TAMPERE UNIVERSITY OF TECHNOLOGY

DAVID LEONARDO ACEVEDO CRUZ

CALIBRATION OF A MULTI-KINECT SYSTEM

Master of Science Thesis

Examiner: Irek Defée
Examiner and topic approved by the
Faculty Council of Computing and
Electrical Engineering on 09.05.2012.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Program in Information Technology
DAVID LEONARDO ACEVEDO CRUZ: Calibration of a Multi-Kinect System
Master of Science Thesis, 60 pages
November 2012
Major: Multimedia
Examiners: Professor Irek Defée
Keywords: Kinect, 3D calibration, multi-sensor, RGB-D sensors, 3D sensing, sensor evaluation.

The recent introduction of low cost RGB-D depth sensors has made available 3D sensing capabilities to many researchers and applications developers. Although the 3D sensing performance of these sensors is able to satisfy the needs of most common applications, there are applications, which require higher accuracy and coverage area. This thesis presents an evaluation of a low cost RGB-D sensor and proposes a multi-sensor system in order to overcome the limitations of a single sensor. The Kinect sensor was chosen as the target of the evaluation process due to its popularity, tools availability and support. The thesis started with an analysis of the 3D sensing mechanism and operation of the sensor, identified its limitations and proceeded with a performance evaluation. The Kinect sensor was evaluated taking into account the behavior of accuracy, precision and resolution and its variation along the distance to the target scene. As a step towards the solution of the limitations of the sensor, a multi-sensor system is proposed and analyzed. The main idea is to combine the data from several sensors located at different positions in order to improve the accuracy of the information and increase the coverage of the scene. The method proposed to combine the information of the multi-sensor system is based on the Kinect performance evaluation results. An experimental testbed with two Kinect sensors has been assembled and tested. It is shown that performance strongly depends on proper calibration of the system. Method for calibration has been developed and experimentally evaluated. Results show good performance of the system. The main outcome of the thesis is the evaluation of the measurements of the Kinect sensor, an automatic calibration system for several sensors and a method to fuse the input from several sensors taking into account limitation on data precision.

PREFACE

This thesis represents a culmination of work and learning cycle during my staying in Finland. The topic that I chose for my thesis answers to my passion for the 3D processing field, not only regarding the theory but also experimenting as I did during the development of the study. In addition, the desire of making a useful contribution to the current and booming 3D technologies, leaded me to focus in low cost RGB-D devices, which I think will become part of our daily life very soon.

I would like to thank to the Professor Irek Defée for his guidance, support and valuable inputs, which helped me to solve all the difficulties I found on the way. In addition, I want to thanks the Signal Processing Department for the sponsorship, and my friends who supported me unconditionally whenever I needed it.

Finally, I want to dedicate this achievement to my parents, Alvaro Acevedo and Pilar Cruz, my dear sisters, Catalina and Karen, and my family in Colombia and in Finland. I would not have made it without their support, advice and encouraging words.

David Leonardo Acevedo Cruz
Tampere FI, November 2012

CONTENTS

ABSTRACT I
PREFACE II
CONTENTS III
List of Figures IV
List of Tables V
List of Abbreviations VI
1. Introduction 1
2. Microsoft Kinect Sensor 3
    2.1. Sensor Technical specification 3
    2.2. Depth sensing mechanism 5
    2.3. Tools and libraries 12
    2.4. Kinect sensor limitations 14
3. Kinect Sensor Evaluation 16
    3.1. Evaluation Methods 17
        3.1.1. Accuracy evaluation 20
        3.1.2. Precision measurement 24
        3.1.3. Resolution measurement 24
    3.2. Results 25
        3.2.1. Accuracy results 25
        3.2.2. Precision results 30
        3.2.3. Resolution results 31
4. Multi-Kinect Sensor System 33
    4.1. Multi-sensor calibration 35
        4.1.1. Empirical multi-sensor calibration 35
        4.1.2. Automatic multi-sensor calibration by using a common plane 36
        4.1.3. Multi-sensor calibration results 44
    4.2. Data fusion 46
5. Conclusion 50
References 52
Appendix 1: Software Implementation Description 54
Appendix 2: Class Diagram of Software Implementation 55

LIST OF FIGURES

Figure 2.1 Kinect sensor (Microsoft) 4
Figure 2.2 Components of the Kinect sensor. (Primesense) 5
Figure 2.3 Components of the depth sensing mechanism [5] 6
Figure 2.4 Light coding pattern [9] 8
Figure 2.5 Sample uncorrelated pattern (Freedman et al.). 9
Figure 2.6 Effect of astigmatic optic element[11] 10
Figure 2.7 Drivers, middleware and libraries available for RGBD cameras 12
Figure 3.1 Representation of high accuracy and low precision. 16
Figure 3.2 Representation of high precision and low accuracy 16
Figure 3.3 Representation of high accuracy and high precision. 17
Figure 3.4 Measurement scenario for sensor evaluation 18
Figure 3.5 Main flow for measurements evaluation 19
Figure 3.6 Flow diagram of measurement process in x and y directions 21
Figure 3.7 Point cloud showing the plane object parallel to the sensor 21
Figure 3.8 Flow diagram of accuracy measurement by average in Z 22
Figure 3.9 Flow diagram of accuracy measurement by plane segmentation 23
Figure 3.10 Accuracy measurements in X dimension 26
Figure 3.11 Accuracy measurements Y dimension 27
Figure 3.12 Accuracy measurements in Z dimension using average 28
Figure 3.13 Accuracy measurements in Z dimension using plane segmentation 29
Figure 3.14 Accuracy comparison in X, Y and Z vs distance 29
Figure 3.15 Variance comparison in X, Y and Z vs distance 30
Figure 3.16 Measured and theoretical resolution vs distance 32
Figure 4.1 Scenario for empirical calibration 36
Figure 4.2 Scenario for automatic Multi-Kinect calibration 38
Figure 4.3 Rotation matrix calculation 39
Figure 4.4 Translation matrix calculation 42
Figure 4.5 Results of empirical calibration. 44
Figure 4.6 Scene used for auto calibration process 45
Figure 4.7 Multi-Kinect sensor auto calibration process. 46
Figure 4.8 Point cloud fuse flow 49

LIST OF TABLES

Table 3.1 Accuracy measurements in X dimension 26
Table 3.2 Accuracy measurements in Y dimension 27
Table 3.3 Accuracy measurements in Z dimension using average 27
Table 3.4 Accuracy measurements in Z dimension using plane segmentation 28
Table 3.5 Variance in X, Y and Z dimensions vs distance 30
Table 3.6 Resolution measurements 31

LIST OF ABBREVIATIONS

3D	Three dimensions
DOE	Diffraction optical element
IR	Infrared
PCL	Point Cloud Library
RGB-D	Red Green and Blue space color + Depth data
SDK	Software Development Kit

1. INTRODUCTION

Capturing 3D objects and scenes has been a challenge that dates from several decades ago. The devices built to accomplish this goal were usually characterized by their high cost and complexity. Devices such as range cameras, multi-camera systems, time of flight cameras and others were the most common way to capture 3D information but it was only possible to acquire them for the cost in the range of many thousand euros. The cost and complexity of these devices complexity made the 3D sensing a subject in which only some research centers and universities had a chance to study.

During the last couple of years this has completely changed due to the release of the Microsoft Kinect sensor provided means to capture 3D scenes in a relative simple way and for a very low cost. Other similar devices are also starting to appear. The Kinect sensor is one of the most successful new consumer products in history since it made its way to the home of over 19 million users in less than two years. The market penetration of this device contributes to the reasons why researchers and product developers are focusing their work on it.

Nowadays, it is common to find many so-called “Kinect hacks” and applications, which are nothing else than new ways to use and exploit the Kinect potential. Some examples can be found below:

In medicine and therapy: Kinepeutics is a physical therapy system for home use[1]. The system tracks body positions, logging the changes in muscular behavior and posture. The system was built in order to help users track their improvement over time through a feedback guidance system.

In shopping: Bodymetrics is a system for cloth fitting that scans and builds up a 3D model of a customer's body[2]. After an evaluation of the 3D model, the system is able to determine the size of clothes and allows a preview of the clothes on the 3D model.

In robotics: ROS or Robot Operating System has taken advantage of all the potential that the Kinect sensor offers. ROS is a project of the family of PCL and OpenCV. It exploits the 3D sensing capabilities of the Kinect that along with its own algorithms to allow functionalities such smart navigation, object identification and many more[3].

In gaming and 3D modeling: Many plugins have been created to use the potential of the Kinect with many 3D modeling software and gaming engines such as: Unity3D, 3DMax, Blender and so on. This plugins allow modeling 3D scenes in matter of minutes, reducing the amount of work and increasing considerably the quality of the work.

In language: There are many projects trying to reach a fully and accurate identification of sign language, there have been some breakthroughs but not enough to offer a

good experience to the user. The main problem of these projects lays on the accuracy of the Kinect sensor, which is not enough to obtain a proper identification of the movement of the fingers.

Although the 3D sensing capabilities offered by the Kinect sensor are very good compared to its price, and they suit most of the needs of basic applications, such as skeleton, gestures recognition, this is not enough when the purpose is to identify small details, slight movements or objects that are located out of the working range of the sensor.

These limitations lead us to a problem that is treated in this thesis in which the goal is to figure out in a concrete way what are the limitations of the Kinect sensor regarding its accuracy, precision and resolution and how these factors are affected by distance and changes in the target scene.

In addition to this problem, this thesis will study a multi-sensor system as a solution to solve the limitations of a single sensor. The challenge is to identify a proper method to fuse and enhance the data of two or more sensors pointing to the same scene.

The topic of Multi-Kinect systems is new. As this thesis has been in finishing stage, some solutions based on this concept were publicized. In October 2012, a company called 3Gear presented a Multi-Kinect system that enhances the depth data making possible to capture hand and finger movements in real time with high quality. Also in October 2012, Microsoft released a new update of its SDK, which allows fusion of data from several Kinect sensors improving depth data and coverage. These developments show the relevance and time pertinence of the topic of this thesis.

The thesis is organized as follows:

The second chapter of the thesis describes the technical characteristics, 3D sensing mechanism and limitations of the Kinect sensor

The third chapter addresses the evaluation of a single sensor taking into account factors such as accuracy, precision and resolution and its variation along the distance with the sensor

The fourth chapter explains how a Multi-Kinect system can solve the limitations of a single Kinect and describes the requirements and methods in order to achieve a proper data fusion.

The fifth chapter presents the contributions achieved by the thesis and discusses potential further work.

2. MICROSOFT KINECT SENSOR

The Microsoft Kinect Sensor is a 3D scene-capturing device developed by PrimeSense company in collaboration with Microsoft, introduced in 2010. The Kinect sensor was initially sold as a gaming accessory for Microsoft Xbox game console and its main purpose was offering a new and revolutionary way of interacting with games. Kinect is in fact a multisensor system with video camera, 3D depth camera and array of microphones, which turn it into sophisticate natural user interface, which allows users to interact without the need of making physical contact but just by using gestures and voice commands.

The Kinect sensor not only became a milestone for the new gaming interfaces but also got the attention of researchers and product developers in many areas other than gaming. The 3D sensing capabilities shown by the sensor make it a very good and cheap option to replace expensive and complex 3D sensing systems such as time-of-flight cameras, laser scanners and multi-camera systems.

Nowadays, the Kinect sensor is considered not only a consumer device but also, it is of interest in many professional areas of applications including medicine, entertainment, marketing, robotics, and many others. Each of these areas has its own different requirements regarding precision, accuracy, resolution, range and so on.

In this area, the Kinect was chosen as topic of due to its wide market penetration, low price, availability of SDKs and libraries, wide support, documentation and reception in the research community. It is important to notice that although the focus is on Microsoft Kinect Sensor for Xbox, the methods investigated in the thesis also apply to other similar sensors starting to appear like Asus Xtion or the PrimeSense sensor.

2.1. Sensor technical specification

The Kinect sensor is composed of four main elements (See figure 2.1): an array of four microphones (3), an RGB camera (2), a depth sensing mechanism (1) and a tilt motor (4).

The array of built-in microphones is made by four microphones located equally spaced in the bottom part of the sensor with a 24-bit analog to digital converter (ADC). The sensor is able to process the audio captured by the Kinect and execute operations such as noise suppression, acoustic echo cancelation and acoustic source location by comparing and triangulating a position out of the four captured sounds. The micro-

phones capture the audio as an uncompressed audio format using pulse code modulation (PCM) in which each channel process 16 bits in mono at a frequency of 16 kHz.

The RGB camera operates at 30 frames per second capturing images of 640x480 pixels with 8 bits per channel and producing a Bayer filter output with a RGGBD pattern. The camera itself is able to perform automatic white balancing, flicker avoidance and color saturation operations. Although the camera works by default with a resolution of 640x480, this can be changed to a higher resolution of 1280x1024 obtaining images at 10 frames per second.

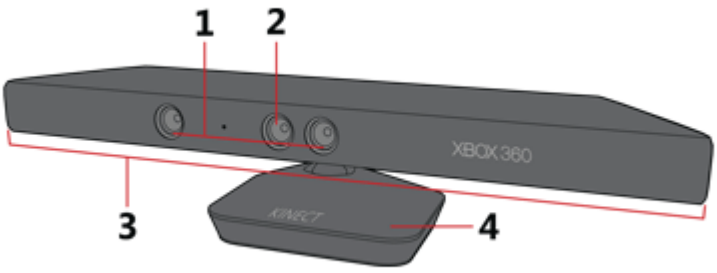

Figure 2.1 Kinect sensor (Microsoft)

The depth sensing mechanism comprises two elements: an infrared projector and an infrared capture device, these two elements are separated from of each other by 7.5 cm. This information is important for image triangulation purposes and will be explained in detail later in this chapter. The depth mechanism is able to capture depth images at a resolution of 1200x900 pixels with 8 bits[4], these images are downsampled to a resolution of 640x480 with 11 bit pixels allowing to store 2048 (2^{11}) levels of depth represented by intensity and decreasing the throughput necessary to transmit the information. According to the specification provided by Microsoft, the effective depth range in which the sensor works is between 0.7 and 3.2 meters.

Both, the RGB camera and the depth sensing mechanism have a common coverage area, 43° vertical field of view and 57° horizontal field of view. As mentioned before, the Kinect sensor is also provided with a tilt motor that is used to move vertically the head of the sensor in order to find the best angle to capture the scene. The tilt motor has a free angle of movement of 27°, according to the documentation published by Microsoft, the tilt capabilities are meant to use in few cases, such as initial calibration of the sensor. The motor is not intended for constant use and there is warning about the possible failure of the tilt motor after excessive use of it.

The transfer of information between the sensor and the receiving device is accomplished using a USB 2.0 connection. The effective throughput of this connection is 35Mb/s, although this capacity is usually enough for most of devices, in the case of the Kinect sensor it occupies most of this capacity making impossible to connect more than one Kinect to a single master USB connection. This is also one of the reasons why the

depth sensing mechanism downsamples the initial 1200x900 pixels image to 640x480 pixels with 11 bits.

2.2. Depth sensing mechanism

In this section, a description of the depth sensing mechanism used by the Kinect is presented. Although the Kinect sensor contains a RGB camera, array of four microphones and a tilt motor, these devices are not used to calculate the depth data. The depth calculation process only takes advantage of the infrared laser projector and infrared detector. In fact, other sensors of this kind such as the ASUS Xtion and the in-house Primesense sensor do not include RGB camera or microphone but do use the same sensing algorithm as the Kinect.

The figure 2.2 shows a diagram of the components of the sensor and it marks the components that are mandatory in order to capture 3D data. As mentioned before, the only mandatory components are the IR projector and detector.

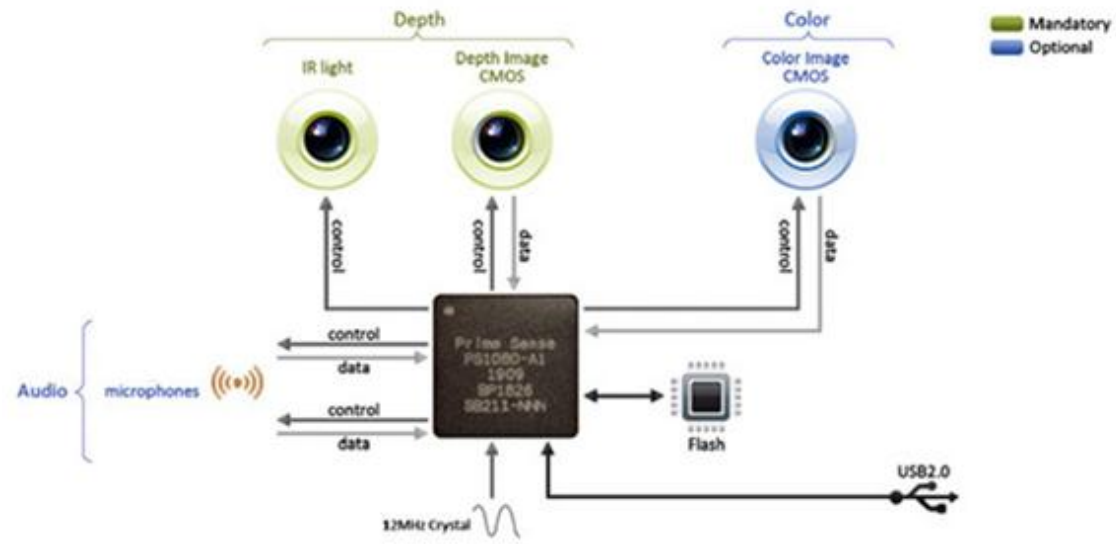

Figure 2.2 Components of the Kinect sensor. (Primesense)

It is important to highlight the fact that the Kinect does not use the RGB camera to calculate the depth, this makes it more robust in the sense that it is not affected by changes in light conditions, it can even produce depth map in darkness.

Explained in a simple fashion, the depth detection process works in the following way: The infrared projector projects a speckle pattern onto the scene, the infrared detector filters it and reads only the projected pattern, a processing unit analyses the pattern and compares to a reference hardcoded pattern. The depth values for each spot belonging to the pattern are calculated by triangulation. In order to increase accuracy, a distortion model is used to identify the shape changes on the spots and correlate them to the direction in which the depth varies. As a result, a 3D map composed by voxels is creat-

ed. A voxel is a volumetric pixel that is composed by three coordinates, one for each dimension. Thus, the value stored in the "z" component of each voxel corresponds to depth.

As hinted before, the sensing mechanism is composed by three main elements: an illumination apparatus, an infrared capture device and a processing unit. These elements can be seen in the figure 2.3.

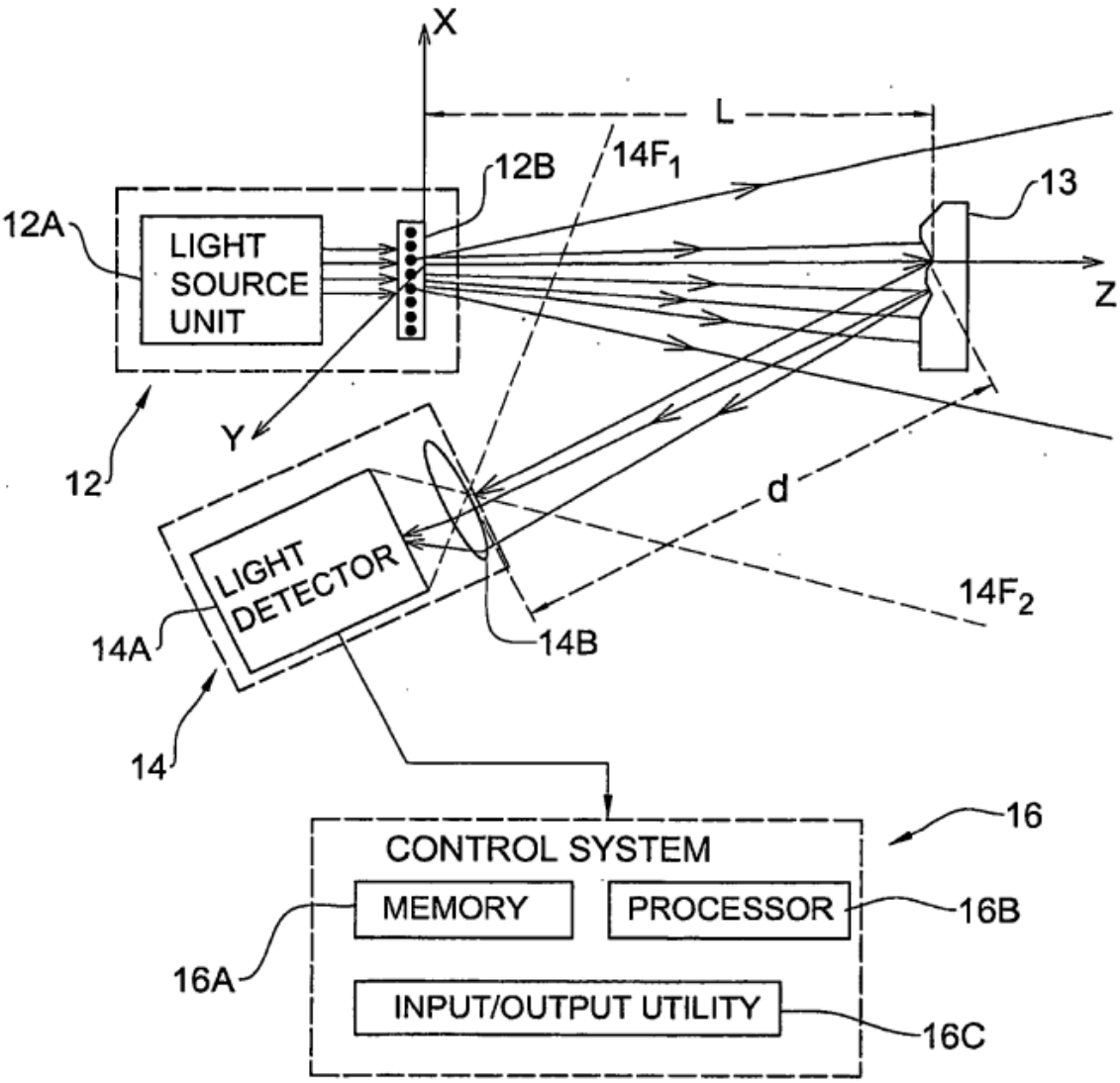

Figure 2.3 Components of the depth sensing mechanism [5]

The illumination apparatus is in charge of projecting a pattern in the scene. The pattern is characterized for being a pseudo-random speckle of uncorrelated bright and dark spots. This is important part of the invention behind Kinect, the uncorrelated pattern refers to a distribution of spots in the pattern, "whose positions are uncorrelated in planes traverse to the projection beam axis"[6].

The pattern projected by the illuminator uses the “Light Coding” technology which is a method patented by Primesense. The details of the Light Coding technology have not been publicly disclosed, for this reason, most of the inferences made in this description are based on reverse engineering experiments and the patens that support this technology.

Light coding falls into the structured light techniques, to be more precise, Light Coding can be classified among invisible structured light technique [7], which use spatial neighborhood or multiplexing pattern coding [8]. Structured light techniques also project a pattern and capture it, process it and depending on the shifts on the pattern calculate the depth values using triangulation.

In structured light, the most simple case scenario uses a pattern with a single spot, whose initial position is previously known and it is easily comparable to the one captured by the sensor. The disparity between the positions of the spots will produce the depth value. In more complex and real scenarios, it is necessary to use a large amount of spots to map the whole scene, making very complicated to identify the correlation between the spots in the reference image and the ones in the captured one. This situation makes the structured light techniques computationally demanding, requiring high amounts of processing power to identify with fair accuracy the corresponding pair of each spot.

In order to make easier the matching of pair spots, these can be arranged at fixed and predetermined distances, turning the task into a region matching instead of a spot matching, in which each region has specific characteristics making easier to identify them. The more coded regions are created, the more accuracy is possible to obtain.

Here is where the Light Coding comes into play, Primesense was able to create a coded speckle pattern that allows a fast correlation between reference and captured image preserving depth accuracy, and thus permits calculation of the depth values in real time. The light coding pattern is characterized by nine brighter dots, which are caused by imperfect filtering of light when projecting the pattern. Similar filters to these have been created but those are only able to create a single and very bright dot. The big advance of the Light Coding is the ability of creating several dots bright enough to be easily identified and processed by the capture device and the processing unit [9]. An image of a captured Light Coding pattern highlighting the nine brighter dots can be seen in figure 2.4.

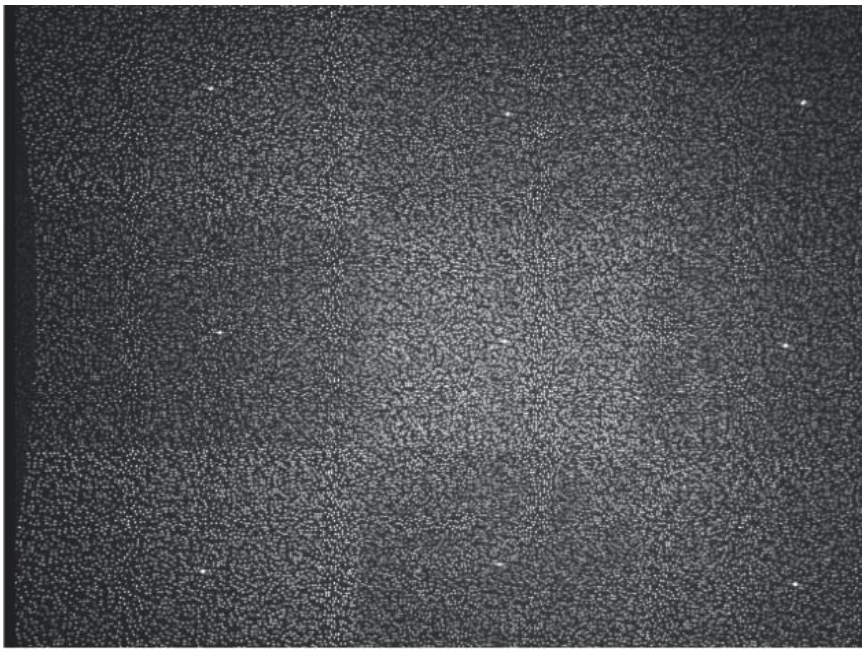

Figure 2.4 Light coding pattern [9]

The projected pattern is composed of random speckles, which are generated by interference of scattered light. As mentioned before, the pattern is composed by black and bright spots. Bright spots are positioned when the source of light is scattered in phase, and dark when it is scattered in anti-phase[5].

The pattern is characterized by a horizontal uncorrelation across each row [8]. In addition, the projected speckle pattern is constant in the "Z" direction. For the purpose of explanation, "constant" refers to none or very small variation [6]. It is important to notice that, although the distribution of the pattern does not vary along the Z direction, the sizes of the spots do, being scaled up linearly along with the distance from the source of light.

The size of each spot is also an important parameter to take into account. The size depends on the desired resolution and it is limited by the kind of light that it is emitted. The authors of the PrimeSense invention claim that spots of size between 2 and 3 pixels provided good results for the resolution required by the Kinect, in addition, due to the small size of the spot, it is not necessary to have a powerful light source [5][10] (See figure 2.5).

By now, it is easy to see that the pattern is one of the most essential pieces to obtain proper results. In addition to this, a special characteristic is added to the pattern when it is projected. Using an optical element with different focal lengths, it is possible to project the pattern onto the objects in such a way that the spots belonging to the pattern change its shape, elongating in the direction of the depth variation, and at the same time keeping its position in the pattern map[11]. It is important to notice the importance of the separation of the spots in the pattern, taking into account the possible elongation of the spot, this is necessary to avoid a situation of spots overlapping. For this reason, the separation of spots has to be bigger than 2 times the maximum elongation of a spot.

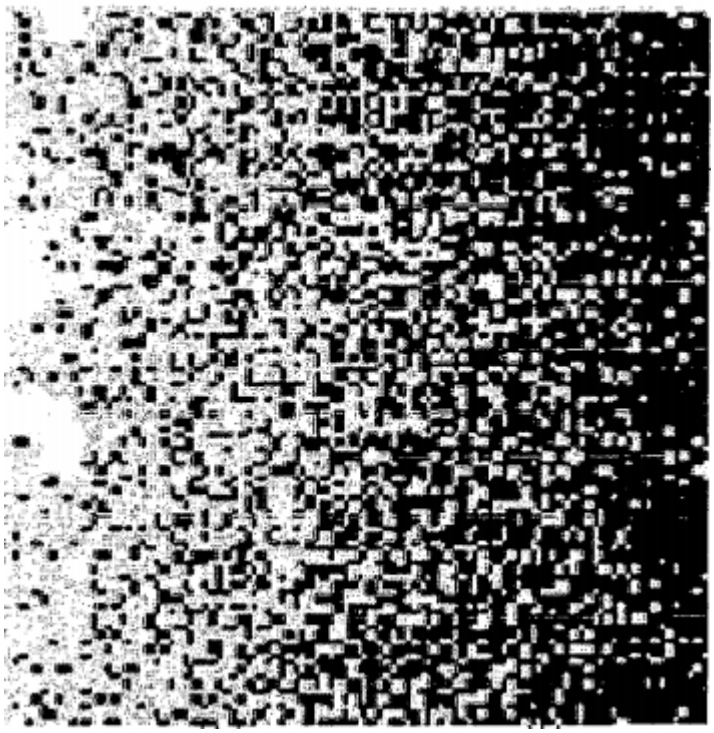

Figure 2.5 Sample uncorrelated pattern (Freedman et al.).

According to the inventors, this technology can be used with several types of light. The main requirement is that the light has to be coherent, which means that the emitted waves have to be in phase. This means that many kinds of light can be used, but in order to achieve successfully low cost, range, invisibility and accuracy requirements, a near infrared light was chosen as the best option since it is cheap to produce, invisible for human beings and easy to capture and filter.

In order to achieve the projection of the described pattern, the illumination apparatus has a coherent (laser) near-infrared light source, a transparency to generate the pseudo speckle pattern and an astigmatic optical element, which changes the projection of the pattern due to its different focal lengths.

The infrared light source has to be coherent and preserve a constant wavelength, these conditions can vary easily with changes of temperature in the laser diode. For this reason, the Kinect sensor has a built-in heater/cooler that ensures a constant temperature of the diode, hence constant wavelength and phase coherency are assured.

The transparency contained in the illuminator might contain a diffuser which is used to generate the speckle pattern; on the other hand, the transparency can have a hardcoded pattern following a pseudorandom distribution which is projected onto the scene [11]. Since the full details of this technology have not been disclosed, two possible methods of the sensor operation can be considered.

To achieve this, an astigmatic optical element is included in the illuminator The presence of this astigmatic optical element causes a distortion in the speckle of the pattern when it is projected onto the object. The distortion observed in the speckle does not refer to size scaling changes but in a precise manner to elongation of the spots in the

direction of the depth. The spots projected in the scene can be seen as ellipses, having minor and major axis where the major axis of the ellipse indicate the direction of the elongation. The astigmatic element has a DOE (diffraction optical element), which strongly benefits the projection of the pattern by reducing the variation of contrast of the speckle, equalizing the intensity of light along the plane traverse to the meridional plane[10]. The enhancements provided by the DOE are important in order to solve problems such as identifications of speckles at far distances which are produced due to drops on the contrast of the speckle [10].

In the figure 2.6 from the original patent one can see a source of light (34), transparency (36), astigmatic optic element (38), a capture sensor and two objects located at different distances in Z direction. In this scenario, the pattern observed by the capture device is different for each one of the target objects. As can be seen, the shape of the speckles in the pattern changes in elongation. In the case of the object at a Z3 distance, the spots look like ellipses and in the case of Z2, the elongation is such that they look like lines. The eccentricity of the ellipse and the length of its axis vary along with the direction of the elongation, hence determining the distance from the object to the capture device.

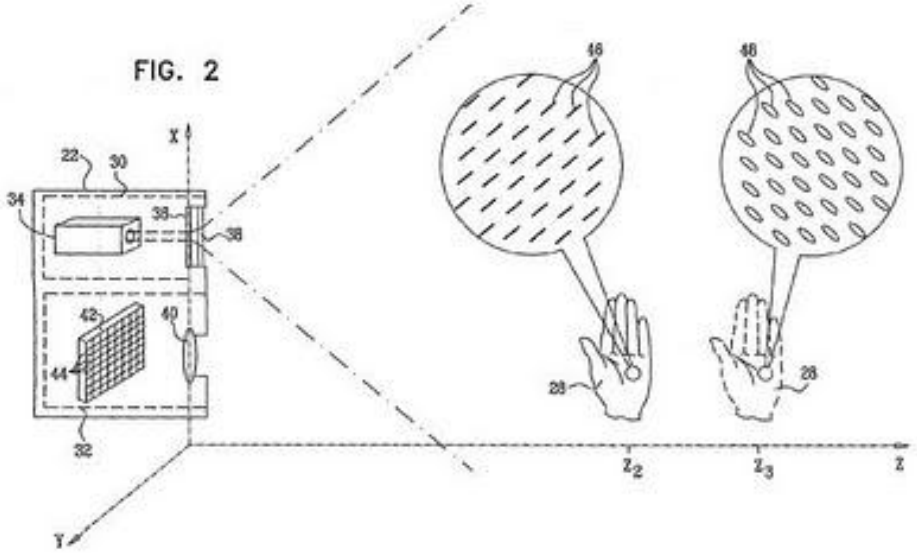

Figure 2.6 Effect of astigmatic optical element[11]

Another component of the sensing mechanism is the capture device or light detector (figure 2.6. 32), which is in charge of capturing the projected pattern. The capture device comprises a rectangular array of detector elements using CMOS-based image sensor array. In addition, it contains built in signal processing algorithms such as filters and contrast enhancers in order to capture only the emitted pattern and filter out ambient light that might reduce the contrast of the image [6], [11].

Due to practical issues, the illuminator and capture device have to be located at a fixed position from each other. This consideration allows to calculate the depth values by disparity for the whole scene using a single capture assembly [6], [10], [11].

In order to make computations simpler, illuminator and capture assemblies should be located so that they are aligned on the X-axis and in the same XY plane, thus both component will be approximately at the same distance from the objects in the scene. Consequently, the size characteristics and space of the spots in the emitted pattern and the captured one do not change significantly with distance[10], [11].

The processing unit is the last piece of the sensing mechanism. Its task is calculation of the depth values using the reference image and the captured one. Thanks to the properties of the emitted pattern, it is possible to determine the depth information in two different ways: calculating the disparity between reference and captured image and a second method based in a distortion model in which the elongation of the spots is measured and depending on this, it is possible to identify the depth value. It is important to make clear that these two methods are used complementarily, in other words the outcome of this process is a result of the agreement of results of these two methods.

The hardcoded pattern is the result of a calibration process in which the speckle pattern is projected on a known surface, presumably a plane parallel to XY. The result is a reference image, which is stored in the memory of the device [10]. The calibration is performed during manufacturing. Since the reference image does not change, any change in the illuminator or capture assemblies will clearly decrease the accuracy of the device.

The depth calculation process is carried in several consecutive stages. First, the correlation between the reference image and captured one is calculated, finding the pair for each spot in the pattern. As mentioned before, this process was improved due to the Light Coding technology, which allows an easy identification of the speckles in both images. After the correlation process is completed, the next step is to find depth values by disparity or triangulation in which only the changes in the x direction are taken into account.

In order to make the triangulation, it is necessary to take into account that the pattern projected by the source of light is in a different position than the image capture device since the illuminator and capture device are separated by 7.5 cm; this value is also hardcoded in the sensor. The Kinect measures the distance by parallax, which is a special case of triangulation where it is required to know the values of all the angles of the triangle and the length of at least one of the sides. In this case, the value of the angles is known along with the length of one of the sides, which is fixed and corresponds to the distance between the infrared camera and infrared projector.

Since the illuminator and capture device are aligned, it is easier to make the calculation of the depth values by disparity. A change in Z direction will correspond to the difference in the X direction between a spot in the reference image and its pair in the captured image. The shifts in Y direction are not taken into account to make the calculation, since both assemblies, illuminator and capture device are located at the same height.

At this point the first depth values have been found, the next steps try to enhance the accuracy of the values by taking advantage of the change of shape in the spots produced by the astigmatic element. Since the spots are elongated in the direction of the depth

variation and the length axis of the ellipses change according to how drastic is the change, it is possible to use this information to calculate more accurately the depth values.

2.3. Tools and libraries

Due to success of the Kinect sensor in the research and software development communities, many different tools, libraries and SDKs have been created to allow faster and more stable creation of applications using the sensor.

The software resources that have been created can be categorized in three different groups as shown in figure 2.7.

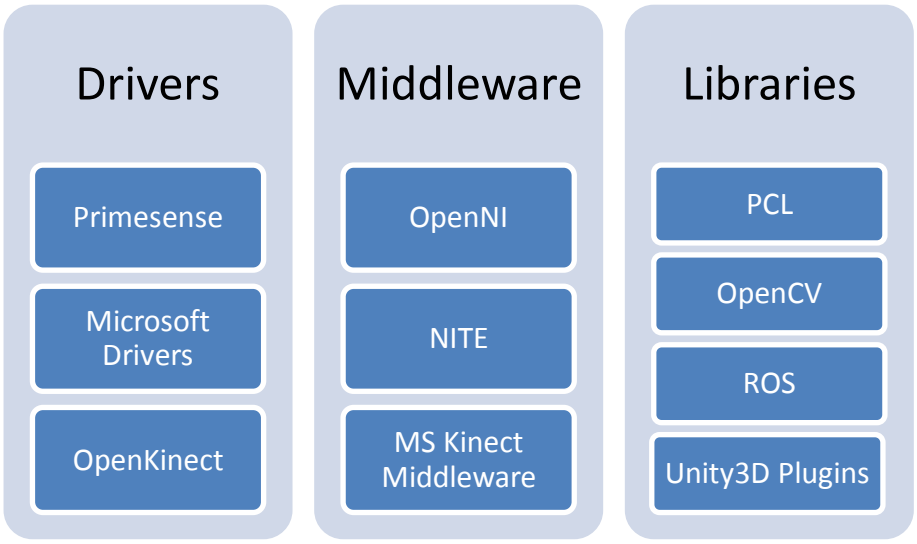

Figure 2.7 Drivers, middleware and libraries available for RGBD cameras

The device drivers section groups a collection of software created by the community (OpenKinect) or companies (Microsoft Drivers and PrimeSense) in order to allow the communication of the hardware device and software.

A device driver works as a translator between the hardware of the device and software. Most of drivers are platform dependent, meaning that several drivers have to be created in order to allow the hardware device to interact with different operating systems or CPU architectures. This situation leaded to the creation of not a single and main driver, but several ones in order to satisfy the needs of all users.

OpenKinect, before known under the name of libfreenect, was the first open device driver that was available for the development community. It was born out of the desire of exploiting the Kinect sensor and the lack of openness of the proprietary solutions. OpenKinect is open source and offers support for Windows, Linux and Mac platforms. Currently, OpenKinect can be seen only as a set of driver, but there are several plans for the future in which it should allow higher-level functionalities like hand tracking, skeleton tracking and many others that will position it also in the middleware group. It is important to point out that these drivers were created after experimenting and reverse

engineering the Kinect sensor, as a result, they are made specifically for this sensor and might not work as well as desired with other RGB-D sensors.

PrimeSense, the company creator of the technology used in the Kinect Sensor, released and opened the source code of its drivers in December 2010, just couple of months after the OpenKinect initiative. The Primesense driver works in Windows and Linux platforms and additionally supports all the RGB-D sensors that work with the PrimeSense patented technology such as: Kinect, Xtion, and its in-house Primesense sensor. Its completeness, flawlessness and multi-support have made this driver the most used by third party projects.

The device driver offered by Microsoft was released along with the beta version of the Kinect SDK in June 2011. It is a proprietary driver, which works only under Windows Vista or higher platforms and supports only the Kinect 360 and Kinect for Windows sensors. The driver is based on the one developed by PrimeSense with the addition of performance modifications for Windows environments. One of the main advantages of using this driver is that it is built-in Windows 7 and higher, therefore it is not necessary to install it.

The middleware classification is a software that is meant to expand functionalities and provide tools, which will be used by software applications. The middleware can be seen as an extension of basic services preparing them to be used in a more easy and consistent way for other software applications. The existent middleware for the Kinect sensor is characterized by functionalities such as body and hands tracking, skeleton construction, recognition of voice commands and hand gestures.

OpenNI is an open source framework, which is a result of the OpenNI organization for creating and becoming a standard for software interoperability with natural interaction devices. OpenNI is supported by several companies, which have been working in the natural user interfaces field such as PrimeSense, Asus and Willow Garage. OpenNI takes advantage of the PrimeSense driver and has it embedded, allowing the framework to work in several platforms and with several RGB--D devices.

NITE is an open source motion tracking middleware developed by Primesense able to identify hand gestures, perform body tracking, joints identification and player identification. This middleware is based on the standards provided by the OpenNI framework and in addition guarantees compatibility and portability with future versions of the PrimeSense sensor.

Microsoft Kinect SDK is the proprietary middleware offered by Microsoft to accomplish similar natural user interface functionalities to the ones offered by NITE. Since it uses the Microsoft Kinect drivers, it can only be deployed on Windows platforms using the Kinect sensor. The advantages of this SDK are the compatibility to create applications or games for Xbox 360 and the option of programming the software using C++ or C#.

The libraries group refers to more specific applications created on top of the middleware. Most of the libraries have been developed based on OpenNI and NITE due to

its portability between platforms and support of different devices. There are many existing libraries but only the most relevant are described below:

OpenCV or Open Source Computer Vision Library is collection of 2D image processing functionalities and methods. Although, this library is meant for 2D applications, there are many cases in which 3D processing can be done in 2D scenarios reducing the complexity of the problem. The communication between OpenCV and the Kinect sensor can be done using through the OpenNI framework or OpenKinect drivers.

PCL or Point Cloud Library is a standalone open-source framework based on OpenCV, which includes several 3D image processing algorithms covering areas such as filtering, features and key-points detection, segmentation, surface reconstruction, visualization and registration[12]. The library also includes an input module, which uses the OpenNI framework making possible to extract data directly from any RGB-D sensor using the Primesense technology. PCL is cross-platform making possible to use it in Windows, Linux, Mac, Android and iOS.

ROS is a project of the same family than PCL and OpenCV, it stands for Robot Operating System. It offers tools to create robot applications starting from the hardware abstraction, device drivers, visualizers, message passing and more. One of the most challenging areas in robotics is the capacity of sensing scene in order to allow the correct navigation of the robot inside it [3]. The Kinect sensor is a very good and affordable option to turn it into the eyes of any robotic project. In order to accomplish this, ROS included among its tools an interface to the RGB-D sensors using OpenNI and NITE. In the same way that OpenCV and PCL, ROS is also an open source and cross-platform.

Unity3D is one of the most used and powerful gaming engines, the release of RGB-D sensors create a new scenario in which players can interact with games, and in which scenarios and players can be created. In order to take advantage of these capabilities, several plugins and wrappers have been created in order to access the data from Kinect and connect it to the game environment. Unity3D takes advantage of the Microsoft Kinect driver, for this reason, these functionalities will be only available for products created for Windows platforms. In the same fashion than Unity3D other 3D modeling software such as MAYA, 3D Studio Max and blender have created its own wrappers in order to access data from Kinect and interoperate in the creation of 3D scenes and animations.

2.4. Kinect sensor limitations

As mentioned before, the Kinect sensor was designed as a new gaming accessory and in its functional requirements was expected to work in indoor environments and with a range that will give enough space to the player in a typical room. It is now known

that the sensor fulfills these requirements very well, but when taken to scenarios other than gaming, it displays several limitations that will be described below:

The Kinect uses a near infrared light to project the pattern that allows the depth calculation, the infrared light was chosen due to its price, invisibility and easy identification by infrared cameras. Although, the infrared light satisfy the needs initial requirements of the sensor, it produces many problems such as not working with sunlight, reflective surfaces, long distances and interference with signals with the similar wavelength.

The infrared light used by the Kinect has wavelength of 830 nm, which is detected by the capture device filtering the input and only allowing this specific wavelength. This measure is enough to filter out signals in other ranges such as from TV remotes, but it is not sufficient to avoid the interference that the high power of the sunlight infrared band has over the Kinect infrared signal [9]. Therefore, the Kinect sensor cannot be used outdoors or in any other environment affected by sunlight.

Another limitation is the coverage range of the sensor, which according to the Microsoft specifications are between 0.7m and 3.2m. The reason of this limitation is that the sensor uses a near infrared light, which cannot be captured easily at bigger distances.

Reflection is a common physical phenomenon that affects the direction of the wavefront, making non-possible for the capture device to obtain a correct measurement of the projected signal. The reflection problems are usually faced when glass or shiny objects are in the scene.

Due to the separation of dots in the pattern and a non-mill metric precision in the measurements, several artifacts appear when capturing the edges of objects. The result usually is a look like saw edge, which makes mandatory pre-processing of the edges before operating with them.

As mentioned before, the Kinect sensor was designed to accomplish specific requirements, unfortunately, projects that researchers and companies want to develop with it require a higher performance regarding accuracy and precision. Besides the sensing mechanism and infrared light emitted by the sensor, a reason that decreases the quality of the measurement is the lack of a big enough channel of transmission. Currently, Kinect transfers the obtained data trough USB 2.0, the data obtained by the sensor floods easily the bandwidth of a common USB 2.0 connection, for this reason, before the information is transferred it is downsampled, therefore the resolution of the measurements is reduced. Another RGB-D providers have improved this issue by using USB 3.0 or even transmitting the data using wireless connections.

3. KINECT SENSOR EVALUATION

This chapter will be dedicated to calculate and evaluate the accuracy, precision and resolution of the measurements obtained by the Kinect sensor. First, it is important to shed light on the definitions of accuracy, precision and resolution. Accuracy and precision are terms usually confused, common thinking is that they mean the same. Although both describe the quality of measurements, their proper understanding is vital to have a clear picture of the results that will be presented later.

Accuracy is defined as the closeness between the measured and the reference or real value, while precision refers to the closeness between independent measurements. Precision is independent of the real value, hence only measures how close the measured values are from each other [13]. Taking into account these definitions, it can be said that accuracy measures the proximity of the measurements to the correct value whereas precision measures the repeatability and reproducibility of the measurement. Therefore, correct value and the precision measurements are independent. This situation leads us to a scenario in which an accurate measurement does not imply precision and a precise measurement does not imply accuracy.

A graphical example of these scenarios can be seen in the next figures:

Figure 3.1 Representation of high accuracy and low precision.

In the figure 3.1, most of the values are close to the center of the bullseye target meaning a high accuracy, on the other hand, the measured values are separated, some at the left others at the right side, thus implying low precision.

Figure 3.2 Representation of high precision and low accuracy

Figure 3.2, shows how the values obtained are close from each other, suggesting a high precision in the measurements. The situation regarding the accuracy is the opposite, although the measurements are close to each other, they are far from the reference value which in this case is the center of the bullseye target, therefore this scenario lacks of accuracy.

Figure 3.3 Representation of high accuracy and high precision.

Figure 3.3 exposes a scenario in which the values are close to the center of the bullseye and at the same time are close to each other. Hence, the measurements are accurate and precise.

Another factor that will be taken into account in the performance evaluation of the sensor is its resolution and its variation with the distance. The resolution can be defined as the amount of information that a measurement holds. In this case, the resolution is represented by the number of voxels that fall onto a specific area at a specific distance.

3.1. Evaluation Methods

In order to evaluate the performance of the sensor, we created a scene that uses a Kinect sensor, a plane object and two tripods to hold the sensor and the object in a fix and predetermined position, the setup of this scene follows the same idea proposed by Khoshelham in [14].

As shown in figure 3.4, the Kinect sensor is located at a fixed distance, it is important to notice that the sensing mechanism has to be orthogonal to the base line, which in our case is the floor. The plane object is located following the same considerations taken with the sensor, it should be located at the same height and its plane area has to be parallel to the sensor and orthogonal to the base plane (floor).

With the purpose of reducing the human error that can be introduced when locating the objects, and possible irregularities in the floor, we use a spirit level to verify the correct position of the objects in the scene. In addition, any other object but the plane is removed from the scene so it is much easier to identify the voxels that fall on the plane and do not confuse them with ones in other objects.

In order to measure the performance of the Kinect at different distances, a set of seven different distances are considered, these distances are within the operation range of the sensor, starting from 0.7m to 3.7m with a variation of 0.5m for each measurement, thus the evaluation is performed at 0.7m, 1.2m, 1.7m, 2.2m, 2.7m, 3.2m and 3.7m.

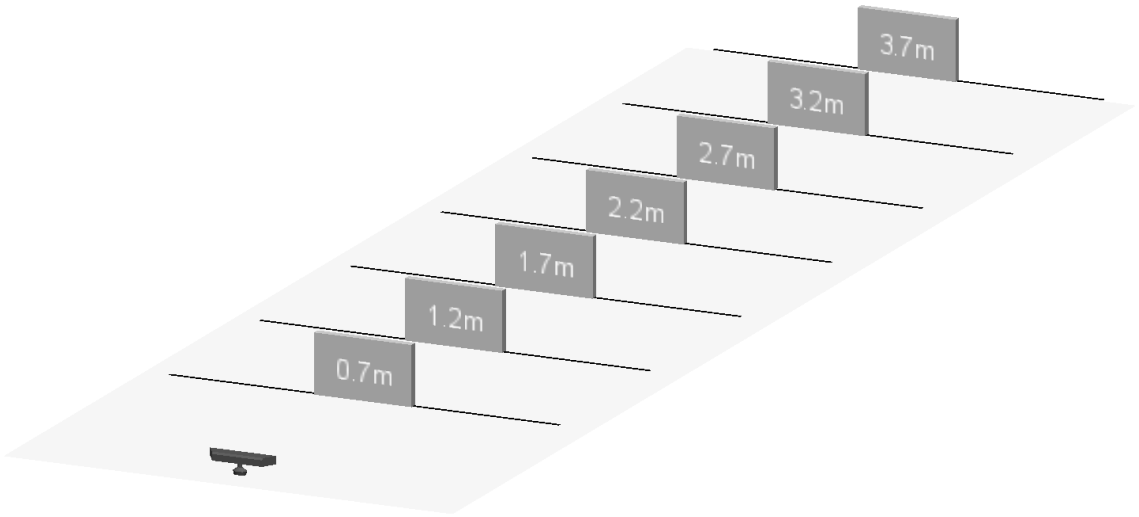

Figure 3.4 Measurement scenario for sensor evaluation

In the figure 3.4, it can be observed the seven different positions in which the plane object is located from 0.7m to 3.7m. It is important to realize that the plane object used in the measurement is always the same, meaning that the further it is located the smaller target it is for the infrared beams projected by the Kinect.

For practical effects, accuracy and precision evaluation is divided in two different cases, one for depth accuracy and other for accuracy in x and y directions.

In order to perform the measurements a software program was created using the Point Cloud Library, which was previously described. This library is able to capture the data stream from the sensor and convert it in a point cloud. A point cloud is a collection of voxels located in a three dimensional system. Voxels or volumetric pixels are elements that represent volume measurements, analogously to the pixel in 2D, the voxel is able to store data of the three dimensions, having X, Y and Z components.

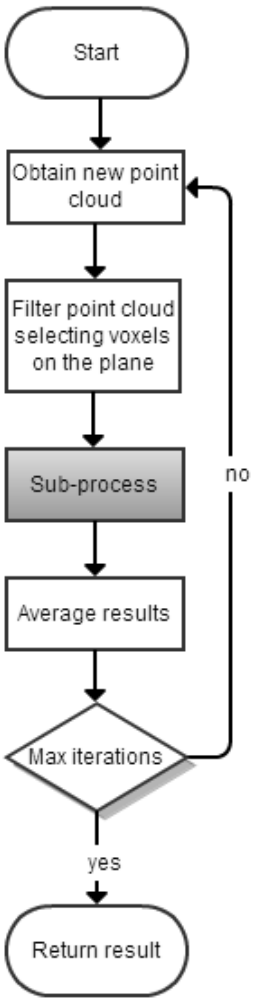

Figure 3.5 Main flow for measurements evaluation

The flow diagram in figure 3.5 shows in white color the common steps that all the evaluation measurements will have, in grey color and under the name of sub-process the position in the flow of the specific measurement process that will be executed are marked.

The first step in the process is to obtain a new point cloud from the sensor. The point cloud contains all the voxels in the scene. By default, the number of voxels contained by the initial cloud is equal to 307200 voxels that is the result of multiplying 640 * 480, which corresponds to the XY resolution of the depth sensor.

Since the only interest is to evaluate the voxels composing the plane object, it is necessary to filter the initial point cloud. Given that the scene is controlled and we know the real position and size of the plane object, it is not necessary to deal with obstructions or overlapping of other objects, thus, an initial prefiltering can be applied by dismissing the voxels out of the X and Y area in which the object is located. This filter reduces the size of the point cloud, thus improving the speed of the calculation by not having to process all the voxels in the scene but only the ones in which we are interested.

The next step is the “Sub-process” which refers to a specific measurement process depending on what wants to be evaluated, i.e., accuracy, precision or resolution. A description of each measurement process is presented in the next subsections.

The measurement process generates the result for a single iteration, the result is averaged with other results obtained. Subsequently, the flow of the process goes back to obtain a new point cloud. This loop is executed until the desired number of iterations is reached. The reason to execute several iterations is to decrease the possible measurement errors and converge to a more correct value.

3.1.1. Accuracy evaluation

For the effect of the implementation, the accuracy in X and Y dimensions is evaluated using a different process than in the Z dimension. The next subsections will explain the procedure for each of them.

3.1.1.1 Accuracy measurement in X and Y dimensions

Taking into account that the plane object is a rectangle and its plane face is located parallel to the sensor it is possible to process the information as a 2D scenario in which we can leave out of account the depth values in the voxels and just think in X and Y coordinates as if they were normal pixels.

Since the target object is a rectangle with known side lengths, it is possible to measure the distance from side to side of the rectangle in order to find experimentally the lengths and compare them to the real values. The diagram in figure 3.6 shows the flow of the process.

The starting input of the process is a point cloud composed of the voxels that fall onto the plane object. The first step of the process is to detect the edges of the plane object in the point cloud. Since it is known that the object has a rectangular shape, we assume that four edges have to be extracted. As mentioned before, the data obtained is treated as 2D image. Therefore, the Z coordinate of the voxels is ignored. As we can see in the figure 3.7, the plane object is orthogonal to the Z coordinate and parallel to the plane XY, which are represented by the colors blue, red and green respectively.

Figure 3.7 also shows that the edges are not straight, presenting peaks and irregularities. This makes necessary to smooth them thus obtaining a better and uniform representation. After the four edges have been extracted, it is possible to calculate the accuracy in X and Y directions. For X direction, the horizontal edges are selected and the average X value of the voxels is calculated for each edge. Subsequently, the difference between the two averages is computed. The resulting value corresponds to the experimental length of the horizontal side of the rectangle.

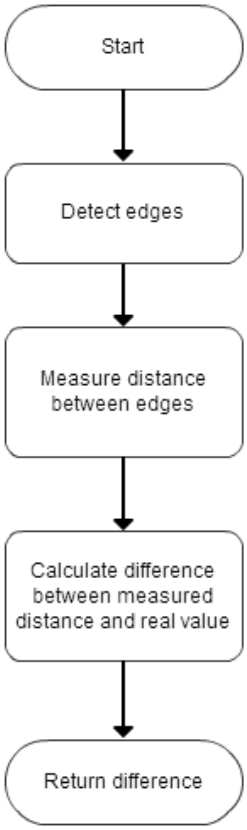

Figure 3.6 Flow diagram of measurement process in x and y directions

Analogously, for Y direction the same process is performed with the difference that, in this case, the operations are performed on the vertical edges and the average is taken from the Y component of the voxel.

To this point, both lengths of the sides of the rectangle have been calculated and since the real length of the rectangle sides are known, it is possible to compare and determine how accurate the measurement is.

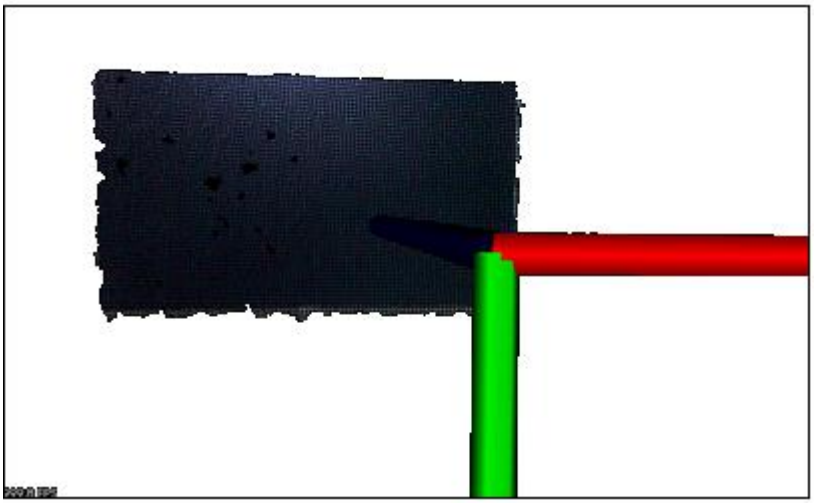

Figure 3.7 Point cloud showing the plane object parallel to the sensor

3.1.1.2 Accuracy measurement in Z coordinate

Two methods to measure the depth values are proposed. The first one uses an average of the depth values that fall onto the plane object and the second one takes advantage of plane segmentation and the plane equation.

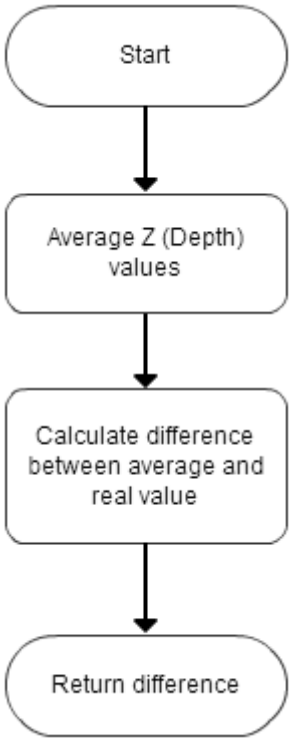

Figure 3.8 Flow diagram of accuracy measurement by average in Z

Figure 3.8 shows the flow of the accuracy measurement in Z using average of the Z component. The input of the process is a point cloud containing only voxels corresponding to the plane object. Since the plane object is parallel to the plane XY and orthogonal to the direction of the depth, we can tell that all the voxels contained in the plane also share this property. Therefore, the depth value for the voxels is equal to their "Z" component.

The first step averages the Z component of all the voxels corresponding to the plane object. The resultant average is compared to the real value, thus finding out the accuracy for a single experiment.

As mentioned before, the second method takes advantage of the plane segmentation algorithms and the plane equation. A diagram showing the flow of the algorithm is presented in figure 3.9.

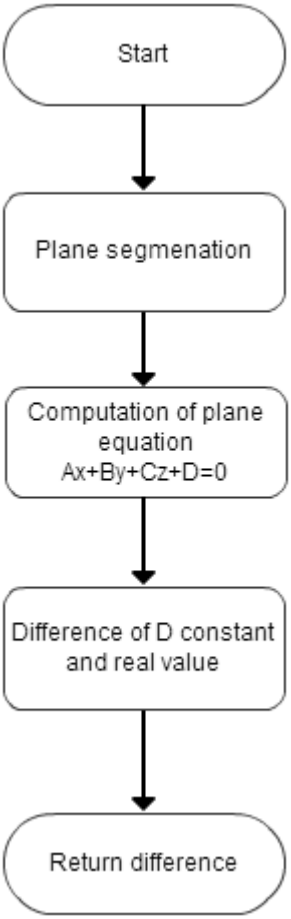

Figure 3.9 Flow diagram of accuracy measurement by plane segmentation

The input of the process is a point cloud, which only contains voxels representing the plane object. In the first step, a plane segmentation algorithm is executed on the cloud identifying a group of voxels that share the same or similar normal vector. The average of the normal of those vectors is taken and the normal value is stored to be used in the next step.

The second step uses the normal vector and a point in the plane in order to identify the equation of the plane, which can be seen in equation 1:

$$\mathrm{Ax} + \mathrm{By} + \mathrm{Cz} + \mathrm{D} = 0 \qquad (1)$$

The coefficients A, B and C are a vector representing the normal of the plane and D is a constant, which represents the distance from the origin to the plane. Since the plane is parallel to the XY plane, the value of the constant D is equal to the Z distance from the plane to the camera.

The fourth step calculates the difference between the constant D and the real depth value.

3.1.2. Precision measurement

As pointed before the precision is an indicator of stability and repeatability of the measurements. It can be evaluated by finding how close the values in a measurement are from to each other. To calculate the precision of the measurement we will use the variance statistic tool [15], which indicates the characteristic dispersion of the values in the measurement.

Since in every measurement we are obtaining the whole population of voxels that fall onto the plane and not just a sample of them, it is possible to use the population variance formula shown in equation 2:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 = \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \mu^2 \qquad (2)$$

In the equation above N represents the size of the population, μ is the population mean and X_i represents each one of the measurements.

In order to calculate the variance for X, Y and Z dimensions we can reuse the algorithms used to calculate the accuracy. Instead of calculating the difference between the real value and the experimental one, we will store the experimental value and continue the loop. When the maximum number of iterations N has been reached, the population mean μ can be calculated using all the measured values that have been stored.

At this point, we know the size of the population N, which is equal to the number of iterations, the population mean μ, which was previously calculated, and we have the array of measured values. With these variables, it is possible to calculate the variance.

3.1.3. Resolution measurement

The resolution indicates the amount of information in a measurement, therefore, in this case, the more voxels are present in the measurement the higher resolution and detail. The input of this process is the point cloud of voxels that compose the plane object. In order to determine the resolution, it is just necessary to count the number of voxels in the point cloud.

Besides finding the number of points in the cloud, we can compare this value to the theoretical number of voxels that should be in the cloud. This value can be calculated using the resolution of the sensor, the range angles in X and Y dimensions, the size of the plane object and the different distances in which the measurements are performed.

The first step is to find the area covered by the Kinect depending on the measurement distance. In order to perform this operation, the range angles and the distance of the measurement are used, where the angles are 57 deg and 43 deg for X and Y dimen-

sion respectively and the distances vary from 0.7 to 3.7 with a delta of 0.5m. The ranges are calculated using the equation 3:

$$Range = 2 * Distance * Sin(Angle/2) * Sin(90 - Angle/2) \quad (3)$$

Replacing in equation 3 we have:

$$Range\ in\ x = 2 * Distance * Sin(57/2) * Sin(90 - 57/2)$$
$$Range\ in\ y = 2 * Distance * Sin(43/2) * Sin(43 - 57/2)$$

Solving the scalar operations:

$$Range\ in\ x = 2 * Distance * Sin(28.5) * Sin(61.5)$$
$$Range\ in\ y = 2 * Distance * Sin(21.5) * Sin(68.5)$$

After calculating the range in X and Y dimensions, we can obtain the theoretical density of pixels per meter in each dimension. Since the resolution of the depth sensor is known to be 640 x 480 we just need to divide the X range in 640 and the Y range in 480.

$$Density\ in\ x = xRange/640$$
$$Density\ in\ y = yRange/480$$

After finding the density of voxels in X and Y dimensions and in every one of the distances considered for the measurement, it is possible to calculate the theoretical number of voxels that should fall onto the plane object. Since the length of the sides of the plane object are known and are equal to 0.29m and 0.205m in X and Y dimensions, the number of voxels on the plane should be:

$$\#voxels: 0.29/DensityX * 0.205/DensityY$$

3.2. Results

The experiments were carried out using a plane object with rectangular shape and a size of 29cm horizontally and 20.5cm vertically. The material of the object is cardboard in order to guarantee a low or almost null reflection of the infrared beams. The number of iterations used for these measurements is 1000.

3.2.1. Accuracy results

The measurements of accuracy in X dimension are presented in the Table 3.1. It is necessary to take into account the size of the plane object in X dimension, which is 0.29m, this value can be compared to the ones calculated experimentally and the difference between the real value and the calculates is shown in the difference field.

Distance	Calculated X	Difference
0.7	0.2886344	0.0013656
1.2	0.292728	0.002728
1.7	0.29319533	0.0031953
2.2	0.29714887	0.0071489
2.7	0.30391093	0.0139109
3.2	0.304417	0.014417
3.7	0.30438233	0.0143823

Table 3.1 Accuracy measurements in X dimension

Figure 3.10 shows how the accuracy decreases inversely proportional to the distance. Although this change seems a radical in the graphic, it is important to point out that at a distance of 3.7m the captured values vary from the real result approximately by 1.5 cm, which is an acceptable value at such a distance.

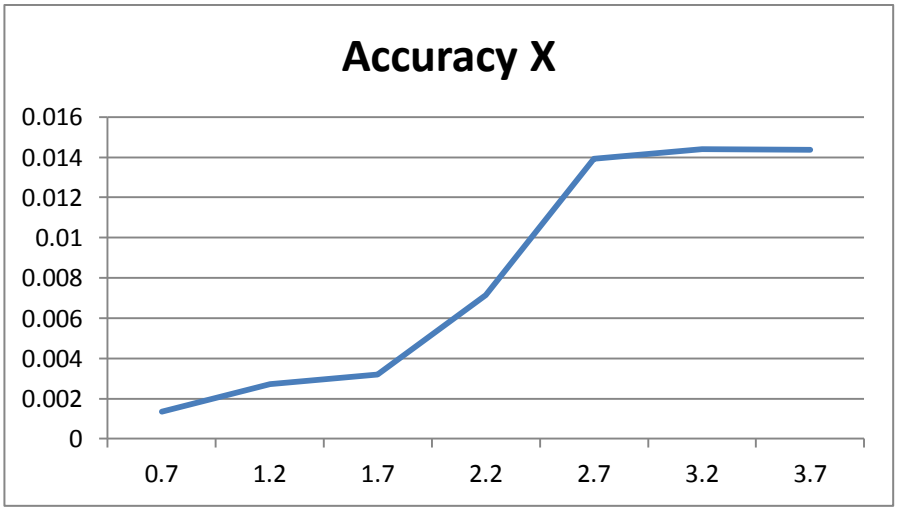

Figure 3.10 Accuracy measurements in X dimension

In the same fashion than for X dimension, Table 3.2 shows the measured values for the Y dimension. In this case the captures values are compared to the height of the object, which is 0.205m.

Distance	Calculated Y	Difference
0.7	0.2034344	0.0015656
1.2	0.20172	0.00328
1.7	0.20919533	0.0041953
2.2	0.19785113	0.0071489
2.7	0.19308907	0.0119109

3.2	0.219417	0.014417
3.7	0.18761767	0.0173823

Table 3.2 Accuracy measurements in Y dimension

Figure 3.11 illustrates the behavior of accuracy in Y dimension. As expected the behavior of the curve is quite similar to the observed in X dimension, in addition the ranges of accuracy are also similar, starting from approximately 0.001m to 0.0017m.

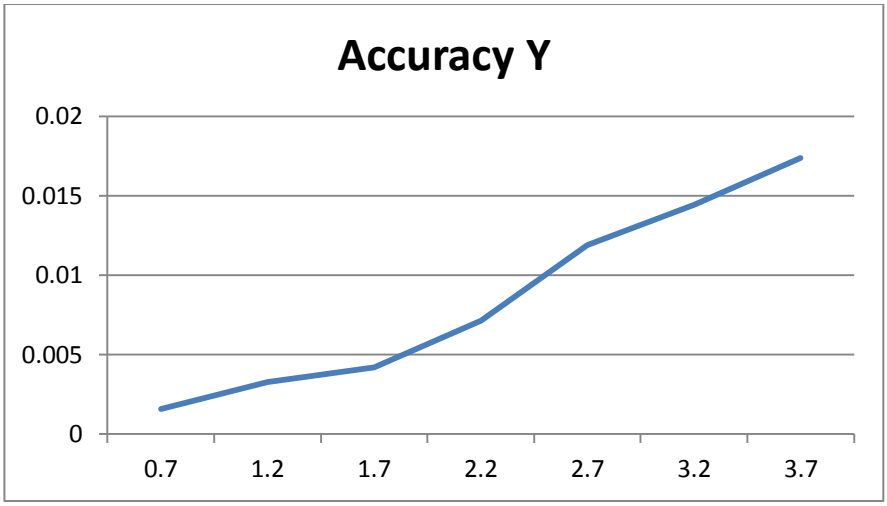

Figure 3.11 Accuracy measurements Y dimension

Table 3.3 shows the values of the measurement for the Z dimension. As it was said before, two methods were proposed to calculate this accuracy. The one in the table 3.3 corresponds to the method using an average of the Z values of the voxels that fall onto the plane object.

Distance	Calculated distance	Difference
0.7	0.6994	0.00065
1.2	1.2028	0.002782
1.7	1.7133	0.01329
2.2	2.23	0.029956
2.7	2.7554	0.055374
3.2	3.2823	0.082314
3.7	3.7487	0.048714

Table 3.3 Accuracy measurements in Z dimension using average

Figure 3.12 illustrates the behavior the accuracy in Z depending on the distance from the sensor. It can be seen that the curve grows exponentially starting with high accuracy at close distance but decreasing considerable its performance when the distance increases.

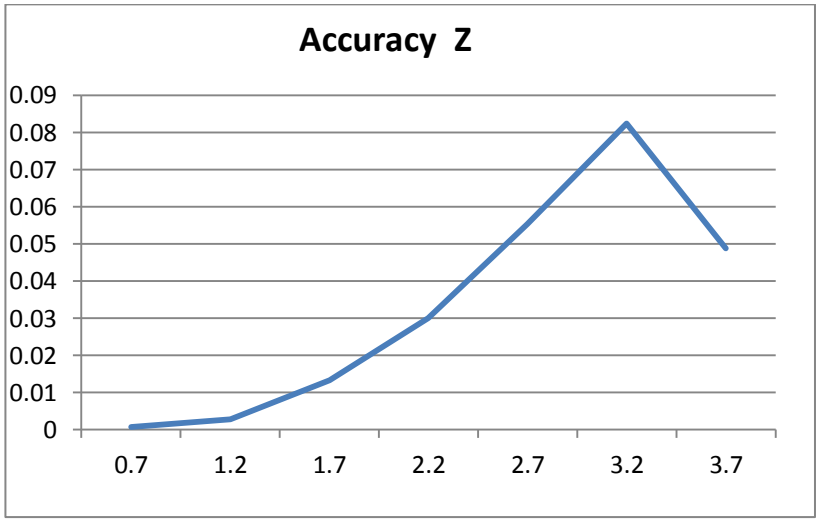

Figure 3.12 Accuracy measurements in Z dimension using average

Table 3.4 and Figure 3.13 represent the measurement of the Z dimension using the plane segmentation method. The results obtained agree with the ones using the average method. The only tangible difference can be seen in the values for the 3.7 meters distance.

Distance	Distance by plane	Difference
0.7	0.70002	2.02E-05
1.2	1.20121	0.0012076
1.7	1.70763	0.0076308
2.2	2.21841	0.0184074
2.7	2.75397	0.0539718
3.2	3.27702	0.0770184
3.7	3.779	0.078998

Table 3.4 Accuracy measurements in Z dimension using plane segmentation

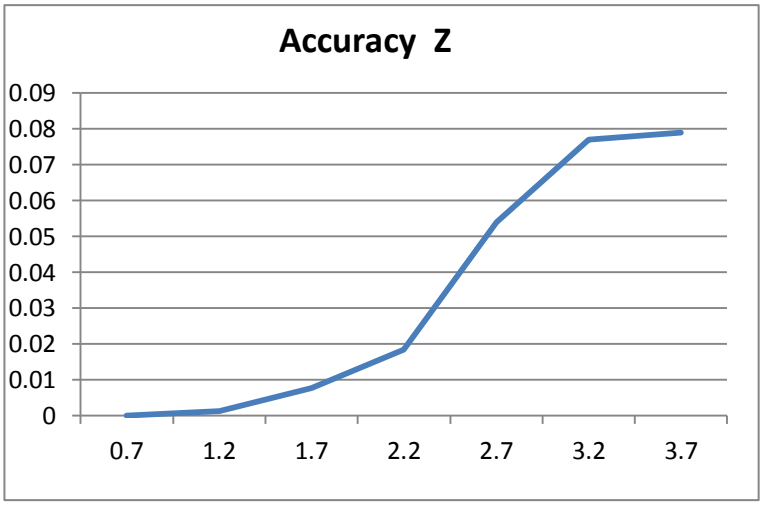

Figure 3.13 Accuracy measurements in Z dimension using plane segmentation

The figure 3.14 shows a general view of the behavior of the accuracy in the three dimensions along the distance. First of all, the accuracy of X, Y and Z is similar when the accuracy is 1.5 meters or lower, after that point the accuracy in Z decreases rapidly. The accuracy for X and Y dimensions increases similarly at a slow pace.

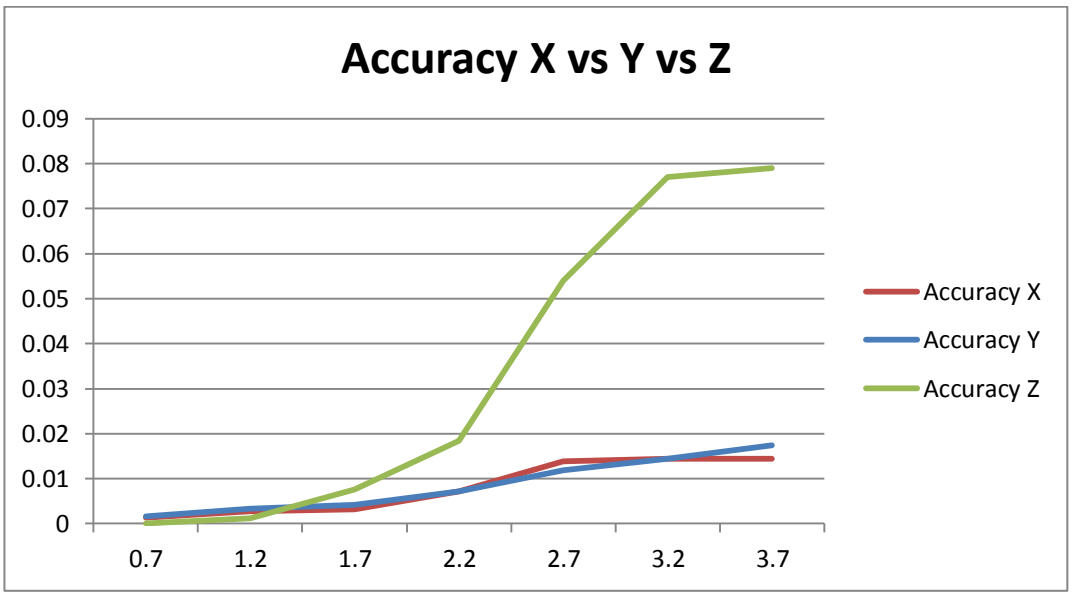

Figure 3.14 Accuracy comparison in X, Y and Z vs distance

3.2.2. Precision results

The precision of the measurements is evaluated using the variance. The table 3.4 contains the results of the variance for the three dimensions along the distance.

Distance	Var X	Var Y	Var Z
0,7	0,000023	0,000023	2,34E-05
1,2	0,0000348	4,26E-05	3,74E-05
1,7	0,0000818	0,000258	0,000125
2,2	0,0001844	0,001082	0,000329
2,7	0,0003328	0,003399	0,000335
3,2	0,0010806	0,007868	0,00112
3,7	0,0025118	0,008564	0,003472

Table 3.5 Variance in X, Y and Z dimensions vs distance

Figure 3.15 shows a graphical representation of the results in Table 3.4. It is possible to identify a similar behavior of the precision and accuracy measurements. Both have a good precision within the range 0.7m to 1.5m but after the distance is bigger than 1.5m the variance in Z dimension increases dramatically compared to X and Y behavior.

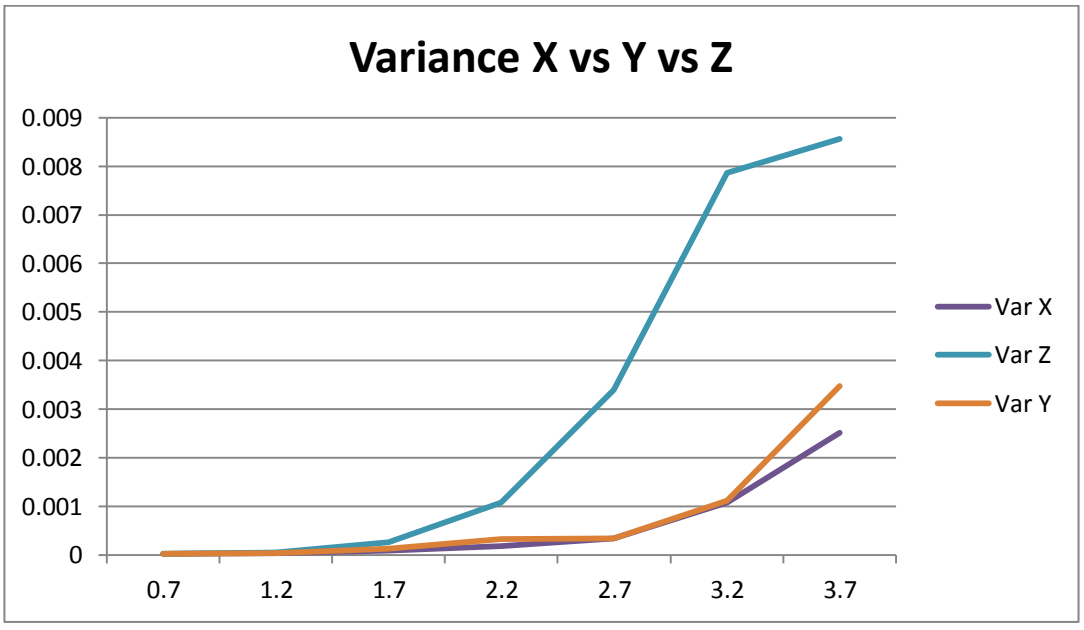

Figure 3.15 Variance comparison in X, Y and Z vs distance

3.2.3. Resolution results

The table 3.5 contains the measurements of the resolution of the sensor measured by the number of voxels found in the plane object. In addition, it also includes a calculation of the theoretical number of points that should have fall on the plane object.

Distance	X Range	Y Range	X voxel density	Y voxel density	Expected # voxels	Average # voxels	Percentage
0,7	0,760138	0,551475	1,187716	1,148906	43566,7	35583,6	81,68 %
1,2	1,303094	0,945385	2,036084	1,969552	14824,8	12526,8	84,50 %
1,7	1,846049	1,339296	2,884452	2,790199	7386,75	6257,4	84,71 %
2,2	2,389005	1,733206	3,73282	3,610846	4410,68	3825,8	86,74 %
2,7	2,931961	2,127117	4,581189	4,431493	2928,35	2498	85,30 %
3,2	3,474916	2,521027	5,429557	5,25214	2084,74	1851,8	88,83 %
3,7	4,017872	2,914938	6,277925	6,072786	1559,36	435	27,90 %

Table 3.6 Resolution measurements

The figure 3.16 shows the theoretical and measured resolutions for each one of the seven distances. The theoretical resolution is shown in blue and the measured resolution in red.

The percentage field in the table represents the relation between the measured number of points and the theoretical one. In the figure 3.15 it is not so easy to spot the dramatic drop of resolution that the sensor suffers at a distance of 3.7 meters. Observing carefully, the relation between the number of measured points and theoretical ones, was bigger than 80% for all the measurements, showing an almost constant behavior. The situation changes at 3.7 meters where the average number of voxels drops to a 27.9%. The most likely explanation to this result is that a 3.7 meters distance is out of the official coverage area of the Kinect and the near infrared beam intensity is not strong enough to be detected.

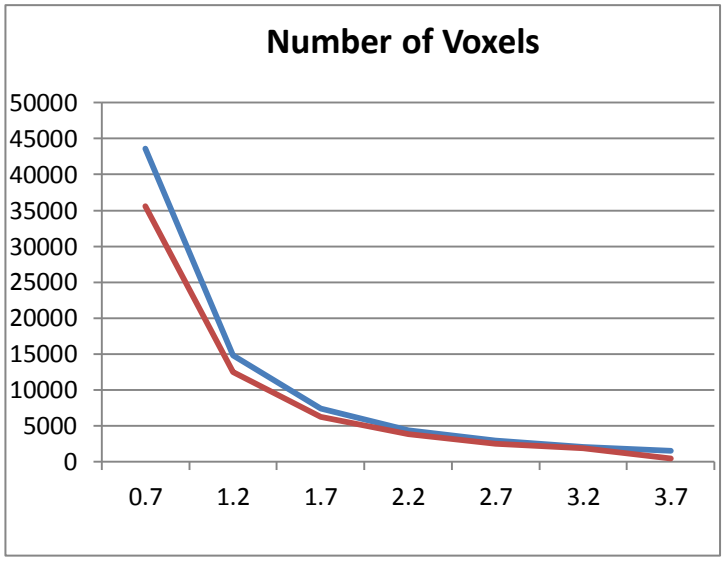

Figure 3.16 Measured and theoretical resolution vs distance

4. MULTI-KINECT SENSOR SYSTEM

In this chapter, a Multi-Kinect sensor system is proposed as an answer to increase the quality of the measurements, making the data suitable for applications that require a higher level of detail. In addition, it also solves occlusion problems and increases the coverage scene.

Object occlusions problems are common in most of the scenes in which an object will cover totally or partially another object. By the use of a several cameras in a scenario, it is possible to capture the blind points that other camera cannot see.

The range of vision of the sensor is other of the limitations that a multi-sensor system can solve. The working range of the Kinect is from 0.7m to 3.2m, if this coverage has to be increased and/or is accuracy have to be increased, we can use a second sensor and combine both streams, obtaining a single point cloud with wider area coverage.

The main goal is to achieve a multi-sensor system, which should be robust and resistant to changes in the scene conditions such as light, irregular shapes, occlusions and so on. Adaptability is also a desired characteristic, since the location of the sensors depends on the scene, thus an optimum coverage can be achieved. In addition, an easy setup and self-calibration are very important because they reduce the human interaction with the system thus avoiding the high impact that the human error can introduce to the system.

The evaluation of the sensor presented in the previous chapter give us tools and reasons to infer that it is possible to increase the performance of the sensor by fusing its data with the stream of other sensor pointing to the same scene.

Since the Kinect sensor offers depth and RGB data, we can think about three different data fusion scenarios: RGB + RGB, Depth + RGB and Depth + Depth. Each one of these scenarios presents different advantages and limitations, for this reason it is necessary to evaluate them and identify the one that accomplish in the best way the goal previously presented.

In the first scenario, only the RGB data from the sensors is contemplated. This means that the depth has to be calculated by doing 2D matching and computing the disparity between both images. The advantages of RGB over the depth data are the sharp edges definition, precise distance between objects in X and Y dimensions regardless the distance from the sensor, and no interference between the sensors. On the other hand, in order to integrate accurately the data from both sensors and determine the depth by disparity, the position of both cameras has to be known precisely. A mis-positioning of the sensors will lead to wrong depth calculations.

Depth calculation by stereo disparity using RGB inputs is a well-known area of study. The quality of the depth calculation depends on the scenario, where colors, shapes, object occlusions and illumination have a drastic effect on the results. This situation is resulting in a lack of stability and robustness of the system, since there is no certainty of the results because they can change radically along with the scenario. Different techniques has been used in order to increase the quality of the measurement such as the addition of a third component, which projects patterns into the scene. This technique works in a similar fashion to the one used in the Kinect and increases considerably the accuracy of the depth calculations under irregular shapes, object occlusions and objects with same color. However, it also increases the complexity of the setup and it is still fragile before change in light conditions.

A second scenario contemplates fusing Depth and RGB data. The results obtained by Chiu in [16] show improvements and also several problems because the sensor does not allow the extraction of Depth and RGB data at the same time. In addition, we will have the same limitations that were presented in the previous scenario in which the conditions of the scene have a high impact in the RGB data. However, in this case we do not have to calculate the depth data since it is obtained from the depth data stream. This depth data can be improved by correlating it with the RGB input and enhancing it using the sharp edges provided by RGB.

It is important to point out that the improvements of this function are dependent on the scene and for this reason are hard to quantify from a general point of view. For example, the results can vary radically if all the objects in the scene have different colors, or if all of them share the same color. In the first case, using the RGB data is possible to identify easily the edges of each object by identifying the color changes. In the second scenario in which the objects share the same color, the edge extraction will be hardly effective.

The third scenario contemplates only the use of depth data from the multisensory system. According to the findings in the previous chapter, the accuracy of the depth data of the Kinect sensor diminishes when the distance from the sensor increases. The effect that distance has in accuracy behaves in different ways for each of the dimensions, being more precise, the Z dimension is affected in a more radical way compared to X and Y dimensions. Taking into account this find, we can infer that the fusion of data of several Kinects will improve the quality of the measurements by smartly combining and conveying values for common voxels according to the highest accuracy of its X, Y and Z components. The main advantage of this scenario is its stability since light conditions, shape of objects or colors do not affect drastically the measurements. On the other hand, the depth measurements lack of the precision than RGB provides, especially in the edge of the objects.

Taking into account the advantages and disadvantages of each one of the three scenarios, the Depth + Depth fusion seems to be a better approach to achieve the goals proposed for this study.

A Multi-Kinect system involves two or more sensors located at different positions pointing to a common scene from different points of view. In order to combine the data from the sensors in the system, two main consecutive processing stages are necessary: multi-sensor calibration and data fusion. The multi-sensor calibration refers to the unification of all the points of view of the sensors to a common one and the data fusion stage is the process in which the common data from all the sensors is combined accuracy wise, so the data is enhanced.

It is important to point out that during the multi-sensor calibration stage, the data is not combined at all, although the point of view of the data captured by each sensor is shifted, each data stream is kept separated.

For sake of simplicity, only two Kinect sensors located at fixed position are used in the implementation of the system. However, the methods and algorithms presented in this chapter can be easily extrapolated to any system with more than two sensors.

4.1. Multi-sensor calibration

In order to combine the data streamed by two Kinect cameras located in different positions it is necessary to calibrate and unify their coordinate system to a common one.

The calibration of two sensors involves six variables, which are three different rotations (yaw, pitch and roll) and translation in each of the dimensions (x,y,z).

In this section, an empirical calibration process is presented with the intention of making a preliminary evaluation of the calibration results, before implementing a more complex and automated calibration method.

4.1.1. Empirical multi-sensor calibration

The scenario used for this calibration is composed by two Kinect cameras and a cube which is the object used as target in the scene. The sensors are located manually orthogonally from each other and separated 1m from the cube as can be seen in the figure 4.1. It is important to notice that the positioning of the objects was done manually without using any professional measurement tools more than a measuring tape and a protactor, thus increasing and making evident the presence of human error.

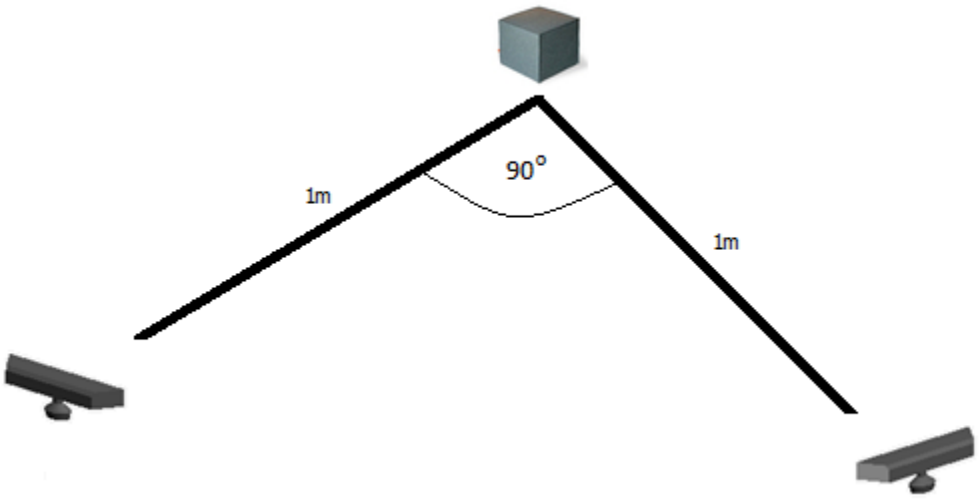

Figure 4.1 Scenario for empirical calibration

A software program is created in order to capture, process and display the data captured by both sensors. The input of one of the sensors is shifted according to the setup of the scene, meaning that it is rotated 90º and shifted in X and Z dimensions 1 meter. If the positioning of the scene were perfect, both data streams would have matched perfectly producing a calibration between both cameras, but due to human error this is not the case.

In order to indicate when both data streams were close, the software program extracts the edges of the target object for both inputs and shows them in the screen. Thus, to complete the calibration one of the cameras is moved manually until they seem to match according to the feedback in given by the visual representation.

The purpose of this empirical method is to make a fast evaluation of the possible results of the calibration. Although it has several disadvantages such as high human interaction and time consumption, it was useful to get a preview of the expected results that a more complex and automated calibration method should provide.

4.1.2. Automatic multi-sensor calibration by using a common plane

In the search of an automatic calibration method, other approaches to calibrate the sensors were examined, such as registration using Iterative Closest Point and SIFT keypoints[17][18]. Although the results were good when the distance and view angle of the sensors was small, the quality of the results decrease considerably when the sensors were located at far points of view. The computational cost of these methods is high, but this does not guarantee a high accuracy calibration of the cameras. In addition, the results of the calibration process were not consistent, in some cases producing results with a mismatch of more than 5cm.

One of the conclusions after using the registration algorithms is that, they are not able to take advantage of the already known parameters, and they execute the same procedure regardless the complexity of the scene. Due to this reason, a customized calibration method is proposed to take advantage of the fix scene and the plane object that is used as a reference to determine the position and angle of view of the sensors.

The method described in this section takes advantage of the 3D capturing capabilities of the Kinect camera, existing plane segmentation algorithms, analytic geometry and Euclidean geometry.

In the proposed scene, the sensors are located at fix positions, therefore we only need to execute the calibration process once. It is important to notice that the calibration is dependent on the location of the sensors and the reference plane object, meaning that if one of them is moved, the calibration has to be executed again.

Although, the use of several sensors makes possible to enhance and increase the amount of information in the point clouds, it also might produce artifacts in the captured data. As it was explained in the section 2.4, the infrared beams can be distorted by interference generated by signals with a similar wavelength. Both sensors use the same wavelength in its infrared beam, thus, it is possible to have collisions affecting the wavelength of the infrared light. Since the capture device filters out the data taking into account its wavelength, many of the beams affected by the interference will be discarded.

In addition to the interference problem, it is possible that one of the sensors reads by mistake the infrared light projected by the other sensor. After several tests, the conclusion is that this problem does not affect heavily the measurements. However in order to avoid any interference problem during the calibration process, the data will be captured once at a time for each sensor.

The scene, which is necessary to execute this method is composed by two Kinect sensors and a squared rigid plane surface. The surface has to be located in an angle in which both Kinect cameras are able to capture the whole plane surface. In addition, the sensors and the plane object have to stay in a fix position. Figure 4.1 shows an example of the scene.

Figure 4.2 Scenario for automatic Multi-Kinect calibration

It is important to notice that the cameras and the plane object can be located at any arbitrary position, and using different angles of view in any of the dimensions. The only constraint is that the view range of both sensors is able to see the plane surface of the object. In addition, the object should be located at a close distance from the sensors, this is not a mandatory constraint, but it is important to take into account since the accuracy of the calibration decreases along with the accuracy of the sensors.

The goal of this method is to obtain a transformation matrix able to unify successfully the measurements of two Kinect cameras under a common coordinate system. This transformation matrix is a 4x4 matrix composed by a rotation matrix (3x3) and a translation vector (3x1), the remaining (1x4) vector is not used[19].

The calibration process is divided in two phases which combined will generate a transformation matrix containing rotation and translation coefficients.

The first phase is to calculate the rotation matrix, this is achieved by taking advantage of the plane surface of the target object and plane segmentation algorithms. The flow of the process can be observed in the figure 4.3.

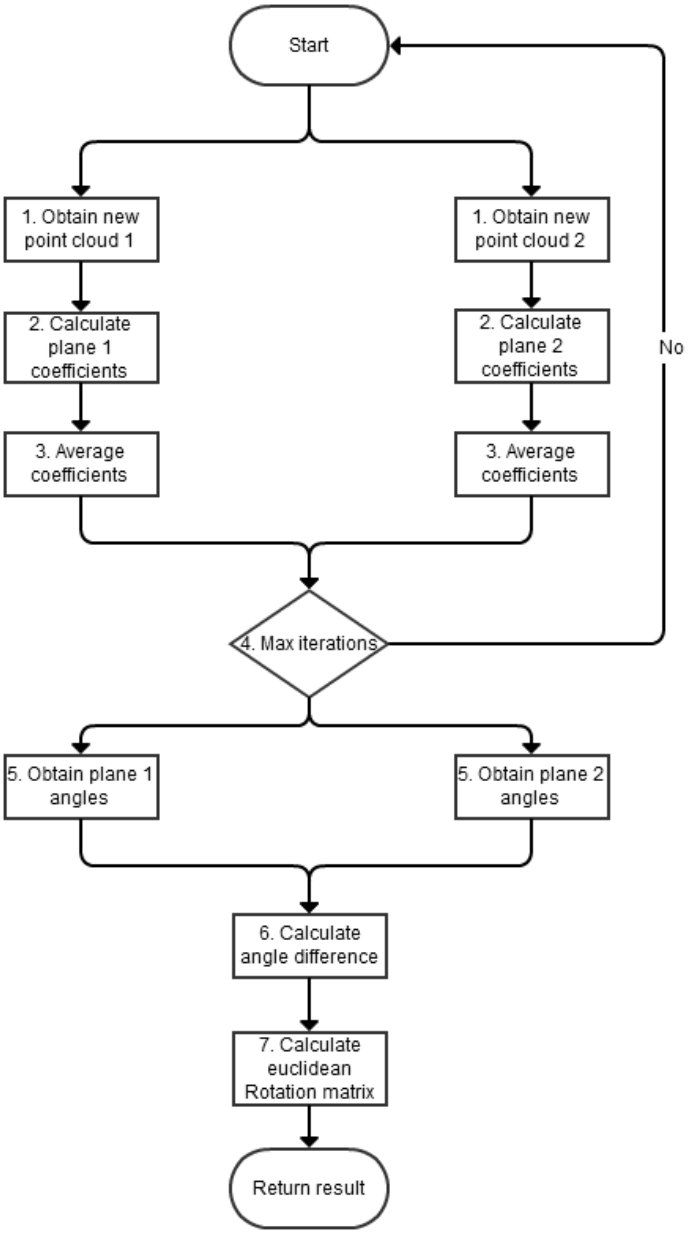

Figure 4.3 Rotation matrix calculation

The first step in the flow obtains a new point cloud from each of the sensors. One of the point clouds is tagged as the base and the other one as the target. Thus, the rotation matrix will represent the rotation needed to apply to the base point cloud in order to have the same angle orientation than the target cloud.

In the second step, a plane segmentation algorithm is used in order to detect the plane object. In order to identify the plane, the normal of each one of the voxels is calculated and grouped along with the voxels with similar value. Voxels which normal is distant from the average are discarded. When the plane normal is detected, it is possible to identify each one of its coefficients and represent the plane with the general plane equation, which is shown below[20]:

$$Ax + By + Cz + D = 0 \tag{4}$$

The coefficients A, B and C are a vector representing the normal of the plane and D is a constant, which represents the distance from the origin to the plane.

In the third and fourth steps, the coefficients that compose the plane equation are stored and averaged with the ones corresponding to another iteration of the method. This process is executed in a loop in order to obtain a stable and more correct set of coefficients.

In the step five, the angle between the planes and the three coordinate axes is calculated. In order to achieve this, we use the directional cosines equations on each of one of the dimensions. The directional cosines are calculated using the equation 5 [21]:

$$\begin{aligned} \cos(\propto) &= A/||V|| \\ \cos(\beta) &= B/||V|| \\ cos(\varphi) &= C/||V|| \end{aligned} \tag{5}$$

In these equations A, B and C correspond to the coefficients of the plane equation and ||V|| is the norm of the vector given by the equation 6:

$$||V|| = \sqrt{A^2 + B^2 + C^2} \tag{6}$$

At this point, we know the value of the angles between the plane and the coordinate axes for each of the planes, therefore, in order to know the difference between the angular point of view of the sensors it is enough with finding the difference of each one of the angles in X, Y and Z dimension. The process is shown in the equation 7:

$$\begin{aligned} \Delta\propto &= \propto_1 - \propto_2 \\ \Delta\beta &= \beta_1 - \beta_2 \\ \Delta\varphi &= \varphi_1 - \varphi_2 \end{aligned} \tag{7}$$

Now that the difference between these angles is known, we can determine the rotation matrix by using the equation 8 for the Euclidean rotation matrix [21]:

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

(8)

The resultant matrix and output of the first part of the method corresponds to a rotation matrix able to transform the base plane in order to put it in a common angular point of view with the target plane.

The second part to the process corresponds to calculate the translation vector necessary to move the base cloud to the same position than the target cloud.

A flow diagram with the process to calculate the translation vector is shown below in figure 4.3.

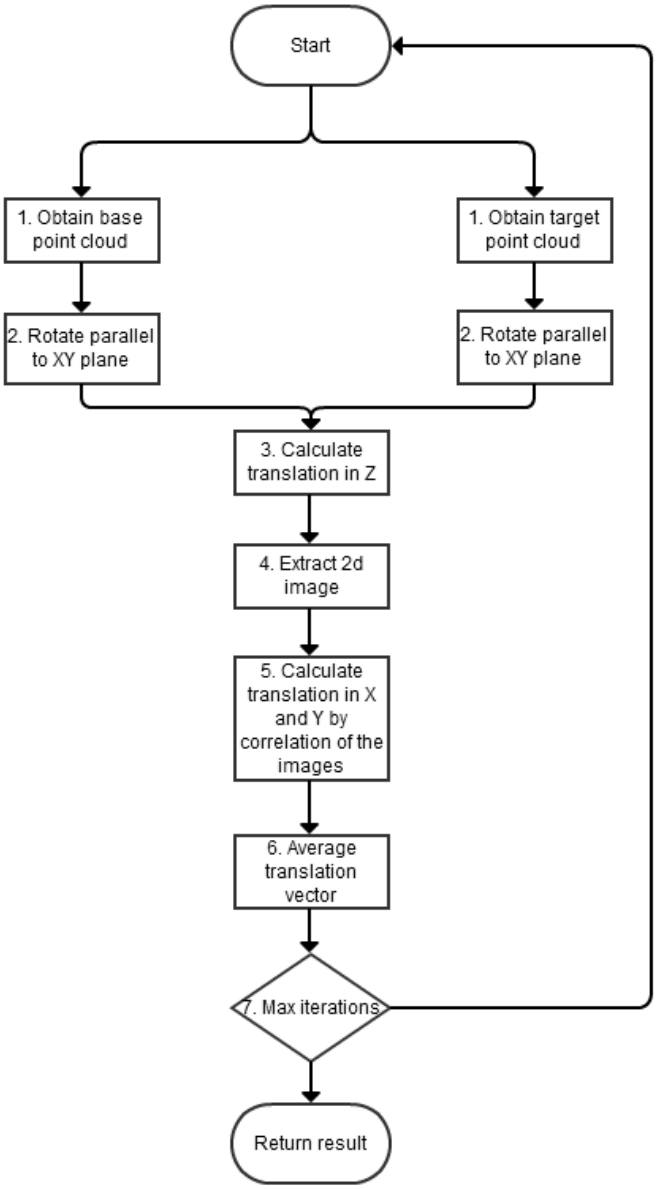

Figure 4.4 Translation matrix calculation

The second part of the process starts right away after the rotation matrix has been calculated and applied over the base cloud. Meaning that both point clouds have the same angle orientation and the problem is reduced to calculate the translation in X, Y and Z dimension.

In order to take advantage of the scene, from which we know the equation representing each of the planes and the rectangular shape of the plane object, we can adjust the coordinate system to create a simpler calculation process.

In the first step, two new point clouds are captured and the rotation matrix is applied over one of them. In the next step a new rotation matrix is calculated, this rotation matrix should be able to rotate both point clouds parallel to the XY plane.

In the third step, as result of the rotation and parallelism of the plane object and the XY plane, the D coefficient of the plane equation corresponds to the difference between

the origin and the Z value for the plane. Therefore, in order to calculate the translation in Z dimension it is enough to calculate the difference between the D coefficients of the plane equations representing each of the planes.

Another advantage of rotating both point clouds parallel to the XY plane is that, it allows us to analyze the X and Y dimensions as a 2D image and apply much simpler and faster algorithms than in a normal 3D scenario.

Although both clouds have the same angle orientation and both are parallel to the XY plane we still cannot treat them as 2D images because they have different scale. Since the Z translation is already known, it is applied to the base cloud, so both, target and base cloud are not only observed from the same angle of view but from the same distance in depth. Now the problem is reduced to find the translation in X and Y dimensions.

In the fourth step, the point clouds are processed and stored in a 2D matrix, since we are not using color images and the depth is not taken into account, there are only two possible values in the matrix, zero or one, representing the existence of data.

In the fifth step number, the correlation between both 2D matrixes is calculated. Before executing the correlation algorithm, both images are smoothed using a Gaussian filter with a window size of 16x16. Following the idea in [22], the images are smoothed in order to improve the edges definition and thus the results of the correlation process. It is important to point out that smoothing process does not lose information since we are dealing with a plane object. Afterwards a correlation between both images is calculated using the squared difference template matching method[23][24], which is represented by the equation 9.

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \qquad (9)$$

In the equation 9, I denote the base image, T the template and R the result of the matching. The result of this method is a matrix containing all the correlation results for each of the positions in which the template was located. The best matching can be identified by finding the minimum value in the resultant matrix.

After identifying the best matching between both images, we can identify the translation coefficients for X and Y dimensions, thus obtaining the full transformation matrix.

In steps six and seven the full process is executed several times averaging the values obtained for the translation vector. When the total number of iterations is reached, the final translation vector is returned.

After completing the two main steps of the calibration process, the result is a transformation matrix composed by the rotation and translation matrix. It is important to notice that the calibration process is not 100% accurate since it depends on the values captured by the sensor. Therefore, the accuracy of the calibration decreases with the distance from the plane object to the sensors.

4.1.3. Multi-sensor calibration results

The figure 4.5 shows the results of the experimental calibration. After the sensor is located in a position that satisfies the calibration indicator, the target object is changed by objects with different shapes and located at different positions within the common range of vision of both sensors. The colors green and red represent different the data captured by each one of the sensors.

Figure 4.5 a), b) and c) show two boxes located as target in the scene from different points of view. As can be seen, the data from both sensors matches regardless the position of the object. In d) and e), the target object has a rounded shape, which differs in shape from the cube used to perform the calibration. The obtained results are good even though the calibration was done manually.

As mentioned before, the experimental calibration was performed in order to prove the multi-sensor idea and identify in an early stage if it is possible to improve depth by fusion data from several sensors. The results let us see that it is possible to overcome occlusion problems, increase the area and points of view of coverage and improve the quality of the data especially in the edges.

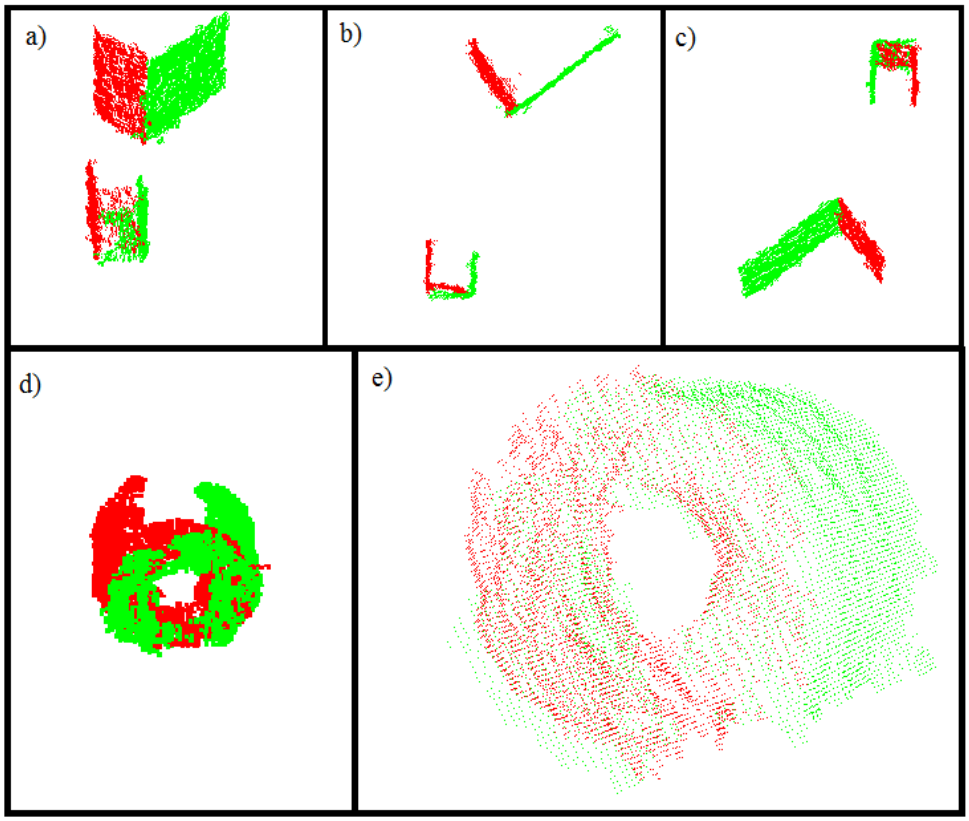

Figure 4.5 Results of empirical calibration.

Figure 4.6 shows the scene used for the auto calibration process, as it can be seen, the angles and locations between the sensors are arbitrary. In addition, we use the plane side of a box as the plane object for the calibration, although the sensors are able to capture other feature of the box, these are discarded and not taken into account.

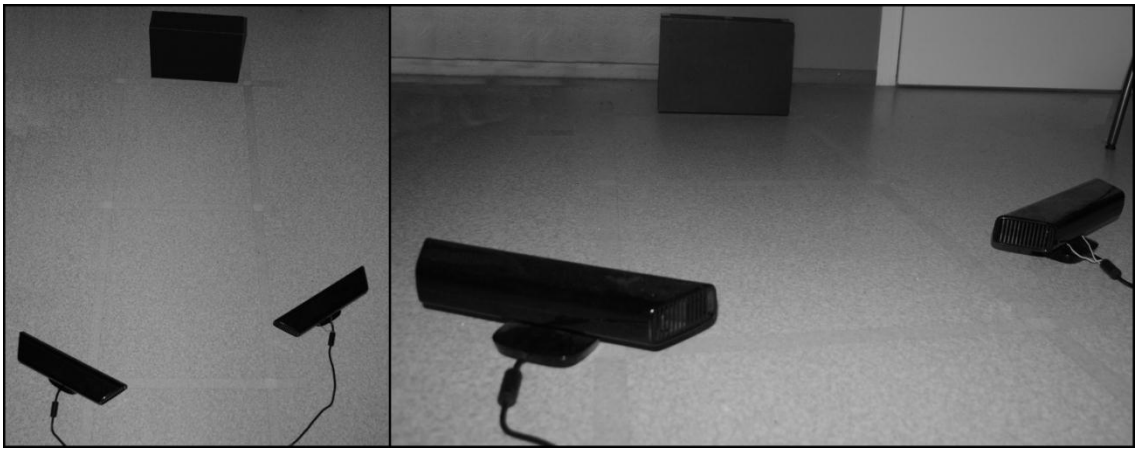

Figure 4.6 Scene used for auto calibration process

The figure 4.7 shows the results of each of the steps performed in the auto calibration process. Figure 4.6 a) shows a graph of the plane equations of both planes, as can be seen the planes have different angles comparing to the origin and their location is also different. After the rotation matrix and the translation coefficient in Z are calculated, the temporary transformation matrix is applied to one of the clouds. The result is shown in b). At this point both clouds share the same view angle and depth. Figure 4.6 c) and d) show the 2D images extracted from the point cloud. Figure 4.6 e) shows the result of the template matching between both 2D images. Figure 4.6 f) shows the final result of the calibration process after the translation coefficients for X and Y are calculated and applied to the cloud.

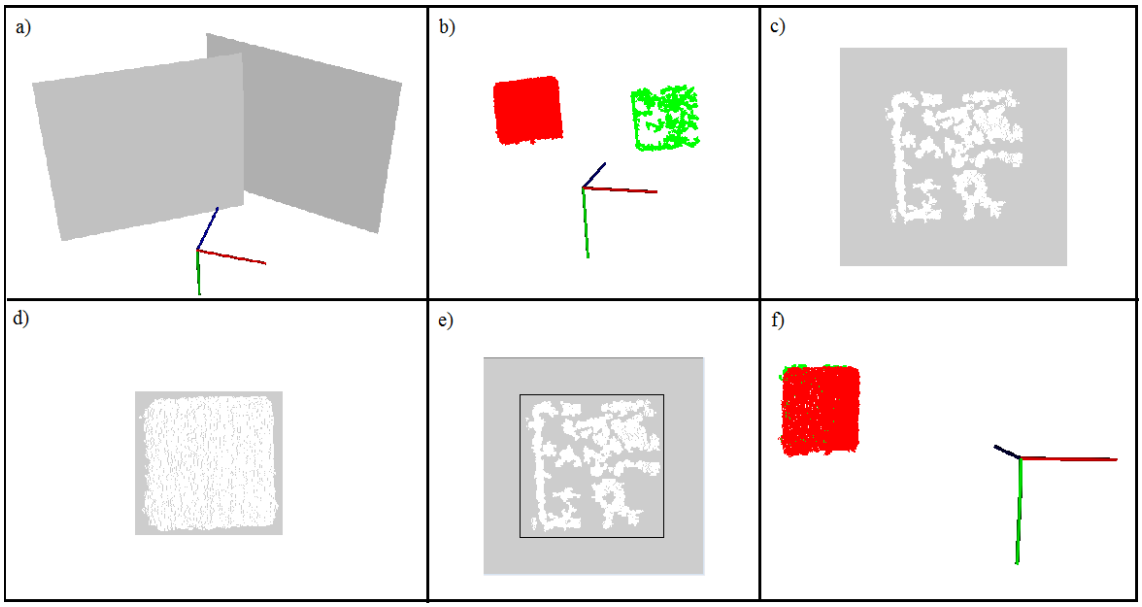

Figure 4.7 Multi-Kinect sensor auto calibration process.

The results shown in the figure 4.6 are presented exactly as they are after execution of the software algorithm. After the algorithm is executed there is no human interaction in order to complete the calibration, thus decreasing the introduction of human error.

This calibration process is very versatile, being able to calibrate sensors at any angle or position, as mentioned before the only constrain is that both sensors should be able to see the face of the plane object that is used as a reference for the calibration. In addition, the distance from the cameras to the target is recommended to be short, taking into account that the results of the calibration depend highly on the accuracy of the sensor.

4.2. Data fusion

After the Kinect sensors are calibrated and sharing a common point of view and coordinate system it is possible to continue with the data fusion process. At a glance, it might seem a good option just to add a point cloud from one sensor to the point cloud from the other once since both share the same coordinate system, however, this process is not possible due to the lack of stability in the accuracy and precision of the measurements from both sensors.

As an example of this problem we can think about a scenario in which two Kinect sensors are located at fixed distance and one of them is positioned at 90 degrees with respect to the other one, there is a target object located at 2.7 meters from both sensors. The result of a simple overlapping will be a mismatch of the values, because the accuracy in the Z dimension of one of the sensors will have to agree with the values obtained by the other sensors in the X dimension. Since the accuracy in both dimensions is different by a wide margin we will have a mismatch of more than 5 cm according to the results in accuracy of chapter 3.

The correct fusion process can be divided in two stages. The first one is to identify common voxels or voxels group between two point clouds. The second one is to pro-

pose a criteria to merge the values of the common voxels taking into account that the accuracy varies depending on the dimension and distance from the sensor.

It is important to realize that in difference to the calibration process the data fusion is executed in real time while the sensors are capturing the data and producing the fused enhanced point cloud. There are no objects fixed at specific positions or under specific conditions, the point cloud fusion is meant to work with different target scenarios including various shapes, materials and light conditions.

Taking into account the previous analysis of the problem and the sensor evaluation results presented in chapter 3, we can infer that the fusion process can be divided in two different scenarios depending on the accuracy and variance of the voxels. Basically, we have one scenario in which the all voxels share a relative stable and high accuracy and variance. A second scenario is the one in which voxels are characterized by different accuracies and high variances.

The results presented in chapter 3 show that the accuracy and variance are good, stable and behave in a similar way for all three dimensions between 0.7m and 1.5m. The uncertainty of the accuracy in this range is ± 5mm, which is very accurate comparing to the one found at bigger distances reaching almost 80mm in Z dimension.

Given that the behavior of each of the dimensions is similar within the 0.7m to 1.5m range and changes radically at bigger distances, it is possible to use two different approaches to the problem. The first one will take the range between 0.7m to 1.5m for stable and good accuracy and variance and the second one for distance bigger than 1.5m in which the accuracy and stability of some voxels decreases.

For the first scenario in which the voxels are within the 0.7 to 1.5 m distance, we propose a 1 to 1 addition of the common voxels of both areas. The common and stable accuracy of the voxels in this area allow us to obtain a very close matching of the voxels. After the addition of both cloths, we propose the execution of an outlier removal algorithm, which as a criteria takes into account the number of neighbor voxels per area. For example, if a voxel does not have two or more neighbor voxels in an area of 1 cm^3 it will be removed. Taking into account that after the addition of the clouds the number of voxels increased considerably, we can execute the removal of outliers with strong deviations in order to guarantee that only voxels very close to each other will remain in the cloud, therefore increasing the accuracy of the final cloud.

A fusion method for voxels at distances bigger than 1.5m is also proposed in this document: The first step is identifying if there are points in common between both point clouds. This is not a trivial process because the accuracy of the data is not constant and in the worst-case scenario, voxels representing the same object might differ in distances bigger than 14 cm.

In order to detect if there are common points between the clouds, we divide and correlate the clouds using the squared difference matching method in which the search window and target window sizes depend on the estimated accuracy of the measurement, which is calculated using the results of the evaluation of the sensor in chapter 3.

The worst-case scenario for accuracy was taken into account to determine the size of the windows, meaning that in order to determine the correlation in areas close to the sensor, the size of the search and target windows is smaller than when the object is located a far distances from the sensor.

As an example, voxels detected at a distance of 3.7 meters will be wrapped in a window of 8x8 cm and a correlation process will be executed in a 32x32cm search window. For voxels located at 1.7 meters the target window size will be 1x1cm and the search window 4x4cm. The use of this method helps to keep the detail level, takes advantage of high resolution at short distances and is able to correlate low-resolution areas by the use of big enough windows.

After common areas have been identified, it is necessary to fuse its values. Since we know that the dimensions X and Y have a higher accuracy than Z especially at far distances, we use this as criteria to give prevalence to the voxel values when making the fusion.

Since the clouds have been rotated during the calibration process, the accuracy and precision of its component has also change and now depend on the angle in which the cloud was rotated. This situation implies that in order to determine the accuracy of a coordinate of a voxel it is not enough to check its accuracy regarding distance but to obtain the new value after it has been rotated using the angle of rotation. After the accuracy is determined for the voxels that are going to be fused, we can select which one has a higher accuracy and replace the value of the one with the lower accuracy.

As a trivial example, if the sensors are located orthogonally from each other and one object is located at a distance of 3.2 m from both sensors, we can infer that the measurement in Z corresponds to the same in X for the other sensor. Due to different accuracies, the values are different, and taking into account the experimental results of accuracy, we will obtain an accuracy of 1 cm if we use the value of the X and 7.5 cm if we use Z. At this point is clear that the enhancement in accuracy is given mainly in the Z dimension. A full flow of the process can be seen in figure 4.8.

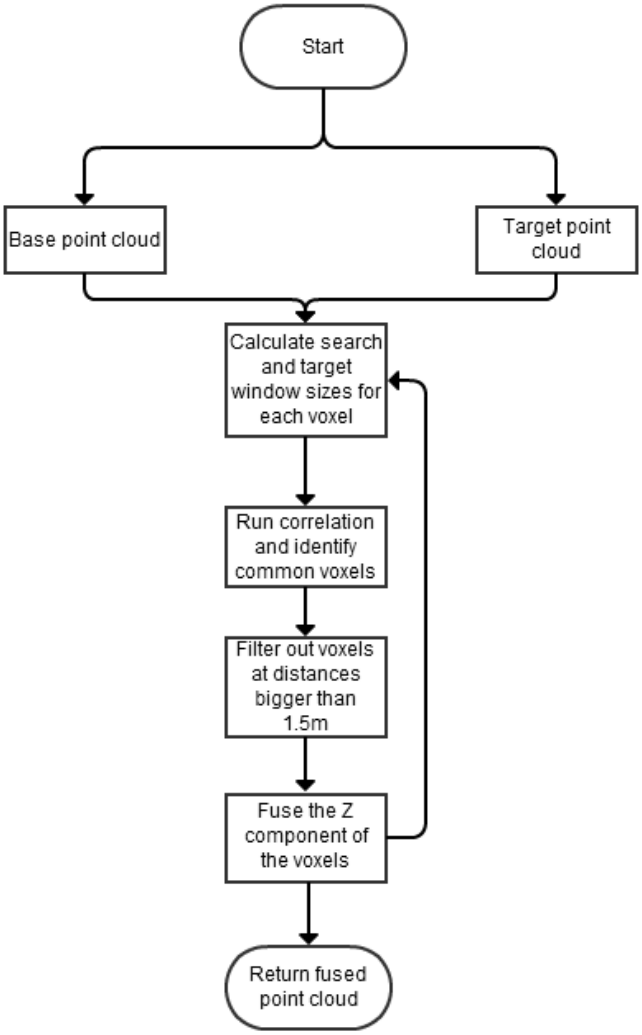

Figure 4.8 Point cloud fuse flow

5. CONCLUSION

This thesis focused on three main points: Understanding clearly the 3D sensing mechanism of the Kinect sensor, making a tangible evaluation of its performance and using the results of this evaluation to build a Multi-Kinect system able to overcome its limitations.

Operational descriptions of the Kinect 3D sensing mechanism can be found in several patents, books, articles and internet resources. After reviewing each one of them, the conclusion is that, none of them provide a full description of the mechanism. In addition, many suppositions are made due to the fact that the Kinect technology has not been disclosed and most of the available information is the result of reverse engineering.

As a result of the study of the Kinect sensor, a full review of the literature regarding its operation was collected, analyzed and summarized in the document providing a full description of the 3D sensing mechanism.

The literature mention several limitations of the Kinect sensor, among them, the lack of detailed accuracy. The definition of this limitation is not clear or numerically tangible. For this reason, a full evaluation in accuracy, precision and resolution was performed, identifying how these three variables vary along the distance from a target object to the sensor.

As a result, we were able to identify that inside the range of 0.7 m to 1.5 m, the accuracy and precision are stable for the three dimensions. In addition, its accuracy has an uncertainty lower than 5mm, which is very good. On the other hand, the behavior changes at distances bigger than 1.5m, the accuracy and precision of the Z dimension drops rapidly compared to the X and Y dimensions. The uncertainty at a distance of 3.7 m in Z increases almost to 80 mm in contrast to the one in X and Y, which remain under 20 mm.

The evaluation of resolution was done comparing the measurement with the theoretical amount of points that should fall in the target area. As a result, we found that a little more than 80% of the expected points fall onto the area and this behavior remained constant until a distance of 3.7 m in which the resolution dropped to a 20% of the theoretical value. Taking into account the specifications of the Kinect sensor, the main reason to this drop of resolution is the intensity of the near infrared beam, which at that distance is not strong enough to be detected.

As part of the multi-Kinect system, an automatic calibration method was described and implemented in software. The results of the calibration are satisfactory and allow a fast, adaptable and robust calibration of sensors located at any position and angle with the single constraint of pointing to the same reference object. Since the method is auto-

matic, it avoids human interaction and thus human error is not introduced to the calibration, contrary to what happened with the experimental calibration.

The data fusion method takes into account the results of the evaluation of the sensor in order to identify what considerations are necessary to combine voxels of each point cloud. Two data fusion scenarios are identified. The first one for short ranges and high accuracy in which voxels are combined one to one and later filtered by an outlier removal. The second scenario contemplates voxels at distances higher than 1.5m where the accuracy and precision are low. For this case, we propose and adaptive template matching where the size of the search window and template depend on the accuracy of the area that is being processed. This results into small window size for good accuracy voxels and bigger windows for lower accuracy. The size of the window depends on the worst matching case scenario between the two areas that will be fused.

The methods presented in the data fusion section are only theoretical and have not been implemented. In order to encourage and simplify the software implementation of them in future work, we include the source code of the automatic calibration and a basic description of its design in Appendices 1 and 2.

REFERENCES

[1] "Kinepeutics: Physical Therapy with the Kinect." [Online]. Available: http://kinepeutics.blogspot.fi/. [Accessed: 15-Nov-2012].

[2] "BetaKit » Bodymetrics Launches Kinect-Powered Body-Sizing Pods At Bloomingdale's." [Online]. Available: http://betakit.com/2012/08/09/bodymetrics-launches-kinect-powered-body-sizing-pods-at-bloomingdales. [Accessed: 15-Nov-2012].

[3] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3.

[4] L. Cruz, D. Lucio, and L. Velho, "Kinect and RGBD Images: Challenges and Applications," *SIBGRAPI Tutorial*, 2012.

[5] Zalevsky, A. Shpunt, A. Maizels, and J. Garcia, "Method and System for Object Rec onstruction", 2006, US 2010/0177164 A1

[6] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns", 2007, US 2008/0240502 A1.

[7] D. Fofi, T. Sliwa, and Y. Voisin, "A comparative survey on invisible structured light," in *Proceedings of SPIE*, 2004, vol. 5303, pp. 90–98.

[8] C. D. Mutto, P. Zanuttigh, and G. M. Cortelazzo, *Time-of-Flight Cameras and Microsoft Kinect™*. 2012.

[9] J. Kramer, M. Parker, D. H. C, N. Burrus, and F. Echtler, *Hacking The Kinect*, New. APRESS, 2012.

[10] A. Shpunt and Z. Zalevsky, "Three-Dimensional Sensing Using Speckle Patterns", 2007, US 2009/0096783 A1.

[11] B. Freedman, A. Shpunt, and Y. Arieli, "Distance-Varying Illumination and Imaging Techniques for Depth Mapping", 2008, US 2010/0290698 A1.

[12] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," 2011, pp. 1–4.

[13] A. Menditto, M. Patriarca, and B. Magnusson, "Understanding the meaning of accuracy, trueness and precision," *Accreditation and Quality Assurance: Journal for Quality, Comparability and Reliability in Chemical Measurement*, vol. 12, no. 1, pp. 45–47, 2007.

[14] Khoshelham, "Accuracy analysis of Kinect depth data," *GeoInformation Science*, vol. 38, no. 5/W12, p. 1, 2010.

[15] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for experimenters: design, innovation, and discovery*, vol. 13. Wiley Online Library, 2005.

[16] U. B. Wei-Chen Chiu and M. Fritz, "Improving the Kinect by Cross-Modal Stereo," in *Proceedings of the British Machine Vision Conference*, 2011, pp. 116.1–116.10.

[17] R. Lemuz-López and M. Arias-Estrada, "Iterative Closest SIFT Formulation for Robust Feature Matching," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. Nefian, G. Meenakshisundaram, V. Pascucci, J. Zara, J. Molineros, H. Theisel, and T. Malzbender, Eds. Springer Berlin Heidelberg, 2006, pp. 502–513.

[18]"How to use iterative closest point." [Online]. Available: http://pointclouds.org/documentation/tutorials/iterative_closest_point.php. [Accessed: 15-Nov-2012].

[19] F. Brickell, *Matrices and Vector Spaces*. Taylor & Francis Group, 1972.

[20] N. J. Mitra and A. Nguyen, “Estimating surface normals in noisy point cloud data,” in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 322–328.

[21] P. J. Ryan, *Euclidean and Non-Euclidean Geometry: An Analytic Approach*, First ed. Cambridge University Press, 1986.

[22] W. J. Tam, G. Alain, L. Zhang, T. Martin, R. Renaud, and D. Wang, “Smoothing depth maps for improved stereoscopic image quality,” in *Proc. SPIE Conf. Three-Dimensional TV, Video, and Display III*, 2004, vol. 5599, pp. 162–172.

[23] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons, 2009.

[24] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

APPENDIX 1: SOFTWARE IMPLEMENTATION DESCRIPTION

The software implementation of the evaluation methods, experimental calibration and automatic calibration is available in a CD-ROM, which is attached to the thesis. The CD-ROM only contains the source code of the solution. Additional libraries are not included. For this reason, in order to execute and reuse it is important to take into account the next dependencies:

PCL 1.6
OpenCV 2.4.0
Boost 1.49.0
FLANN 1.7.1
VTK 5.8.0
QT 5.8.0
QHull 2011.1
OpenNI 1.5.4

The dependencies and the software solutions are portable between platforms, so it can be executed in Linux, Windows or Mac environments.

APPENDIX 2: CLASS DIAGRAM OF SOFTWARE IMPLEMENTATION