, *Three Dimensioanl facial imaging using a static light screen and a dynamic subject.* IEEE Transactions on Instrumentation and Measurement, April 2010.

[11] D. García and J. González, *3D Inspection system for manufactured machine parts.* Machine Vision Systems for Inspection and Metrology VIII, Vol. 3835, 1999, pp. 236.243.

[12] G. Zhang and Z. Wei, *A novel calibration approach to structured light 3D vision inspection.* Optics and Laser Technology, Vol. 34, 2002, pp. 373380.

[13] R. Klette and A. Koschan, *Computer Vision: Three-Dimensional Data from Images.* Springer-Verlag, 1998.

[14] J. Batlle and J. Salvi, *Recent progress in coded structured light as a technique to solve the correspondence problem: A survey.* Pattern Recognition, vol. 31, no. 7, 1998, pp. 963–982.

[15] F. Bernardini and H. Rushmeier, *The 3d model acquisition pipeline.* Computer Graphics Forum, vol. 21, no. 2, June, 2002, pp. 149–172.

[16] J. Forest and E. Cabruja, *A proposal for laser scanners sub-pixel accuracy peak detector.* Workshop on European Scientific and Industrial Collaboration, Mickolj, Hungary, May, 2003, pp. 525–532.

[17] A. Dipanda and S. Woo, *Towards a real-time 3d shape reconstruction using a structured light system.* Pattern Recognition, vol. 38, no. 10, October 2005, pp. 1632–1650.

[18] B. Horn, *Robot Vision.* MIT Press, 1986.

[19] R. Hartley and A. Zisserman, *Vision Artificial: Aplicacion practica de la vision artificial en el control de procesos industriales.* Institut La Garrotxa, Abril 2012.

[20] A. Labarile and A. Distante, *Ballast 3D reconstruction by a matching pursuit based stereo matcher.* 2004 IEEE Intelligent Vehicles Symposium, Parma, Italy, 2004.

[21] A. Hoover, *The space envelope representation for 3D scenes.* Depatament of Computer Science and Engineering, U. of South Florida, 1996.

[22] F. Bertagnolli and R. Dillmann, *Flexible automated process assurance through non-contact 3D measuring technology.* IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2003),Tokyo, Japan, 2003.

[23] J. Leopold and R. Leopold, *New developments in fast 3Dsurface quality control.* Measurement, Vol.33, 2003.

[24] F. Pernkopf and OLeary., *Image acquisition techniques for automatic visual inspection of metallic surfaces.* NDTE International, Vol. 39, No. 8, 2003, pp. 609617.

[25] D. García and V. Fernández, *Real-time flatness inspection system for steel strip production lines.* Real-Time Imaging, Vol. 5, 1999, pp. 3547.

[26] F. Pernkopf, *3D Surface inspection using coupled HMMs.* Proc. of the 17th International Conference on Pattern Recognition (ICPR04), Vol. 3, 2004, pp. 223226.

[27] J. Pagès and C. Matabosch, *Overview of coded light projection techniques for automatic 3D profiling.* Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 2003.

[28] L. Zhang and S. Seitr, *Rapid shape acquisition using color structured light and multi-pass dynamic programming.* International Symposium on 3D Data Processing Visualization and Transmission, Padova, Italy, 2002.

[29] J. Gühring, *Dense 3-D surface acquisition by structured light using off-the-shelf components.* Videometrics and Optical Methods for 3D Shape Measurement, Vol. 4309, 2001, pp. 220231.

[30] J. Pagès and J. Forest, *Optimised De Bruijn patterns for one-shot shape acquisition.* Image and Vision Computing, Vol. 23, 2005, pp. 707720.

[31] M. Hebert and E. Krotkov, *3-D Measurements from imaging laser radars: How good are they.* IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS91, Osaka, Japan, 1991, pp.127-152.

[32] M. Schaulin and K. Wolter, *White light interferometry, a method for optical 3D-inspection of advanced packages.* IEEE 2Th Int. Spring Seminar on Elect. Tech, 2004.

[33] W. Osten, *Application of optical shape measurement for the nondestructive evaluation of complex objects.* Opt. Eng., Vol. 39, No. 1, 2000, pp. 232243.

[34] L. D. P. A. Gastelum and J. Marquez, *Sistema de Estereo-Vision para la captura de Rostros Humanos*. SOMI Congreso de Instrumentación, XXIX Edición, Octubre 2014.

[35] L. Li, *Time of Flight Camera - An Introduction*. Technical White Paper, Texas Instruments, January 2014.

[36] C. Rocchini and R. Scopigno, *A low cost 3d scanner based on structured light*. Computer Graphics Forum, vol. 20, no. 3, September, 2001, pp. 299–308.

[37] K. Hattori and Y. Sato, *Handy rangefinder for active robot vision*. IEEE International Conference on Robotics and Automation, vol. 2, Nagoya, Japan, May 1995, pp. 1423–1428.

[38] L. Matthies and B. Wilcox, *Fast optical hazard detection for planetary rovers using multiple spot laser triangulation*. IEEE International Conference on Robotics and Automation, vol. 1, Albuquerque, USA, April, 1997, pp. 859–866.

[39] P. de la Hamette and G. Troster, *Laser triangulation as a means of robust visual input for wearable computer*. Proceedings of the 8th IEEE International Symposium on Wearable Computers, Arlington, USA, October, 2004, pp. 18–20.

[40] M. Asada and S. Tsuji, *Determining surface orientation by projecting a stripe pattern*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 5,, September, 1988, pp. 749–754.

[41] S. Winkelbach and F. Wahl, *Shape from single stripe pattern illumination*. Pattern Recognition : Proceedings of the 24th DAGM Symposium,, ser. Lecture Notes in Computer Science, L. V. Gool, Ed, vol. 2449, Zurich, Switzerland, September, 2002, pp.240–247.

[42] N. Shrikhande and G. Stockman, *Surface orientation from a projected grid*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 6, June, 1989, pp. 650–655.

[43] L. Guisser and S. Castan, *Pgsd: an accurate 3d vision system using a projected grid for surface descriptions*. Image and Vision Computing, vol. 18, no. 6-7, May, 2000, pp. 463–491.

[44] D. Rocheleau and W. Ranson, *Planar surface reconstruction using circle generator laser light*. Optics and Lasers in Engineering, vol. 30, no. 1, July, 1998, pp. 39–52.

[45] R. Valkenburg and A. McIvor, *Accurate 3d measurement using a structured light system*. Image and Vision Computing, vol. 16, no. 2, February, 1998, pp. 99–110.

[46] L. Zhang and S. Seitz, *Rapid shape acquisition using color structured light and multi-pass dynamic programming*. Proceedings of the 1st International Symposium on 3D Data Processing, Visualization and Transmission,, Padova, Italy, June, 2002, pp. 24–36.

[47] H. Li and H. Prautzsch, *Fast subpixel accurate reconstruction using color structured light,*. Proceedings of Visualization, Imaging and Image Processing, Marbella,, Spain, September, 2004.

[48] E. Sojka, *A New and Efficient Algorithm for Detecting the Corners in Digital Images*. Pattern Recognition, Luc Van Gool (Editor), LNCS 2449, Springer Verlag, 2002, pp. 125-132.

[49] D. Caspi and J. Shamir, *Range imaging with adaptive color structured light*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 5, May, 1998, pp. 470–480.

[50] T. Koninckx and L. Gool, *Real-time range scanning of deformable surfaces by adaptively coded structured light*. Proceedings of the 4th International Conference on 3-D Digital Imaging and Modeling, Canada, October, 2003, pp. 293–300.

[51] F. MacWillians and N. Sloane, *Pseudorandom sequences and arrays.* Proc. of IEEE 64(12), 1976.

[52] T. Monks, *Measuring the shape of time varying objects.* PhD Thesis University of Southampton, 1994.

[53] J. Salvi and E. Mouaddib, *A robust coded pattern projection for dynamic 3D scene measurement.* Pattern Recognition Letters, ELSEVIER, 19, 1998, pp. 1055-1065.

[54] L. Zhang and S. Seitz, *Rapid shape acquisition using color structured light and multi-pass dynamic programming.* Proceedings of 3D DPVT, IEEE, 2002, pp. 532-535.

[55] O. Hall-Halt and S. Rusinkiewicz, *Stripe boundary codes for real-time structured light range scanning of moving objects.* International Conference on Computer Vision Computing 11, 1993, pp. 251-256.

[56] P. Vulsteke and A. Osterlinck, *Range image acquisition with a single binary-encoded light pattern.* Transactions on Pattern Analysis and Machine Intelligence, IEEE, 12(2), 1990, pp. 148-164.

[57] P. G. S. Yee, *Generation of uniquely encoded light patterns for range data acquisition.* International Journal on Pattern Recognition, 1992, pp. 609-616.

[58] C. Davies and M. Nixon, *A hough transform for detecting the location and orientation of three-dimensional surfaces via color spots.* Transactions on Systems Man and Cybernetics, IEEE, 1998, pp. 90-95.

[59] R. Morano and J. Nissanov, *Structured light using pseudorandom codes.* Transactions on Pattern Analysis and Machine Intelligence, IEEE, 1998, pp. 322-327.

[60] M. Maruyama and S. Abe, *Range sensing by projecting multiple slits with random cuts.* Transactions on Pattern Analysis and Machine Intelligence, IEEE, 1987, pp. 647-651.

[61] K. Boyer and A. Kak, *Color-encoded structured light for rapid active ranging.* Transactions on Pattern Analysis and Machine Intelligence, IEEE, 1987, pp. 14-28.

[62] C. Chen and J. Wu, *Range data acquisition using color structured light and stereo vision.* International Journal on Image and Vision Computing, 1997, pp. 445-456.

[63] J. Salvi and X. Llado, *A state of the art in structured light patterns for surface profilmetry.* Pattern Recognition, Vol. 42, No. 8, 2010, pp. 2666-2680.

[64] A. Adan and A. Jimenez, *Sistema de Recuperación de información 3D en escenas dinámicas.* XXV Jornadas de Automática, 2004.

[65] J. Hoyos and L. Orozco, *Técnicas de calibración de cámaras para visión estéreo y reconstrucción.* XV Simposio de Tratamiento de señales, imagenes y vision artificial, February 2010.

[66] W. Förstner and E. Gülch, *A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Circular features.* In Proceedings of the Intercommission Conference on Fast Processing of Photogrametric Data, Interlaken, 1987, pp. 281-305.

[67] W. Förstner, *A Framework for Low Level Feature Extraction.* European Conference on Computer Vision, LNCS 802, Springer Verlag, 1994, pp. 383-394.

[68] C. Harris and M. Stephens, *A combined corner and edge detector*. Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147-151.

[69] J. Canny, *A Computational Approach to EdgeDetection*. IEEE Trans. Pattern Analysis and Machine Intelligence, November, 1986.

[70] S. E. Mejdani and F. Dubeau, *Old and new straight-line detectors: Description and comparison*, September, 2005.

[71] D. Ballard, *Generalizing the Hough Transform to detect arbitrary shapes*, September, 1980.

[72] *State-of-the-Art Survey on Color Segmentation Methods*, 2005.

Appendices

Appendix A

Appendix

A.1 Appendix I: Camera Basler acA

A.1.1 Installation and Setup

A.1.1.1 Introduction

This document provides the information you will need to install and operate Basler GigE Vision, IEEE 1394, and Camera Link® cameras. The installation procedure relates to both hardware and Basler pylon software. Unless otherwise noted, the material in this manual applies to all Basler cameras using Basler pylon software, regardless of camera model or type of interface. We strongly recommend that you read and follow the precautions given in this document and all further precautions given in the camera user manuals.

Refer to the camera user manuals for additional important information such as:

- mechanical specifications, including mounting points
- mechanical stress test results
- environmental requirements.

If you are using a GigE Vision camera, refer to the camera's User Manual for information on improving your camera's performance in a network and on using multiple cameras.

A.1.1.2 Licensing Information

A.1.1.2.1 pylon API

The pylon API is based on the GenApi module of the GenICam™ reference implementation distributed under a modified BSD license and is copyright (c) 2005, Basler Vision Technologies.

A.1.1.3 Avoiding EMI and ESD Problems

The cameras are frequently installed in industrial environments. These environments often include devices that generate electromagnetic interference (EMI) and they are prone to electrostatic discharge (ESD). Excessive EMI and ESD can cause problems with your camera such as false triggering or can cause the camera to suddenly stop capturing images. EMI and ESD can also have a negative impact on the quality of the image data transmitted by the camera.

To avoid problems with EMI and ESD, you should follow these general guidelines:

- Always use high quality shielded cables. The use of high quality cables is one of the best defenses against EMI and ESD.
- Try to use camera cables that are the correct length and try to run the camera cables and power cables parallel to each other. Avoid coiling camera cables. If the cables are too long, use a meandering path rather than coiling the cables.
- Avoid placing camera cables parallel to wires carrying high-current, switching voltages such as wires supplying stepper motors or electrical devices that employ switching technology. Placing camera cables near to these types of devices may cause problems with the camera.
- Install the camera and camera cables as far as possible from devices generating sparks. If necessary, use additional shielding.

A.1.1.4 Installing a GigE Vision Camera

A.1.1.4.1 General Considerations

The installation procedures assume that you will be making a peer-to-peer connection between your camera and a desktop computer. Make sure that the following items are available before starting the installation:

- A Basler GigE camera
- As applicable, a power supply or a GigE power injector: Make sure that the power supply meets all of the requirements listed in the Physical Interface section of the camera User's Manual. If you want to use Power over Ethernet (PoE) as an alternative for the ace camera use a GigE power injector
- A desktop computer with a GigE network adapter installed. We recommend that you use an Intel® PRO 1000 series adapter. These adapters have been tested with Basler cameras and work well. This installation procedure assumes that your computer is equipped with a PRO 1000 series adapter
- A standard Ethernet patch cable. We recommend the use of a category 6 or category 7 cable that has S/STP shielding (two cables if you are using a power injector)
- Basler pylon Software

During installation of the Basler pylon Driver Package, Basler network drivers are bound to all network adapters installed in your computer. This applies not only to all network adapters used to connect to cameras, but also to all other network adapters installed in your PC. Often, your PC will have two network adapters installed, with one used to connect to cameras and the other used to connect to a local area network.

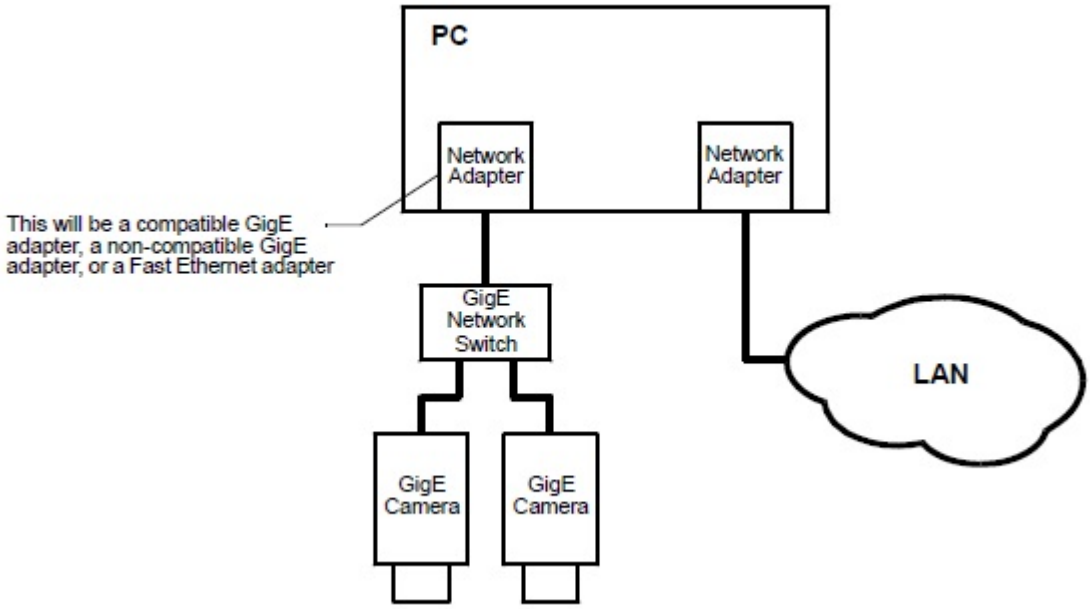


Figure A.1: Network adapter for the cameras

Two drivers are available for use with your GigE cameras:

- The Basler filter driver: is a basic GigE Vision network driver that is compatible with all network adapters. The advantage of the filter driver is its extensive compatibility.
- The Basler performance driver: is a hardware specific GigE Vision network driver. The performance driver is only compatible with network adapters that use specific Intel chipsets ("compatible chipsets"). The advantage of the performance driver is that it significantly lowers the CPU load needed to service the network traffic between the PC and the camera(s). It also has a more robust packet resend mechanism

A.1.1.4.2 Basler pylon Software Installation

For the installation, choose the appropriate name of the installer from the following figure, depending on the operating system of your PC.

	32 bit Operating System	64 bit Operating System
Basler pylon Driver Package	Basler pylon SDK x86 x.x.x.xxx.exe	Basler pylon SDK x64 x.x.x.xxx.exe

Figure A.2: Installer Names

1. If you have old Basler pylon software installed on your system, make sure to uninstall the software.
2. Make sure your GigE camera is disconnected from your computer.
3. Close all open applications.
4. Download the installer from the Basler website
5. Launch the downloaded executable. The program will prepare to install and then a Welcome window will open.
6. Click the Next button. A License Agreement window will open.
7. Accept the agreement and click the Next button. A Customer Information window will open.
8. Enter the appropriate information and click the Next button.
9. In the Destination window determine the directory where you want to install the software to and click the Next button.

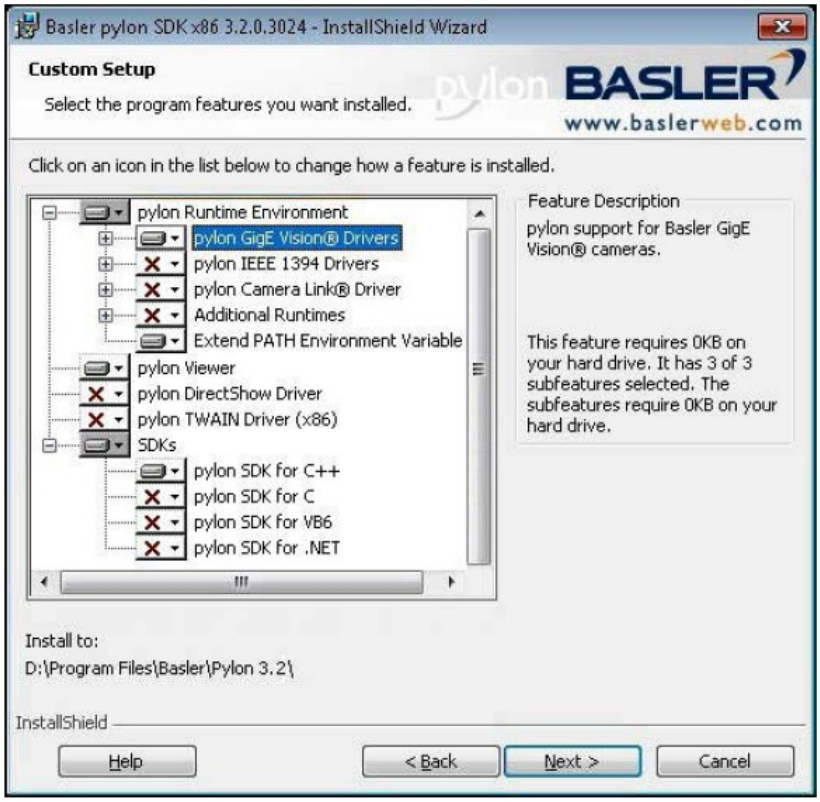


Figure A.3: Installation of Basler

10. Click the Next button.
11. A Ready to Install the Program window will open.
12. Click the Install button. When the installation process is complete, a Completed window will open.
13. Click the Finish button

A.1.1.5 Adjusting the Installation

This section provides information on adjustments that must be made after the installation of the Basler pylon Driver Package and on additional adjustments that may be needed.

A.1.1.5.1 Disabling the Windows Firewall

The Basler pylon software requires the Windows firewall to be disabled for all interfaces where cameras are connected, with the exception indicated below.

If you use Basler GigE cameras with Windows 7 you do not necessarily need to disable the firewall.

If you leave the firewall enabled, the camera can be fully used with the exception of the event reporting feature. You must only disable the firewall if you want to preserve the possibility of receiving events from the camera.

However, you will have to respond to cumbersome messages when not disabling the firewall: Whenever a program addresses a camera for the first time, a Windows Security Alert will open asking you to allow incoming requests. In these cases, click Cancel to block each message. It recommend to disable the firewall for the connections with the cameras.

A.1.1.6 Camera and Network Adapter IP Configuration

A.1.1.6.1 Changing a Camera's IP Configuration

An application called the pylon IP Configuration Tool is included as part of the pylon driver installation package. The pylon IP Configuration Tool shows you the current IP configuration of your camera and allows you to change it.

When you start the pylon IP Configuration Tool, it scans the PC for network adapters ("connections") and attached cameras. All discovered network adapters and cameras will be displayed in the top pane. Detailed information about the item selected in the top pane will be displayed in the central area below.

Depending on the firmware version of your camera, the process to change the IP configuration can differ. With older cameras, an intermediate step may be necessary in which you assign a temporary IP address in order to establish communication between the camera and the PC. If this is the case, the pylon IP Configuration Tool will automatically open the Assign Temporary IP Address (Force IP) dialog. For newer cameras this is not necessary, because they can automatically establish communication with the PC.

During normal operation, you may want to change the camera's IP configuration in e.g. the following typical situations:

- A different way of IP address assignment is desired for operational reasons, e.g. via a DHCP server instead of using a static IP address.

- A temporary IP address has to be assigned when the camera is moved to a different port or network adapter and therefore has to operate in a different subnet.

When you configure a camera to use either a temporary or a static IP address, there are some things that you must keep in mind:

- For a camera to communicate properly, it must be in the same subnet as the adapter to which it is connected.
- The camera must have an IP address that is unique within the network.
- The recommended range for static IP addresses is from 172.16.0.1 to 172.32.255.254 and from 192.168.0.1 to 192.168.255.254. These address ranges have been reserved for private use according to IP standards.
- If your PC has multiple network adapters, each adapter must be in a different subnet.

A.2 Appendix II: Programming's Patforms used during the project

A.2.1 HALCON

A.2.1.1 Acquisition of Images

```
dev_update_off ()
info_framegrabber ('DirectShow', 'general', GeneralInfo, GeneralInfoValues)
info_framegrabber ('DirectShow', 'revision', RevInfo, RevInfoValues)
info_framegrabber ('DirectShow', 'camera_types', CameraTypeInfo, CameraTypeInfoValues)
info_framegrabber ('DirectShow', 'info_boards', BoardInfo, BoardInfoValues)
MyCameraType1 := 'default'
MyCameraType2 := 'default'
MyDevice1 := '0'
MyDevice2 := '1'
MyPort1 := -1
MyPort2 := -1
open_framegrabber ('DirectShow', 1, 1, 0, 0, 0, 0, 'default', -1, 'gray', -1, 'false', MyCameraType1,
MyDevice1, MyPort1, -1, AcqHandle1)
open_framegrabber ('DirectShow', 1, 1, 0, 0, 0, 0, 'default', -1, 'gray', -1, 'false', MyCameraType2,
MyDevice2, MyPort2, -1, AcqHandle2)
get_framegrabber_param (AcqHandle1, ['camera_type','frame_rate'], CurrentSettings1)
get_framegrabber_param (AcqHandle2, ['camera_type','frame_rate'], CurrentSettings2)
grab_image (Image1, AcqHandle1)
grab_image (Image2, AcqHandle2)
get_image_size (Image1, Width1, Height1)
get_image_size (Image2, Width2, Height2)
dev_close_window ()
dev_close_window ()
dev_open_window (0, 0, Width1 / 2, Height1 / 2, 'black', WindowHandle1)
dev_open_window (0, Width1 / 2 + 20, Width2 / 2, Height2 / 2, 'black', WindowHandle2)
count_seconds (SecondsOld)
while (1)
grab_image (Image1, AcqHandle1)
dev_set_window (WindowHandle1)
dev_display (Image1)
grab_image (Image2, AcqHandle2)
dev_set_window (WindowHandle2)
dev_display (Image2)
count_seconds (SecondsNew)
FrameRate := 1 / (SecondsNew - SecondsOld)
SecondsOld := SecondsNew
endwhile
close_framegrabber (AcqHandle1)
close_framegrabber (AcqHandle2)
```

A.2.1.2 Calibration of Cameras

```
ImageFiles := []
ImageFiles[0] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_01.png'
ImageFiles[1] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_02.png'
ImageFiles[2] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_03.png'
ImageFiles[3] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_04.png'
ImageFiles[4] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_05.png'
ImageFiles[5] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_06.png'
ImageFiles[6] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_07.png'
ImageFiles[7] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_08.png'
ImageFiles[8] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_09.png'
ImageFiles[9] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_10.png'
ImageFiles[10] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_13.png'
ImageFiles[11] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_16.png'
ImageFiles[12] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_17.png'
ImageFiles[13] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_18.png'
ImageFiles[14] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_19.png'
ImageFiles[15] := 'C:/Users/Public/Documents/MVTec/HALCON-11.0/examples/images/Calibracion
Basler/image_20.png'
TmpCtrl_ReferenceIndex := 13
TmpCtrl_PlateDescription := 'C:/Program Files/MVTec/HALCON-11.0/calib/caltab_30mm.descr'
StartParameters := [0.008,0,8.3e-006,8.3e-006,640,480,1280,960]
TmpCtrl_FindCalObjParNames := ['gap_tolerance','alpha','skip_find_caltab']
TmpCtrl_FindCalObjParValues := [1,0.9,'false']
create_calib_data ('calibration_object', 1, 1, CalibHandle)
set_calib_data_cam_param (CalibHandle, 0, 'area_scan_division', StartParameters)
set_calib_data_calib_object (CalibHandle, 0, TmpCtrl_PlateDescription)
for Index := 0 to —ImageFiles—-1 by 1
read_image (Image, ImageFiles[Index])
find_calib_object (Image, CalibHandle, 0, 0, Index, TmpCtrl_FindCalObjParNames,
TmpCtrl_FindCalObjParValues)
endfor
```

calibrate_cameras (CalibHandle, TmpCtrl_Errors)
get_calib_data (CalibHandle, 'camera', 0, 'params', CameraParameters)
get_calib_data (CalibHandle, 'calib_obj_pose', [0, TmpCtrl_ReferenceIndex], 'pose', CameraPose)
set_origin_pose (CameraPose, 0.0, 0.0, 0.001, CameraPose)
clear_calib_data (CalibHandle)
stop ()

A.2.1.3 Obtaining of Points Cloud

```
* Part 1 Introduction
dev_update_off ()
list_image_files ('proyecto_recons3d', 'default', [], AllImageFiles)
tuple_regexp_select (AllImageFiles, 'intensities', ImageFiles) ImageFiles := sort(ImageFiles)
* Prepare windows for introduction
dev_close_window ()
dev_open_window (250, 300, 748 + 12, 300, 'black', WindowHandle3D)
dev_open_window (0, 300, 748 / 2, 240, 'black', WindowHandle1)
dev_open_window (0, 300 + 748 / 2 + 12, 748 / 2, 240, 'black', WindowHandle2)
set_display_font (WindowHandle1, 16, 'mono', 'true', 'false')
set_display_font (WindowHandle2, 16, 'mono', 'true', 'false')
set_display_font (WindowHandle3D, 16, 'mono', 'true', 'false')
* Prepare Visualization
Color1 := 'green'
Color2 := 'yellow'
Color3 := 'gray'
dev_set_window (WindowHandle1)
dev_set_draw ('margin')
dev_set_color (Color1)
dev_set_window (WindowHandle2)
dev_set_draw ('margin')
dev_set_color (Color2)
* Prepare 3D Visualization
CamParam1 := [0.06,0,8.5e-6,8.5e-6,380.0,150.0,760,300]
CamParam2 := [0.06,0,8.5e-6,8.5e-6,380.0,275.0,760,550]
create_pose (70, -20, 1800, 125, 345, 185, 'Rp+T', 'gba', 'point', PoseVisualize)
VisualizeParameterValues := [Color3,Color1,Color2,1.0,1.0,1.0]
read_image (Image, 'proyecto_recons3d/cuadro')
read_image (Image1, 'proyecto_recons3d/cuadro')
read_image (Image2, 'proyecto_recons3d/cuadro')
read_image (Image3, 'proyecto_recons3d/cuadro')
read_image (Image4, 'proyecto_recons3d/cuadro')
read_image (Image5, 'proyecto_recons3d/cuadro')
read_image (Image6, 'proyecto_recons3d/cuadro')
read_image (Image7, 'proyecto_recons3d/cuadro')
read_image (Image8, 'proyecto_recons3d/cuadro')
read_image (Image9, 'proyecto_recons3d/cuadro')
read_image (Image10, 'proyecto_recons3d/cuadro')
read_image (Image11, 'proyecto_recons3d/cuadro')
read_image (Image12, 'proyecto_recons3d/cuadro')
read_image (Image13, 'proyecto_recons3d/cuadro')
read_image (Image14, 'proyecto_recons3d/cuadro')
read_image (Image15, 'proyecto_recons3d/cuadro')
read_image (Image16, 'proyecto_recons3d/cuadro')
read_image (Image17, 'proyecto_recons3d/cuadro')
read_image (Image18, 'proyecto_recons3d/cuadro')
```

```
read_image (Ima, 'proyecto_recons3d/recons')
read_image (Log, 'proyecto_recons3d/logo')
gen_rectangle1 (Rectangle, 0, 0, 479, 747)
dev_set_window (WindowHandle1)
dev_display (Ima)
dev_set_window (WindowHandle2)
dev_display (Log)
Message := 'ARTIFICIAL VISION SYSTEM FOR THE ACQUISITION OF A POINTS CLOUD'
Message[1] := 'USING 3D VISION TECHNOLOGIES'
Message[2] := ' '
Message[3] := 'Deyby Maycol Huamanchahua Canchanya'
Message[4] := ' '
Message[5] := 'Advisor: Dr Federico Guedea Elizalde'
Message[6] := ' '
Message[7] := 'Master Degree in Automation and Control '
Message[8] := ' '
Message[9] := 'Instituto Tecnologico de Estudios Superiores de Monterrey'
disp_message (WindowHandle3D, Message, 'window', 12, 12, 'white', 'false')
disp_continue_message (WindowHandle3D, 'black', 'true')
stop ()


*Part 2 Detection of points of interest
Angle := rad(45)
Size := 4
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_open_window (5, 0, 748 / 2, 250, 'black', WindowHandle3)
dev_open_window (5, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle4)
dev_open_window (5, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle5)
dev_open_window (320, 0, 748 / 2, 250, 'black', WindowHandle6)
dev_open_window (320, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle7)
dev_open_window (320, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle8)
dev_open_window (125, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 125, 'black',
WindowHandle9)
dev_open_window (260, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 60, 'black', Win-
dowHandle18)
set_display_font (WindowHandle9, 16, 'mono', 'true', 'false')


* Original Image
dev_set_window (WindowHandle3)
dev_clear_window ()
dev_display (Image2)
disp_message (WindowHandle3, 'Original Image', 'window', 12, 12, 'black', 'true')
```

```
* Sojka interest points detector
dev_set_window (WindowHandle4)
rectangle1_domain (Image, ImageReduced, 235, 390, 780, 830)
count_seconds (S1)
points_sojka (ImageReduced, 9, 2.5, 0.75, 30, 90, 0.5, 'true', Row, Col)
count_seconds (S2)
gen_cross_contour_xld (Cross, Row, Col, Size, Angle)
dev_display (Image)
dev_set_color ('yellow')
dev_display (Cross)
disp_message (WindowHandle4, 'Sojka interest points detector'+' '+—Row—+' '+'points found in'+' '+(1000 * (S2 - S1))$'.3' + ' ms', 'window', 12, 12, 'black', 'true')
```

```
* Lepetit interest points detector
dev_set_window (WindowHandle5)
rectangle1_domain (Image1, ImageReduced, 235, 390, 780, 830)
count_seconds (S3)
points_lepetit (ImageReduced, 3, 1, 20, 35, 'interpolation', RowLepetit, ColLepetit)
count_seconds (S4)
gen_cross_contour_xld (CrossLepetit, RowLepetit, ColLepetit, Size, Angle)
dev_display (Image1)
dev_set_color ('yellow')
dev_display (CrossLepetit)
disp_message (WindowHandle5, 'Lepetit interest points detector'+' '+—RowLepetit—+' '+'points found in'+' '+(1000 * (S4 - S3))$'.3' + ' ms', 'window', 12, 12, 'black', 'true')
```

```
* Foerstner interest points detector
dev_set_window (WindowHandle6)
rectangle1_domain (Image3, ImageReduced, 235, 390, 780, 830)
count_seconds (S5)
points_foerstner (ImageReduced, 1, 2, 3, 200, 0.3, 'gauss', 'true', RowJunctions, ColJunctions, CoRRJunctions, CoRCJunctions, CoCCJunctions, RowArea, ColArea, CoRRArea, CoRCArea, CoCCArea)
count_seconds (S6)
gen_cross_contour_xld (Junctions, RowJunctions, ColJunctions, Size, Angle)
dev_display (Image3)
dev_set_color ('yellow')
dev_display (Junctions)
disp_message (WindowHandle6, 'Foerstner interest points detector'+' '+—RowJunctions—+' '+'points found in'+' '+(1000 * (S6 - S5))$'.3' + ' ms', 'window', 12, 12, 'black', 'true')
```

```
* Harris interest points detector
dev_set_window (WindowHandle7)
rectangle1_domain (Image4, ImageReduced, 235, 390, 780, 830)
count_seconds (S7)
points_harris (ImageReduced, 0.7, 2, 0.04, 0, RowHarris, ColHarris)
count_seconds (S8)
gen_cross_contour_xld (CrossHarris, RowHarris, ColHarris, Size, Angle)
dev_display (Image4)
dev_set_color ('yellow')
dev_display (CrossHarris)
disp_message (WindowHandle7, 'Harris interest points detector'+' '+—RowHarris—+' '+'points
found in'+' '+(1000 * (S8 - S7))$'.3' + ' ms', 'window', 12, 12, 'black', 'true')


* Harris binomial interest points detector
dev_set_window (WindowHandle8)
rectangle1_domain (Image5, ImageReduced, 235, 390, 780, 830)
count_seconds (S9)
points_harris_binomial (ImageReduced, 5, 15, 0.04, 1000, 'on', RowHarrisBinomial, ColHarrisBi-
nomial)
count_seconds (S10)
gen_cross_contour_xld (CrossHarrisBinom, RowHarrisBinomial, ColHarrisBinomial, Size, Angle)
dev_display (Image5)
dev_set_color ('yellow')
dev_display (CrossHarrisBinom)
disp_message (WindowHandle8, 'Harris binomial interest points detector'+' '+—RowHarrisBino-
mial—+' '+'points found in'+' '+(1000 * (S10 - S9))$'.3' + ' ms', 'window', 12, 12, 'black', 'true')
Message1 := 'DETECTION OF '
Message1[1] := 'POINTS OF'
Message1[2] := 'INTEREST'
disp_message (WindowHandle9, Message1, 'window', 12, 12, 'white', 'false')
disp_continue_message (WindowHandle18, 'black', 'true')
stop ()


* Part 3 Detection of edges part 1
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_close_window ()
dev_open_window (5, 0, 748 / 2, 250, 'black', WindowHandle10)
dev_open_window (5, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle11)
dev_open_window (5, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle12)
dev_open_window (320, 0, 748 / 2, 250, 'black', WindowHandle13)
```

dev_open_window (320, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle14)
dev_open_window (320, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle15)
dev_open_window (125, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 125, 'black', WindowHandle16)
dev_open_window (260, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 60, 'black', WindowHandle17)
set_display_font (WindowHandle16, 16, 'mono', 'true', 'false')

* Prewitt operator
dev_set_window (WindowHandle10)
rectangle1_domain (Image11, ImageReduced, 235, 390, 780, 830)
prewitt_amp (ImageReduced, ImageEdgeAmp)
threshold (ImageEdgeAmp, Region0, 20, 255)
skeleton (Region0, Skeleton0)
dev_display (Image11)
dev_set_color ('red')
dev_display (Skeleton0)
disp_message (WindowHandle10, 'Prewitt Operator', 'window', 12, 12, 'black', 'true')

* Sobel operator
dev_set_window (WindowHandle11)
rectangle1_domain (Image6, ImageReduced, 235, 390, 780, 830)
sobel_amp (ImageReduced, EdgeAmplitude, 'sum_abs', 3)
threshold (EdgeAmplitude, Region1, 20, 255)
skeleton (Region1, Skeleton1)
dev_display (Image6)
dev_set_color ('red')
dev_display (Skeleton1)
dev_set_color ('yellow')
disp_message (WindowHandle11, 'Sobel Operator', 'window', 12, 12, 'black', 'true')

* Robinson operator
dev_set_window (WindowHandle12)
rectangle1_domain (Image7, ImageReduced, 235, 390, 780, 830)
robinson_amp (ImageReduced, ImageEdgeAmp)
threshold (ImageEdgeAmp, Region5, 10, 255)
skeleton (Region5, Skeleton5)
dev_display (Image7)
dev_set_color ('red')
dev_display (Skeleton5)
disp_message (WindowHandle12, 'Robinson Operator', 'window', 12, 12, 'black', 'true')

* Robert operator
dev_set_window (WindowHandle13)
rectangle1_domain (Image8, ImageReduced, 235, 390, 780, 830)
roberts (ImageReduced, ImageRoberts, 'roberts$_max'$)
$threshold(ImageRoberts, Region2, 15, 255)$
$skeleton(Region2, Skeleton2)$
$dev\_display(Image8)$
$dev\_set\_color('red')$
$dev\_display(Skeleton2)$
$disp\_message(WindowHandle13,'RobertOperator','window', 12, 12,'black','true')$

* Frei-Chen operator
dev_set_window (WindowHandle14)
rectangle1_domain (Image9, ImageReduced, 235, 390, 780, 830)
frei_amp (ImageReduced, ImageEdgeAmp)
threshold (ImageEdgeAmp, Region3, 40, 255)
skeleton (Region3, Skeleton3)
dev_display (Image9)
dev_set_color ('red')
dev_display (Skeleton3)
disp_message (WindowHandle14, 'Frei-Chen Operator', 'window', 12, 12, 'black', 'true')

* Kirsch operator
dev_set_window (WindowHandle15)
rectangle1_domain (Image10, ImageReduced, 235, 390, 780, 830)
kirsch_amp (ImageReduced, ImageEdgeAmp)
threshold (ImageEdgeAmp, Region4, 60, 255)
skeleton (Region4, Skeleton4)
dev_display (Image10)
dev_set_color ('red')
dev_display (Skeleton4)
disp_message (WindowHandle15, 'Kirsch Operator', 'window', 12, 12, 'black', 'true')
Message1 := 'DETECTION'
Message1[1] := 'OF'
Message1[2] := 'EDGES'
disp_message (WindowHandle16, Message1, 'window', 12, 12, 'white', 'false')
disp_continue_message (WindowHandle17, 'black', 'true')
stop ()

```
* Detection of edges part 2
dev_open_window (5, 0, 748 / 2, 250, 'black', WindowHandle101)
dev_open_window (5, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle111)
dev_open_window (5, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle121)
dev_open_window (320, 0, 748 / 2, 250, 'black', WindowHandle131)
dev_open_window (320, 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle141)
dev_open_window (320, 748 / 2 + 12 + 748 / 2 + 12, 748 / 2, 250, 'black', WindowHandle151)
dev_open_window (125, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 125, 'black',
WindowHandle161)
dev_open_window (260, 748 / 2 + 12 + 748 / 2 + 12 + 748 / 2 + 12, 748 / 4, 60, 'black', Win-
dowHandle171)
set_display_font (WindowHandle161, 16, 'mono', 'true', 'false')


* Edge Detection with Lanser Filter
dev_set_window (WindowHandle101)
rectangle1_domain (Image11, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'lanser1', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image11)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle101, 'Edge Detection with Lanser Filter', 'window', 12, 12, 'black',
'true')


* Edge Detection with Lanser Filter
dev_set_window (WindowHandle111)
rectangle1_domain (Image6, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'lanser2', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image6)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle111, 'Edge Detection with Lanser Filter', 'window', 12, 12, 'black',
'true')


* Edge Sub Pix Detection with Lanser Filter
dev_set_window (WindowHandle121)
rectangle1_domain (Image7, ImageReduced, 235, 390, 780, 830)
edges_sub_pix (ImageReduced, Edges, 'lanser1', 2, 12, 22)
dev_display (Image7)
dev_set_color ('red')
```

```
dev_display (Edges)
disp_message (WindowHandle121, 'Edge Sub Pix Detection with Lanser Filter', 'window', 12, 12,
'black', 'true')


* Edge Detection with Shen Filter
dev_set_window (WindowHandle131)
rectangle1_domain (Image8, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'shen', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image8)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle131, 'Edge Detection with Shen Filter', 'window', 12, 12, 'black',
'true')


* Edge Detection with Shen Filter
dev_set_window (WindowHandle141)
rectangle1_domain (Image9, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'mshen', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image9)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle141, 'Edge Detection with Shen Filter', 'window', 12, 12, 'black',
'true')


* Edge Sub Pix Detection with Shen Filter
dev_set_window (WindowHandle151)
rectangle1_domain (Image9, ImageReduced, 235, 390, 780, 830)
edges_sub_pix (ImageReduced, Edges, 'shen', 2, 12, 22)
dev_display (Image9)
dev_set_color ('red')
dev_display (Edges)
disp_message (WindowHandle151, 'Edge Sub Pix Detection with Shen Filter', 'window', 12, 12,
'black', 'true')
```

* Detection of edges part 3
* Edge Detection with Deriche Filter
dev_set_window (WindowHandle102)
rectangle1_domain (Image11, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'deriche1', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image11)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle102, 'Edge Detection with Deriche Filter', 'window', 12, 12, 'black', 'true')

* Edge Detection with Deriche Filter
dev_set_window (WindowHandle112)
rectangle1_domain (Image6, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'deriche2', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton$_(Skeleton, Contours, 20, 'filter')$
$dev\_display(Image6)$
$dev\_set\_color('red')$
$dev\_display(Contours)$
$disp\_message(WindowHandle112, 'EdgeDetectionwithDerichеFilter', 'window', 12, 12, 'black', 'true')$

* Edge Sub Pix Detection with Deriche Filter
dev_set_window (WindowHandle122)
rectangle1_domain (Image7, ImageReduced, 235, 390, 780, 830)
edges_sub_pix (ImageReduced, Edges, 'deriche2', 2, 12, 22)
dev_display (Image7)
dev_set_color ('red')
dev_display (Edges)
disp_message (WindowHandle122, 'Edge Sub Pix Detection with Deriche Filter', 'window', 12, 12, 'black', 'true')

* Edge Detection with Canny Filter
dev_set_window (WindowHandle132)
rectangle1_domain (Image8, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'canny', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image8)
dev_set_color ('red')

dev_display (Contours)
disp_message (WindowHandle132, 'Edge Detection with Canny Filter', 'window', 12, 12, 'black', 'true')

* Edge Sub Pix Detection with Canny Filter
dev_set_window (WindowHandle142)
rectangle1_domain (Image9, ImageReduced, 235, 390, 780, 830)
edges_sub_pix (ImageReduced, Edges, 'canny', 2, 12, 22)
dev_display (Image9)
dev_set_color ('red')
dev_display (Edges)
disp_message (WindowHandle142, 'Edge Sub Pix Detection with Canny Filter', 'window', 12, 12, 'black', 'true')

* Edge Detection with Sobel Filter
dev_set_window (WindowHandle152)
rectangle1_domain (Image9, ImageReduced, 235, 390, 780, 830)
edges_image (ImageReduced, ImaAmp, ImaDir, 'sobel_fast', 2, 'nms', 12, 22)
threshold (ImaAmp, Edges, 1, 255)
skeleton (Edges, Skeleton)
gen_contours_skeleton_xld (Skeleton, Contours, 20, 'filter')
dev_display (Image9)
dev_set_color ('red')
dev_display (Contours)
disp_message (WindowHandle152, 'Edge Detection with Sobel Filter', 'window', 12, 12, 'black', 'true')

*Part 3 Detection of lines using Hough Transform
* Hough line operator
dev_set_window (WindowHandle19)
gray_dilation_rect (Image13, Dilation, 11, 11)
gray_erosion_rect (Dilation, Closing, 11, 11)
rectangle1_domain (Closing, ImageReduced, 235, 390, 780, 830)
sobel_amp (ImageReduced, EdgeAmplitude, 'thin_sum_abs', 3)
dev_set_color ('red')
threshold (EdgeAmplitude, Region, 25, 255)
hough_lines (Region, 4, 90, 5, 5, Angle, Dist)
dev_display(Image12)
dev_display(Region)
dev_set_color ('blue')
gen_region_hline (Regions, Angle, Dist)
dev_display(Regions)
disp_message (WindowHandle19, 'Hough Line Operator', 'window', 12, 12, 'black', 'true')

```
* Hough line operator using local gradient
dev_set_window (WindowHandle20)
gray_dilation_rect (Image13, Dilation, 11, 11)
gray_erosion_rect (Dilation, Closing, 11, 11)
rectangle1_domain (Closing, ImageReduced, 235, 390, 780, 830)
sobel_dir (ImageReduced, EdgeAmplitude, EdgeDirection, 'sum_abs', 3)
dev_set_color ('red')
threshold (EdgeAmplitude, Region, 27, 255)
reduce_domain (EdgeDirection, Region, EdgeDirectionReduced)
hough_lines_dir (EdgeDirectionReduced, HoughImage, Lines, 4, 2, 'mean', 3, 25, 5, 5, 'true', Angle,
Dist)
gen_region_hline (LinesHNF, Angle, Dist)
dev_display (Image13)
dev_set_colored (12)
dev_set_draw ('margin')
dev_display (LinesHNF)
dev_set_draw ('fill')
dev_display (Lines)
disp_message (WindowHandle20, 'Hough Line Operator using local gradient', 'window', 12, 12,
'black', 'true')


*Part 4 Segmentation using Region Part 1
* Segmentation using Region
dev_set_window (WindowHandle23)
median_image (Image11, ImageMedian, 'circle', 2, 'mirrored')
regiongrowing (ImageMedian, Regions, 1, 1, 2, 100)
fill_up_shape (Regions, RegionFillUp, 'area', 1, 100)
closing_circle (RegionFillUp, RegionClosing, 7.5)
dev_display(Image11)
dev_set_draw ('fill')
dev_set_colored (12)
dev_display (RegionClosing)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (RegionClosing)
disp_message (WindowHandle23, 'Segmentation using Region', 'window', 12, 12, 'black', 'true')


* Segmentation using Region
dev_set_window (WindowHandle24)
median_image (Image11, ImageMedian, 'circle', 2, 'mirrored')
regiongrowing (ImageMedian, Regions, 1, 1, 2, 5000)
shape_trans (Regions, Centers, 'inner_center')
connection (Centers, SingleCenters)
area_center (SingleCenters, Area23, Row, Column)
regiongrowing_mean (ImageMedian, RegionsMean, Row, Column, 25, 100)
dev_display(Image11)
```

```
dev_set_draw ('fill')
dev_set_colored (12)
dev_display (RegionsMean)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (RegionsMean)
disp_message (WindowHandle24, 'Segmentation using Region', 'window', 12, 12, 'black', 'true')


* Segmentation using Region
dev_set_window (WindowHandle25)
median_image (Image11, ImageMedian, 'circle', 2, 'mirrored')
regiongrowing_n (ImageMedian, Regions, '1-norm', 0, 2, 30)
dev_display(Image11)
dev_set_draw ('fill')
dev_set_colored (12)
dev_display (Regions)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions)
disp_message (WindowHandle25, 'Segmentation using Region', 'window', 12, 12, 'black', 'true')


* Segmentation using Region
dev_set_window (WindowHandle26)
gray_dilation_rect (Image11, Dilation, 11, 11)
gray_erosion_rect (Dilation, Closing, 11, 11)
rectangle1_domain (Closing, Imag, 235, 390, 780, 830)
median_image (Imag, ImageMedian26, 'square', 10, 'mirrored')
regiongrowing_n (ImageMedian26, Regions26, 'dot-product', 0, 60, 800)
dev_display(Image11)
dev_set_draw ('fill')
dev_set_color ('white')
dev_clear_window ()
dev_display (Regions26)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions26)
disp_message (WindowHandle26, 'Segmentation using Region', 'window', 12, 12, 'black', 'true')


* Segmentation using Contour Part 2
* Segmentation using Contour
dev_set_window (WindowHandle31)
edges_sub_pix (Image, Edges, 'canny', 1.5, 15, 40)
segment_contours_xld (Edges, ContoursSplit, 'lines_circles', 5, 4, 2)
count_obj (ContoursSplit, Number)
gen_empty_obj (Lines)
gen_empty_obj (Circles)
```

```
for I := 1 to Number by 1
select_obj (ContoursSplit, Contour, I)
get_contour_global_attrib_xld (Contour, 'cont_approx', Type)
if (Type == -1)
concat_obj (Lines, Contour, Lines)
else
concat_obj (Circles, Contour, Circles)
endif
endfor
fit_line_contour_xld (Lines, 'tukey', -1, 0, 5, 2, RowBegin, ColBegin, RowEnd, ColEnd, Nr, Nc,
Dist)
fit_circle_contour_xld (Circles, 'atukey', -1, 2, 0, 3, 2, Row, Column, Radius, StartPhi, EndPhi,
PointOrder)
dev_display(Image)
dev_set_draw ('fill')
dev_set_line_width (2)
dev_set_colored (12)
dev_display (Lines)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Circles)
disp_message (WindowHandle31, 'Segmentation using Contour', 'window', 12, 12, 'black', 'true')


* Segmentation using Contour
dev_set_window (WindowHandle32)
regiongrowing (Image, Regions, 1, 1, 2, 5000)
dev_set_colored (12)
dev_clear_window ()
gen_contour_region_xld (Regions, Contours, 'border')
dev_display(Image)
dev_display(Contours)
disp_message (WindowHandle32, 'Segmentation using Contour', 'window', 12, 12, 'black', 'true')


* Segmentation using Contour
dev_set_window (WindowHandle33)
edges_sub_pix (Image, Edges, 'canny', 1.1, 22, 30)
segment_contours_xld (Edges, LineSegments, 'lines_circles', 5, 4, 2)
regress_contours_xld (LineSegments, RegressContours, 'no', 1)
union_collinear_contours_xld (RegressContours, UnionContours, 10, 1, 2, 0.1, 'attr_keep')
sort_contours_xld (UnionContours, SortedContours, 'upper_left', 'true', 'column')
dev_set_draw ('margin')
dev_set_line_width (2)
dev_display (Image)
dev_set_colored (12)
```

```
dev_display (Edges)
dev_display (SortedContours)
disp_message (WindowHandle33, 'Segmentation using Contour', 'window', 12, 12, 'black', 'true')


* Showing images
dev_open_window (5, 0, 324, 200, 'black', WindowHandle23a)
dev_open_window (5, 324 + 12, 324, 200, 'black', WindowHandle24a)
dev_open_window (5, 324 + 12 + 324 + 12, 324, 200, 'black', WindowHandle25a)
dev_open_window (370, 0, 324, 200, 'black', WindowHandle26a)
dev_open_window (370, 324 + 12, 324, 200, 'black', WindowHandle27a)
dev_open_window (370, 324 + 12 + 324 + 12, 324, 200, 'black', WindowHandle28a)
dev_open_window (5, 324 + 12 + 324 + 12 + 324 + 12, 324, 200, 'black', WindowHandle29a)
dev_open_window (370, 324 + 12 + 324 + 12 + 324 + 12, 324, 200, 'black', WindowHandle30a)
dev_open_window (270, 324 + 12, 324, 40, 'black', WindowHandle31a)
dev_open_window (270, 324 + 12 + 324 + 12, 324, 40, 'black', WindowHandle32a)
set_display_font (WindowHandle31a, 16, 'mono', 'true', 'false')
dev_set_window (WindowHandle23a)
read_image (Image1a, 'proyecto_recons3d/ci1')
dev_display(Image1a)
disp_message (WindowHandle23a, 'Left Image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle24a)
read_image (Image2a, 'proyecto_recons3d/cd1')
dev_display(Image2a)
disp_message (WindowHandle24a, 'Right Image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle26a)
read_image (Image3a, 'proyecto_recons3d/c3')
dev_display(Image3a)
disp_message (WindowHandle26a, 'Image with pattern', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle27a)
read_image (Image4a, 'proyecto_recons3d/c4')
dev_display(Image4a)
disp_message (WindowHandle27a, 'Image with pattern', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle25a)
read_image (Image5a, 'proyecto_recons3d/c1')
dev_display(Image5a)
disp_message (WindowHandle25a, 'Image with pattern', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle29a)
read_image (Image6a, 'proyecto_recons3d/c2')
dev_display(Image6a)
disp_message (WindowHandle29a, 'Image with pattern', 'window', 12, 12, 'black', 'true')


* First Step for Reconstruction for the first object
dev_set_window (WindowHandle39)
read_image (Image, 'proyecto_recons3d/Im3d')
dev_display(Image)
disp_message (WindowHandle39, 'Right Image', 'window', 12, 12, 'black', 'true')
```

```
dev_set_window (WindowHandle40)
gray_dilation_rect (Image, Dilation, 11, 11)
gray_erosion_rect (Dilation, Closing, 11, 11)
rectangle1_domain (Closing, Image, 235, 390, 780, 830)
median_image (Closing, ImageMedian2, 'square', 10, 'continued')
regiongrowing_n (ImageMedian2, Regions2, 'dot-product', 0, 60, 800)
dev_display(Image)
dev_set_draw ('fill')
dev_set_color ('white')
dev_clear_window ()
dev_display (Regions2)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions2)
disp_message (WindowHandle40, 'Right Segmented Image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle41)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions2)
rectangle1_domain (Closing, ImageReduced, 235, 200, 780, 640)
points_sojka (ImageReduced, 9, 2.5, 0.75, 15, 40, 0.5, 'true', Row, Col)
gen_cross_contour_xld (Cross, Row, Col,4, rad(45))
dev_display(Image)
dev_set_draw ('fill')
dev_set_color ('white')
dev_clear_window ()
dev_display (Regions2)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions2)
dev_set_line_width (5)
dev_set_color ('blue')
dev_display (Cross)
disp_message (WindowHandle41, 'Right Characteristic Points', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle42)
read_image (Image1, 'proyecto_recons3d/Im3i')
dev_display(Image1)
disp_message (WindowHandle42, 'Left Image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle43)
gray_dilation_rect (Image1, Dilation1, 11, 11)
gray_erosion_rect (Dilation1, Closing1, 11, 11)
rectangle1_domain (Closing1, Image1, 235, 390, 780, 830)
median_image (Closing1, ImageMedian21, 'square', 10, 'continued')
regiongrowing_n (ImageMedian21, Regions21, 'dot-product', 0, 60, 800)
dev_display(Image1)
dev_set_draw ('fill')
dev_set_color ('white')
```

```
dev_clear_window ()
dev_display (Regions21)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions21)
disp_message (WindowHandle43, 'Left Segmented Image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle44)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions21)
rectangle1_domain (Closing1, ImageReduced1, 235, 390, 780, 830)
points_sojka (ImageReduced1, 9, 2.5, 0.75, 15, 50, 0.5, 'true', Row1, Col1)
gen_cross_contour_xld (Cross1, Row1, Col1,4, rad(45))
dev_display(Image1)
dev_set_draw ('fill')
dev_set_color ('white')
dev_clear_window ()
dev_display (Regions21)
dev_set_color ('black')
dev_set_draw ('margin')
dev_display (Regions21)
dev_set_line_width (5)
dev_set_color ('blue')
dev_display (Cross1)
disp_message (WindowHandle44, 'Left Characteristic Points', 'window', 12, 12, 'black', 'true')
stop ()
```

```
* Second Step for Reconstruction for the second object
Path := 'stereo/epipolar/'
read_image (Image1, Path + 'C_l')
rectangle1_domain (Image1, LImageReduced, 235, 390, 780, 830)
read_image (Image2, Path + 'C_r')
rectangle1_domain (Image2, RImageReduced, 235, 390, 780, 830)
dev_set_window (WindowHandle63)
dev_display (Image1)
disp_message (WindowHandle63, 'Left image', 'window', 12, 12, 'black', 'true')
dev_set_window (WindowHandle65)
dev_display (Image2)
disp_message (WindowHandle65, 'Right image', 'window', 12, 12, 'black', 'true')
disp_continue_message (WindowHandle63, 'black', 'true')
stop ()
binocular_disparity (LImageReduced, RImageReduced, Disparity1, Score1, 'sad', 11, 11, 0, -30, 30,
2, 30, 'left_right_check', 'interpolation')
dev_open_window (5, 748 / 2 + 12, 700, 565, 'gray', WindowHandle64)
set_display_font (WindowHandle64, 14, 'mono', 'true', 'false')
dev_set_window (WindowHandle64)
```

```
dev_display (Disparity1)
disp_message (WindowHandle64, 'Disparity image', 'window', 12, 12, 'black', 'true')
* Extract the object from the background
min_max_gray (Disparity1, Disparity1, 0, Min, Max, Range)
scale_image (Disparity1, Disparity1, Max * 255 / (Max - Min), Min)
convert_image_type (Disparity1, ByteDisparity, 'byte')
bin_threshold (ByteDisparity, SegmRec)
smallest_rectangle1(SegmRec, Row1, Column1, Row2, Column2)
* The green rectangle marks the segmented foreground
gen_rectangle1(Rect, Row1, Column1, Row2, Column2)
dev_display (Rect)
dev_set_window (WindowHandle63)
dev_display (Image1)
dev_display (Rect)
disp_message (WindowHandle63, 'Left image', 'window', 12, 12, 'black', 'true')
dev_display (Rect)
dev_set_window (WindowHandle65)
dev_display (Image2)
dev_display (Rect)
disp_message (WindowHandle65, 'Right image', 'window', 12, 12, 'black', 'true')
disp_continue_message (WindowHandle66, 'black', 'true')
stop ()
* The images to be read:
Path := 'stereo/epipolar/'
read_image (LImage, 'proyecto_recons3d/Im3i')
rectangle1_domain (LImage, LImageReduced, 255, 400, 760, 825)
read_image (RImage, 'proyecto_recons3d/Im3d')
rectangle1_domain (RImage, RImageReduced, 255, 210, 760, 635)
binocular_disparity (LImage, RImageReduced, DisparityNCC, Score, 'ncc', 11, 11, 0, -45, 10, 3,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC1, Score, 'ncc', 19, 19, 5, -45, -10, 2,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC2, Score, 'ncc', 25, 25, 5, -25, 30, 2,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC3, Score, 'ncc', 5, 5, 5, -12, -8, 2, 0.3,
'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC4, Score, 'ncc', 11, 11, 5, -50, -20, 2,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC5, Score, 'ncc', 6, 6, 5, -30, -15, 2,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC6, Score, 'ncc', 30, 30, 5, -10, -10, 2,
0.3, 'left_right_check', 'interpolation')
binocular_disparity (LImage, RImageReduced, DisparityNCC7, Score, 'ncc', 7, 7, 5, -60, -10, 2,
0.3, 'left_right_check', 'interpolation')
dev_set_window (WindowHandle3b)
dev_display (DisparityNCC)
disp_message (WindowHandle3b, 'Points Cloud 1', 'window', 12, 12, 'black', 'true')
```

```
dev_close_window ()
dev_update_off ()
Sequence := [1:3]
Names := 'dff/cal_' + (Sequence$'.2')
read_image (Image, Names)
channels_to_image (Image, Image)
Width := 640
Height := 480
dev_open_window (5, 0, 748 / 2, 250, 'black', WindowHandle55)
for I := 1 to 3 by 1
access_channel (Image, Image1, I)
dev_display (Image1)
disp_message (WindowHandle55, 'Focus series (' + I + '/10)', 'window', 12, 12, 'black', 'true')
wait_seconds (0.5)
endfor
disp_continue_message (WindowHandle55, 'black', 'true')
disp_continue_message (WindowHandle55, 'black', 'true')
stop ()
dev_open_window (320, 0, 748 / 2, 250, 'black', WindowHandle57)
depth_from_focus (Image, Depth, Confidence, 'highpass', 'next_maximum')
* Construct sharp image
mean_image (Depth, DepthHighConf, 51, 51)
select_grayvalues_from_channels (Image, DepthHighConf, SharpImage)
* Smooth depth map
scale_image_max (DepthHighConf, ImageScaleMax)
mean_image (ImageScaleMax, DepthMean, 51, 51)
dev_display (SharpImage)
disp_message (WindowHandle57, 'Sharp image', 'window', 12, 12, 'black', 'true')
disp_continue_message (WindowHandle57, 'black', 'true')
stop ()
* 3D reconstruction
dev_open_window (5, 748 / 2 + 12, 700, 565, 'black', WindowHandle56)
dev_set_paint (['3d_plot','texture'])
compose2 (DepthMean, SharpImage, TexturedImage)
dev_display (TexturedImage)
disp_message (WindowHandle56, 'Textured depth image', 'window', 12, 12, 'black', 'true')
```

A.3 Appendix III: HALCON Configuration

A.3.1 HALCON

A.3.1.1 What is 3D Vision?

3D vision means the utilization of 3D information with the aid of machine vision, allowing to approach applications which so far could not be solved with classical 2D technologies.It includes two main objectives, which both contain many different technologies:

- 3D reconstruction – determining the 3D shape of arbitrary objects.
- 3D alignment – finding the 3D pose (position & orientation) of an object.

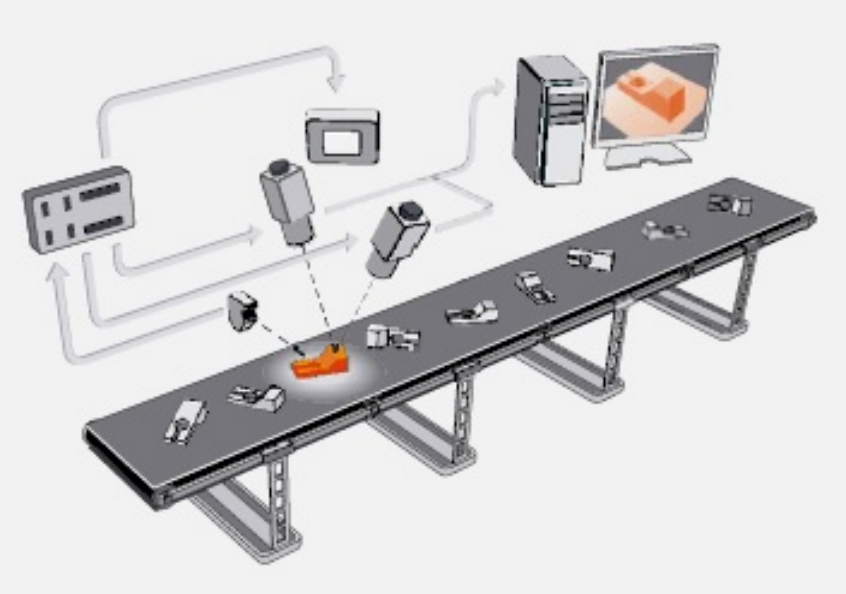


Figure A.4: The 3D shape of an object is reconstructed from the images of a two camera setup

A.3.1.2 3D Vision Technologies

A.3.1.2.1 3D Calibration

Introduction of the technology: With 3D calibration, you establish the relationship between a camera and the object. For robotics applications, additionally the relationship between the robot and the camera is determined. With this technology, you get an explicit and accurate description of your area or line scan camera: A set of so-called internal and external camera parameters map the image coordinates to real world coordinates.

Particularly suitable for The correction of lens and perspective distortions. This can be realized by rectifying the image, or by measuring in the distorted image and correcting the measuring result. Furthermore, measuring results can be determined in world coordinates and the geometric

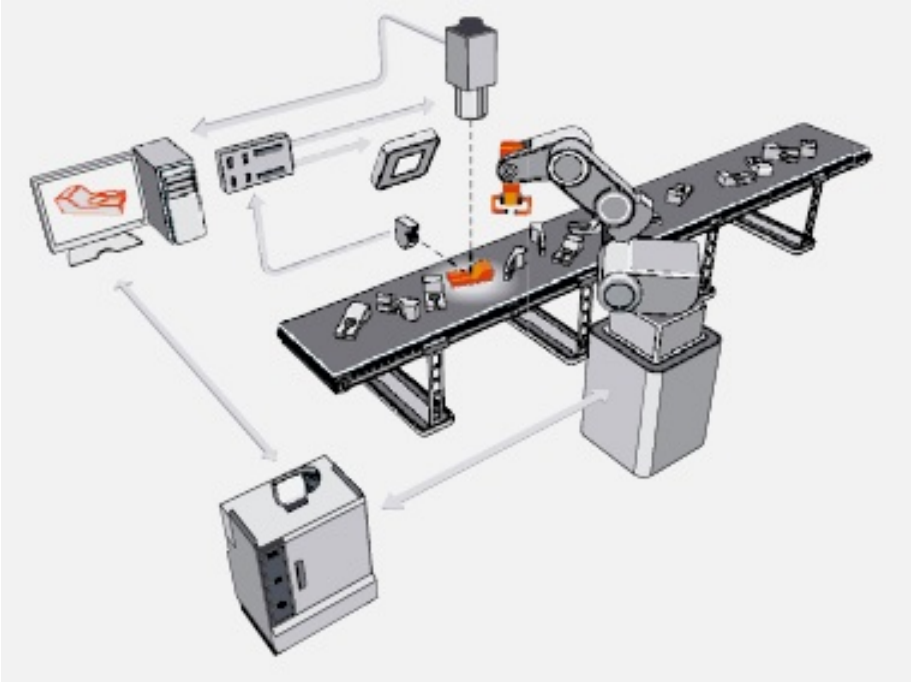


Figure A.5: Using a camera image and various image processing functions, the 3D pose of an object is determined. This data is used to control the robot

relationship between the camera and the object can be calculated by the calibration – crucial for robotics applications. All 3D applications require this technology. Typical examples include bin picking as well as stereo applications.

Supported by HALCON: HALCON's 3D calibration supports area and line scan cameras, as well as cameras with telecentric lenses. 3D calibration works also with multiple cameras (multi-view 3D calibration). Usually, a specific calibration object is used for calibration (e.g., calibration plate). Alternatively, self-calibration, without the need of a calibration object, can be used. HALCON's 3D calibration permits, subpixel-accurate measurements up to 1 um in a field of view of 10 mm.

A.3.1.2.2 Stereo Vision

Introduction of the technology: The 3D coordinates of the visible points on the object surface can be determined based on two or more images that are acquired from different points of view. This is done by calculating the disparity map of the calibrated camera setup.

Particularly suitable for 3D reconstruction, i.e. determining the 3D shape of arbitrary objects, especially useful for mid- and large-sized textured objects. It can be used for quality inspection of 3D objects or for the position recognition of 3D objects. Furthermore, stereo vision can be a pre-processing step for 3D matching.

Supported by HALCON: HALCON supports stereo vision by calculating the 3D coordinates on the object surface. For a stationary object, this can be done with the aid of a two camera setup (binocular) or multiple cameras (multi-view). Using only one camera, the 3D object shape can be computed from the relative motion between camera and object. Stereo vision is realized either by calculating dense 3D coordinates or distance images, or by determining the coordinates for specific

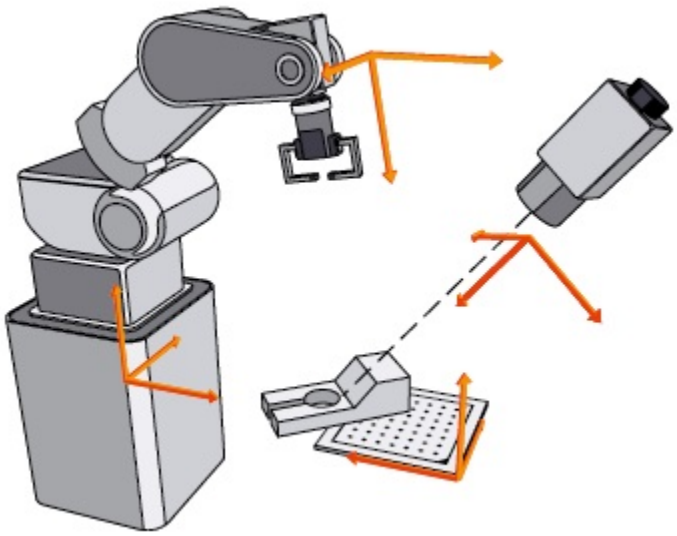


Figure A.6: The relationship between camera, robot, and object is established with the help of a calibration plate

points or edges – especially suitable for highly accurate height measuring. Furthermore, HALCON offers multigrid stereo – an advanced method to interpolate the 3D data in homogeneous image parts. This method yields higher accuracy for small objects.

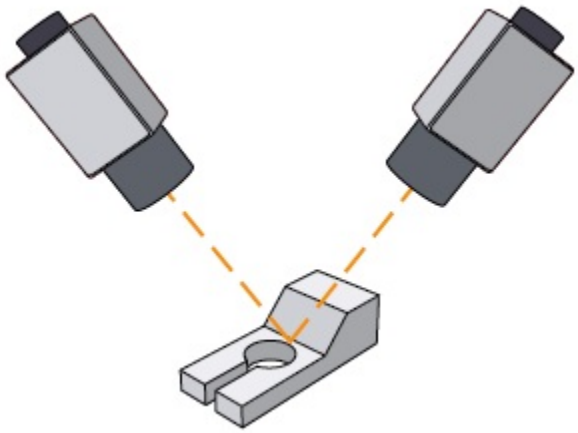


Figure A.7: Two or more cameras acquire images from different points of view. The disparity map of these images is calculated and based on this, the 3D shape of the object is determined.

A.3.1.2.3 Monocular 2 1/2 D - Depth from Focus & Sheet of Light

Introduction of the technology: There are various so-called active technologies for the extraction of height information with one camera. The resulting 3D information is very similar to binocular stereo. The most often used methods are:

- Depth from focus (DFF) extracts distance information by calculating the focus of all pixels of the image. A small depth of field is used to calculate the distance of the object's surface to the camera.

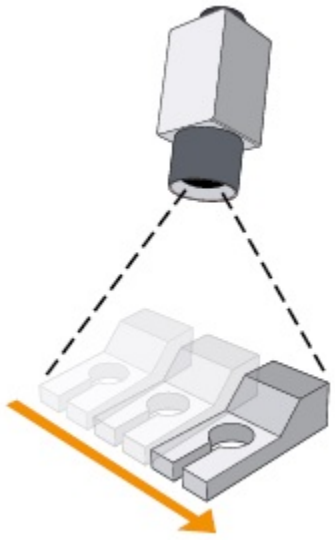


Figure A.8: One camera captures overlapping images of a moving object. Based on this, the 3D shape of the object is determined

- Sheet of light means measuring an elevation profile of an object by reconstructing the projected line of light on this object.

Particularly suitable for 3D reconstruction – in case of DFF especially suitable for small objects, in case of sheet of light for objects without texture. Typical application examples include quality inspection of 3D objects as well as position recognition of 3D objects.

Supported by HALCON: For dense height maps, HALCON offers various methods that can be used to process 2 1/2 D images, e.g., to determine object edges or angles between 3D planes. In the case of sheet of light, also highly accurate line- or point-oriented 3D measurements can be applied.

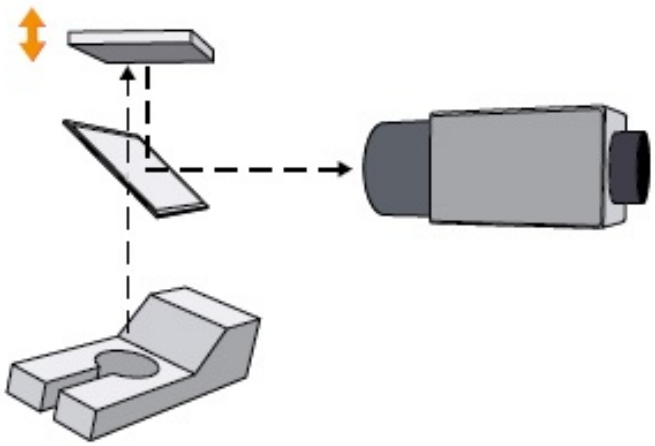


Figure A.9: Several images of the object are taken with different distances. The change in focus is used to calculate distance information

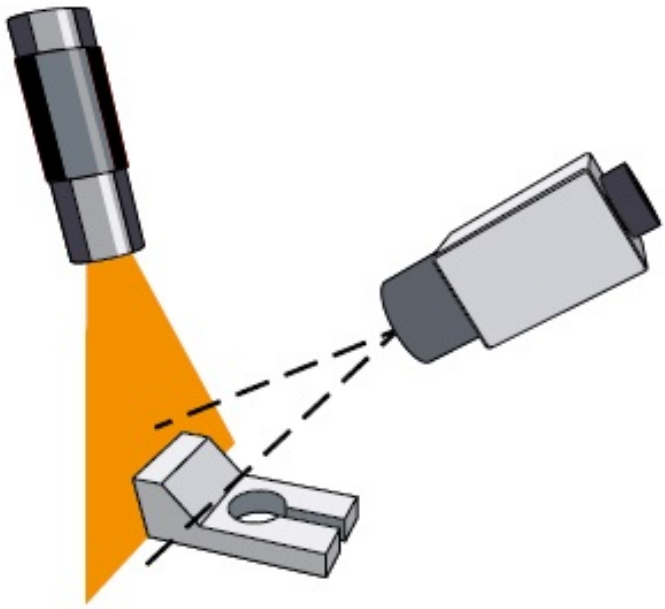


Figure A.10: The 3D shape of an object is determined by measuring the profile of the object along a projected line of light

A.3.1.2.4 3D Registration

Introduction of the technology: 3D sensors directly capture the 3D coordinates of the visible points on the object surface. Using multiple 3D images of the object acquired from different points of view, the pairwise overlap of the 3D data is computed and a globally optimized shape is derived.

Particularly suitable for 3D reconstruction, i.e. determining the 3D shape of arbitrary objects. The 3D object shape can be used in 3D matching, for quality inspection of 3D objects, or for the position recognition of 3D objects.

Supported by HALCON: HALCON supports the direct derivation of the 3D object shape from 3D sensor data using 3D registration. Furthermore, HALCON offers advanced 3D shape processing techniques like smoothing, subsampling, and triangulation to prepare the shape model for 3D matching and object comparison.

A.3.1.2.5 Perspective Matching

Introduction of the technology: Instead of using the full 3D shape of an object, for many applications it is possible to restrict the model area to a planar part of the object. For arbitrarily shaped object parts, perspective matching allows to determine the 3D pose with only one camera. The model generation is done by training a sample image of the object typically inside a specified ROI.

Particularly suitable for 3D alignment, e.g., applications for which the 3D pose (position and orientation) of an object must be found. Examples are automotive and robotics applications, pick-and-place applications, and bin picking. A further possibility is the measuring of geometric features on complex 3D objects after 3D alignment.

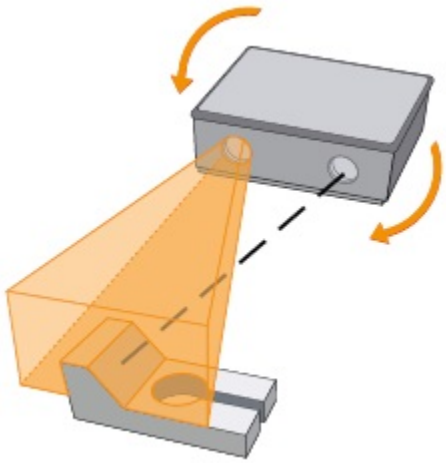


Figure A.11: Using 3D registration, the 3D object shape is determined from multiple views with a 3D sensor

Supported by HALCON: HALCON's perspective matching helps finding objects easily with only one camera. For perspective matching, HALCON provides two different methods suitable for two different classes of objects. Depending on the object's shape and appearance, HALCON offers both deformable matching, which is based on the shape-based matching technology (object edges), and descriptor-based matching, which uses so-called interest points.

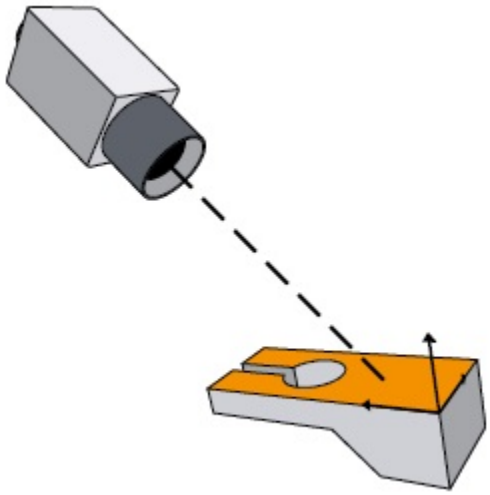


Figure A.12: Only a planar part of the object is necessary to determine the 3D pose of the object by exploiting perspective matching techniques

A.3.1.3 A detailed comparison of HALCON vs. Cognex VisionPro

Table A.1: Executive Summary

	Strengths	Weaknesses
MVTec HALCON	A significantly larger and powerful 2D and full scope 3D machine vision library at a significantly lower run time license price. HALCON supports 5 times the number of image acquisition devices, provides higher bit depth image processing, GPU acceleration, support for Windows, MAC OS X, & Linux and several embedded platforms, and ongoing support for COM, .NET, Native C, C#, C++, & Delphi programming. MVTec's only focus is Machine Vision for PC and embedded vision processing with total hardware independence. HALCON has very large and distinct advantages in their 3D vision technology and application capabilities over Cognex VisionPro	Small market share in North America, longer learning curve for non-programmers, simpler applications can take longer to deploy, higher cost Software Development Package price than VisionPro, lack of tools for US Postal barcodes
Cognex VisionPro	VisionPro enjoys a much larger market share in the USA, provides an easier to use interface for the non-programmer, low system software development license cost, and barcode tools for US Postal applications (Postnet and IMBD symbologies) The QuickBuild environment in general allows non programmers to deploy applications fairly quickly	Very limited capability 3D machine vision algorithm library, high run time software license costs, low image bit depth support, lack of GPU processing, and a small number of image processing algorithms (for instance: no FFT) The VisionPro QuickBuild environment for advanced vision applications can add unnecessary complexity and you may be better off programming everything in C# or.NET and avoiding the QuickBuild environment altogether

Table A.2: HALCON vs. VisionPro

	HALCON	VisionPro
2D Pattern Matching	HALCON can perform 2D pattern matching on 16 bit images	VisionPro pattern matching is limited to 8 bit image processing
1D and 2D Metrology	HALCON supports 32 bit depth processing	VisionPro performs primarily 8 bit processing, but a small number of tools are 16 bit
Blob analysis	HALCON blob tool is more extensive, much more flexible and powerful	VisionPro's tool is fairly basic and adequate for typical applications

Table A.3: Vision Performance: HALCON vs. VisionPro

	HALCON	VisionPro
Image processing filters	HALCON has upwards of 100 Image Processing filters including FFT (Fast Fourier Transformation) that run at a higher bit depth - Most HALCON filters run at 32 bits	VisionPro has a short list of image processing filters and most are 8 bit and some are 16 bit. VisionPro lacks an FFT (Fast Fourier Transformation) algorithm which is important in inspection applications to remove fixed pattern textures
Image Classifier Tools	HALCON supports a large # of predefined classifier tools for identification and inspection applications. HALCON supports multi-layer perceptron neural net classifier; support vector machine classifier; Gaussian mixture models classifier; clustering with n-dimensional boxes and spheres for data sets with a non-normal distribution; k-nearest neighbors classifier; automatic feature selection	Cognex has an "Inspection Designer" or CogDataAnalysis tool that allows you to create a custom classifier from the results within your application. There are no predefined classifier tools in VisionPro. There is an add-on software module called VisionPro Surface - training algorithms learn the appearance of each class of defect based on the user's visual cues
OCR and OCV	HALCON has a library of pre-trained industrial fonts that can be used without training which is a nice feature	Equivalent performance on OCR applications, VisionPro has an advantage on OCV applications due to an easier to use interface
1D and 2D Barcode tools	Equivalent performance on 1D and 2D barcode symbologies	Cognex enjoys an advantage on the number of barcode grading metrics and also has US Postal Barcode capability
Image Acquisition and features	HALCON supports all of the image acquisition standards - GigE Vision, USB 3 Vision, GeniCam, GenTL, DirectShow, TWAIN. HALCON also has the ability to create HDR (High Dynamic Range) images from standard non-HDR cameras - typically you can get over 100dB of dynamic range using the HALCON algorithms with a typical Basler GigE camera vision camera	VisionPro has support for cameras through their Image Acquisition partners. A camera manufacturer or Cognex must create a custom camera configuration file - also known as a CCF. VisionPro does support generic GeniCam for Gig E Vision devices, but oddly enough does not support generic GeniCam for other frameworks such as CameraLink, CoaXpress, or USB 3
3D Alignment and Guidance	HALCON's 3D vision application capability is the biggest technical advantage over Cognex VisionPro by far. HALCON provides true 3D shape based point cloud pattern matching. HALCON has the ability to search for 3D shapes based upon CAD file import or "golden template" point cloud data from a large variety of 3D image acquisition devices. HALCON uses the entire 3D shape (point cloud) of an object for recognition and guidance. HALCON provides further methods for 3D position recognition that work on images from a single camera such as Shape-based Matching of 3D CAD models as well as methods to find flat objects in 3D based on their perspective deformation	Cognex VisionPro provides pseudo 3D pattern matching by performing triangulation on multiple 2D shapes using PatMax pose data. (VisionPro is not true 3D point cloud) Some would consider the Cognex technique 2.5D, not 3D. HALCON can also perform the same 2.5D matching technique that VisionPro employs if that solution provides "good enough" results for the application

Table A.4: Vision Processing Performance: HALCON vs. VisionPro

	HALCON	VisionPro
3D Inspection (Metrology)	HALCON provides the ability to extract objects from 3D point clouds and measure their 3D size and shape. HALCON also can register and combine point cloud data from multiple sensors. HALCON can detect 3D defects through surface comparison with a CAD model or "golden" image. HALCON can create an unlimted number of base planes for making measurements or 3D matching	VisionPro 3D inspection is limited to translating 3D data into arrays of data. It can only perform measurements such as height, width and volume from a single base plane. I.E. - Cognex VisionPro cannot perform 3D shape based surface inspection within 3D Point Clouds using 3D CAD models or a golden template
3D Image acquisition	HALCON can acquire 3D data from stereo images, laser line profilers, time of flight sensors, interferometers, Microsoft Kinect and fringe projection systems using industry standard interfaces such as GigE, USB3, GeniCam, GenTL, DirectShow	VisionPro can acquire 3D data from only laser line profilers and stereo camera pairs if the camera manufacturer or Cognex has created a CCF (Camera Configuration File)
Programming and hardware environment flexibility	HALCON supports Windows, Mac OS X and Linux operating systems and more programming languages and environments than Cognex	VisionPro only supports Windows and has stopped supporting their customers who want to use ActiveX and COM
GPU processing capability	HALCON supports GPU processing	VisionPro does not supports GPU processing
Technical Support	MVTec has over 45 software and applications engineers working on and supporting the HALCON product. In North America the factory trained distributors such as us (JMAK Automation) are the first line of support, who then rely on a dedicated team of support engineers located in the US and Germany. Supporting a software product via the internet today via remote log in makes support location trivial. HALCON is a very large software package and can perform very complicated tasks. Therefore in complex vision tasks HALCON can sometimes be overwhelming and an excellent support network is necessary	VisonPro support - Cognex is a large corporation with many different product lines. The customers we work with tell us that VisionPro technical support has waned over time. More attention is put into the ID and InSight products so even though Cognex has more application and support engineers located in the US, only a select few actually know and can support VisionPro in depth
Up Front License Costs	HALCON - $6875 to get started with a perpetual development license of that version and any future releases within 12 months. Advantage - HALCON development license can also be deployed as a runtime license	Advantage to Cognex - Provides a timed 12 month USB Development System dongle for $995/yr that must be renewed annually at $995. Disadvantage - Cognex's VisionPro Development license cannot be deployed as a run time license

A.3.1.4 Image Acquisition

Obviously, the acquisition of images is a task that must be solved in all machine vision applications. Unfortunately, this task mainly consists of interacting with special, non-standardized hardware in the form of the image acquisition device, e.g., a frame grabber board or an IEEE 1394 camera. To let you concentrate on the actual machine vision problem, HALCON provides

you with interfaces performing this interaction for a large number of image acquisition devices (see *http : //www.mvtec.com/halcon/framegrabber* for the latest information). Within your HALCON application, the task of image acquisition is thus reduced to a few lines of code, i.e., a few operator calls. What's more, this simplicity is not achieved at the cost of limiting the available functionality: Using HALCON, you can acquire images from various configurations of acquisition devices and cameras in different timing modes.

Besides acquiring images from cameras, HALCON also allows you to input images that were stored in files (supported formats: BMP, TIFF, GIF, JPEG, PNG, PNM, PCX, XWD). Of course, you can also store acquired images in files.

A.3.1.4.1 Basic Concept

A.3.1.4.1.1 Open Image Acquisition Device

If you want to acquire images from a frame grabber board or an image acquisition device like an IEEE 1394 camera, the first step is to connect to this device. HALCON relieves you of all device-specific details; all you need to do is to call the operator **open_framegrabber**, specifying the name of the corresponding image acquisition interface.

There is also a "virtual" image acquisition interface called File. As its name suggests, this "frame grabber" reads images from files, and also from so-called image sequence files. The latter are HALCONspecific files, typically with the extension .seq; they contain a list of image file names, separated by new lines (you can create it easily using a text editor). If you connect to such a sequence, subsequent calls to **grab_image** return the images in the sequence specified in the file. Alternatively, you can also read all images from a specific directory. Then, you do not have to create a sequence file, but simply specify the directory name instead of the sequence file as value for the parameter 'CameraType'.

Now, subsequent calls to **grab_image** return the images found in the specified image directory. Both approaches are useful if you want to test your application with a sequence of image files and later switch to a real image acquisition device.

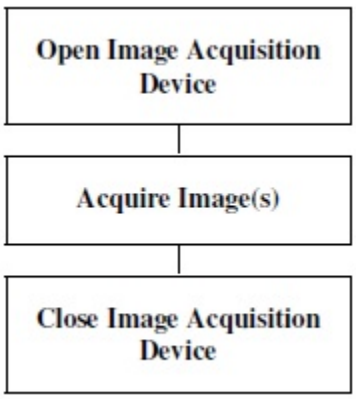


Figure A.13: Open Image Acquisition

A.3.1.4.1.2 Acquire Image(s)

Having connected to the device, you acquire images by simply calling **grab_image**.

To load an image from disk, you use **read_image**. Images are searched for in the current directory and in the directories specified in the environment variable HALCONIMAGES.

A.3.1.4.1.3 Close Image Acquisition Device

At the end of the application, you close the connection to the image acquisition device to free its resources with the operator **close_framegrabber**.

A.3.1.4.2 Extended Concept

In real applications, it is typically not enough to tell the camera to acquire an image; instead, it may be important that images are acquired at the correct moment or rate, and that the camera and the image acquisition interface are configured suitably. Therefore, HALCON allows to further parameterize the acquisition process. In HDevelop, an assistant is provided via the menu item Assistants > Image Acquisition that assists you when selecting your image source, adjusting the parameters, and generating suitable code.

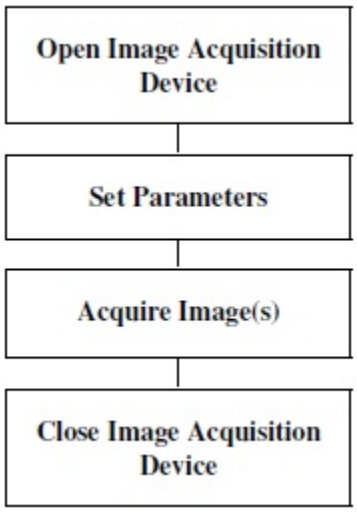


Figure A.14: Extended Concept

A.3.1.4.2.1 Open Image Acquisition Device

When connecting to your image acquisition device with **open_framegrabber**, the main parameter is the name of the corresponding HALCON image acquisition interface. As a result, you obtain a socalled handle, with which you can access the device later, e.g., to acquire images with **grab_image** or **grab_image_async**.

With other parameters of **open_framegrabber** you can describe the configuration of image acquisition device(s) and camera(s), which is necessary when using more complex configurations, e.g.,

multiple cameras connected to different ports on different frame grabber boards. Further parameters allow you to specify the desired image format (size, resolution, pixel type, color space). For most of these parameters there are default values that are used if you specify the values 'default' (string parameters) or -1 (numeric parameters).

With the operator **info_framegrabber** you can query information like the version number of the interface or the available boards, port numbers, and camera types.

A.3.1.4.2.2 Set Parameters

As described above, you already set parameters when connecting to the image acquisition device. These parameters (configuration of image_acquisition device(s) camera(s) and image size etc.) are the so-called general parameters, because they are common to almost all image acquisition interfaces. However, image acquisition devices differ widely regarding the provided functionality, leading to many more special parameters. These parameters can be customized with the operator **set_framegrabber_param**.

With the operator **get_framegrabber_param** you can query the current values of the common and special parameters.

A.3.1.4.2.3 Acquire Image(s)

Actually, in a typical machine vision application you will not use the operator **grab_image** to acquire images, but **grab_image_async**. The difference between these two operators is the following: If you acquire and process images in a loop, **grab_image** always requests the acquisition of a new image and then blocks the program until the acquisition has finished. Then, the image is processed, and afterwards, the program waits for the next image. When using **grab_image_async**, in contrast, images are acquired and processed in parallel: While an image is processed, the next image is already being acquired. This, of course, leads to a significant speedup of the applications.

HALCON offers many more modes of acquiring images, e.g., triggering the acquisition by external signals or acquiring images simultaneously from multiple cameras.