University of Padua

FACULTY OF ENGINEERING
Department of Information Engineering

Dynamic 3D Sensors: Data Characterization and Post–Processing

Laurea Magistrale in Informatic Engineering

Supervisor:
Prof. Pietro Zanuttigh

Graduate:
Lucio Bezze

Co-Supervisors:
Ch.mo Prof. Guido M. Cortelazzo
Ing. Carlo Dal Mutto

July 12th, 2011

Academic Year
2010–2011

Dedicated to my family and to Valentina

Abstract

After a brief introduction about Time–of–Flight range cameras and 3D sensor of Microsoft Kinect characteristics, a deep analysis on statistical distribution of data retrieved from these sensors, is performed. A set of algorithms and procedures are designed and implemented to improve the general quality of the depth–maps acquired, on the basis of the problems highlighted. They are computed in particular denoising and upscaling operations, through the use of an innovative and smart smoothing filter, the trilateral filter. The main attention is focused towards Kinect sensor, but the procedure can be adapted to other setting of utilizations. In the end they are presented experimental results, applications and further improvements.

Contents

Introduction

Even tough computer vision is one of the branches of Information Technology that has more involved researchers all over the world, and it contains inside a huge fascinating potential, it is clear how in these years it had not taken to real production of commercial products, but it had simply performed vast forwarding steps from a theoretical and algorithmic point of view.
During last years the interest against different devices that are capable to retrieve information about geometry of surrounding environment, has grown up. This kind of information is very interesting to try to build even more realistic model about real world. While normal cameras detect exclusively color informations, these devices can retrieve the distance of every point of the subject framed. Naturally the state-of-the-art of this research field, and also high costs of materials and hardware deployment of these sensors permitted, until recently, the production of restricted-access prototype, and not large-scale products.
In the recent past, Microsoft released to general public a device of very low costs, but preserving inside the previous named characteristics of depth and shape retrieving. Such device, Microsoft Kinect, although designed for gaming purpose (in combination with the well known game console Microsoft Xbox 360), paves the way to several applications, above all the ones that need an accurate knowledge of tridimensional surrounding space. The purposes of possible utilization range over robotic, video-surveillance, acquisition and subsequently the fruition of multimedia 3D contents, industrial ambit, and so on.
The more interesting ones are certainly those that in the past became set apart, not only for low quality levels, but above all for huge realization costs. The work that is illustrated in this thesis, is the achievements of more information about the measurements behaviour of those kind of 3D dynamic sensors. Next step is an extensive analysis of data acquired from real scenes, specifically built to highlight differences, strengths and weaknesses of these 3D sensors. On the basis of the considerations made upon analysis, it has been proposed a method to improve the quality of the data retrieved, with particular attention against Microsoft Kinect device. This method contains several algorithms, some of them are well known, and other ones are more

innovative and less utilized. In the end experimental results are shown and compared each other, with some consideration on the overall performances and further improvements presented.

Chapter 1

General Description of 3D Sensors

1.1 Time Of Flight Range Cameras

Time-of-Flight cameras are sensors which are designed to retrieve geometrical information from surrounding environment. In particular the technology included inside permits the acquisition of the depth, thanks to the measurements of the time of flight of a signal emitted and subsequently received back to the sensor.
The devices considered on further analysis are MESA SR4k[2] and Canesta Range Camera (now Microsoft); both cameras are based upon Time-of-Flight technology, but they present different hardware implementation and performances.

1.1.1 ToF technology

The skill to measure the distance is based upon an optical system of time of flight. A modulated infra-red radiation is emitted by the sensor and hits the objects in the environments. The reflected waves are received, and a measurement of the time t used to travel, permits a simple calculation of the distance d. In fact considering the speed c of the irradiated waves as the light speed (i.e. $c = 3 \cdot 10^8 [m/s]$) becomes clear that

$$d = \frac{c \cdot t}{2} \tag{1.1}$$

The real physical data that is measured from the sensor is the phase shift $\Delta\varphi$ between sent and received signal. This phase delay allows to calculate the time of flight t of the wave. The signal that is produced at the source is a sinusoidal signal, which is modulated with amplitude modulation (AM) to permit the transmission over the air channel. The signal $s(t)$ transmitted is of the form

$$s(t) = A \cos(2\pi f_c t) \cos(2\pi f_s t) \tag{1.2}$$

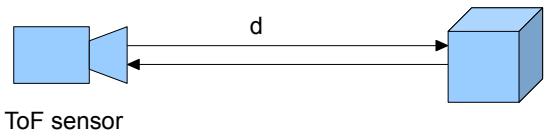


Figure 1.1: Time of Flight working principle.

where f_c is modulation frequency and f_s is the frequency of signal created by the sensor.
The receiver can acquire the signal that is of the form

$$r(t) = R \cdot s(t - \tau) + B + w(t) \tag{1.3}$$

where R is the amplitude of the received signal, τ is the transmission delay, B is the illumination cover and in the end $w(t)$ contains the error components, that affect the overall transmission.

1.1.2 Error components for ToF processes

It is important to know the maximum number of error causes, to deeply understand the behaviours of devices based upon ToF technology.
For every point, the estimation of the depth $\hat{z}$ can be considered as the sum of different components [11]

$$\hat{z} = z + \Delta z + w(t) \tag{1.4}$$

- Δz is the systematic component, which is time-invariant.
- $w(t)$ is the random white gaussian noise component, which is time-variant.

The random noise component $w(t)$ is mainly due to the sensor accuracy and intrinsic noise that affects every measurements process.
The systematic component Δz is due for a lot of factors, that can be divided in two major categories: internal and external factors. Internal factors are correlated to the calibration of the device, and they strictly depend on the hardware and firmware implementation of the sensor, and it is not simple to create a model for this kind of error for this reason. Instead, external factors are recurrent in measurement processes. They are principally environmental factors that modify the depth values returned from the sensor. Following factors can be considered as technology-dependent, in fact different devices based upon ToF principle are affected from following problems:

- amplitude sensitivity
- scattering phenomenon
- mixed pixel
- external temperature
- warm-up effects

Amplitude sensitivity

The amplitude value of received signal $r(t)$ is function of different agents. Several experiments [14, 11] show how this value is highly susceptible.
First of all near infra-red component of environment light radiation (previously called B) cannot be distinguished by the sensor, which retrieve its value as an additive component in the received signal $r(t)$.
Secondly the reflectance capability of the objects in the scene can affect measurement consistency. In fact in case of high reflectance surfaces, saturation phenomenon can occur, precluding the possibility of the sensor to retrieve a good measurement.
Again, the integration time plays a role, too. In fact too high integration time can lead to fully saturated depth-maps, and on the contrary too low integration time can prevent in some cases a good depth-map estimation.

Scattering phenomenon

The scattering phenomenon is a limiting factor for actual ToF devices, because the hypothesis that every single pixel only receives optical energy only from the are of the space that is related to, is never verified at all.
For this reflectance phenomenon, every pixel receives in reality the contribute of various contiguous pixels. The first undesired effect is that in correspondence of depth discontinuity, or simply in correspondence of high reflectance surfaces (i.e. white walls), the depth can assume values distributed in a weighted mean of all the contributes. For example it happens frequently that the squared rooms can assume a more curved appearance in correspondence of the corners for the multiple reflections path that are captured from the receiver. This phenomenon can be viewed in Figure 1.2.

Mixed pixel

They are simply pixel for which the depth value is affected by ambiguity value assignation. Measurements are subject to a so called back-folding phenomenon that is due to the periodicity of the signal that is used for the distance measurement. If the objects could be present in the scene at distances which differ by more than the distance D corresponding to a full

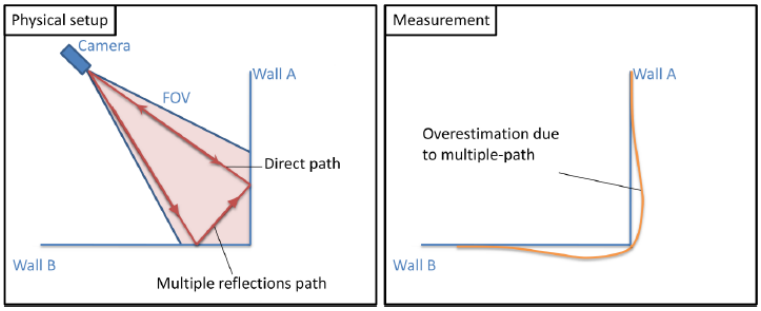


Figure 1.2: Multiple reflections phenomenon on correspondance of walls.

modulation period, the measurement of their position is ambiguous: it could be at x or at $x + D$, or even $x + 2D$, etcetera.
This problem is often solved filtering the received signal for which reflecting object are farther than the non-ambiguity range, in fact an higher attenuation phenomenon on these reflected signals will occur and so lower amplitude is acquired.
A simple explanation can be viewed on Figure 1.3.

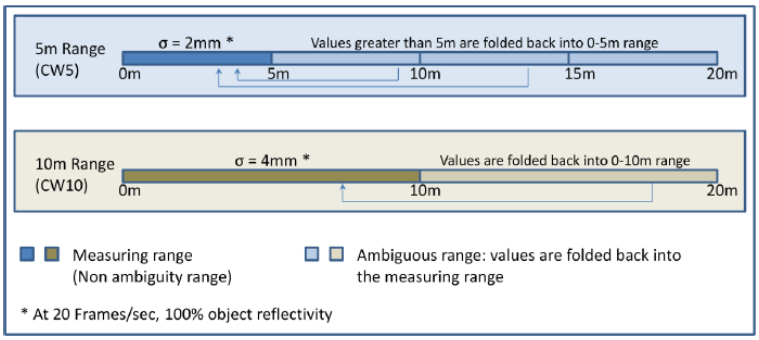


Figure 1.3: Backfolding phenomenon.

External temperature

Another factor that is responsible for measurement error is the environmental temperature. Often the producers of range cameras declare the best working temperature value, and design a built-in cooling system which can in some cases misrepresents the depth values retrieved.

Warm-up effects

Some devices can present warm-up effects, that can be simply described as different behaviours as function of the moment of utilization. In fact optimal operating trend is reached after some minutes of exploitation.

1.1.3 Mesa SR4000

Mesa Swiss Ranger 4000 is the first ToF sensor analysed in this work. It is a range camera produced by Mesa Corporate [2] in 2008, and it is one of the most advanced device that implements ToF technology.
The device is constituted by 24 LEDs emitters, arranged in a square manner, that irradiate the environment with a signal $s(t)$ with wavelength λ_c 850$[nm]$, modulated with frequency f_s often set to 30$[MHz]$[1]. The receiver component is constituted by a CCD/CMOS matrix of sensors, which dimensions are 176×144. Every receiver has an area of $40 \times 40[\mu m^2]$.

At least other two components must be cited to describe quite well this

Figure 1.4: Mesa Swiss Ranger 4000 camera overview.

device:

- Optical Filter allows only light of wavelengths near that of the illumination LEDs to pass into the camera lens.
- Illumination LED cover protects the LEDs while allowing their light to be transmitted.

They are important because permit a better filtering of unwanted frequency light, improving acquisition quality.
The physical process of analogical signal acquiring is composed by a sampling process of $r(t)$ signal. SR4000 range camera executes a 4-times for every period sampling process, and from these samples it starts to calculate the phase shift $\Delta\varphi$; it is the phase shift itself that allows the camera to produce the depth-map data. The amplitude values may be used as a measure

[1] This value can be easily tuned via software to change the maximum distance value that can be identified by SR4k sensor.

of quality of the distance measurement, or to generate a grayscale image of the scene.
From a software point of view, the first step after the analogical, physical acquisition is the A/D conversion. The analog electrical signals are converted into digital values in a conversion process, from which a 16-bit distance is calculated. This is the *raw* output of the camera, with the full-phase value of $0xFFFF$ corresponding to a distance of $5[m]$, for a modulation frequency value of $30[MHz]$. The camera produces also a 16-bit digital amplitude signal.

1.1.4 Software Toolkit for SR4000

With the official supplied kit of Mesa (`SR_3D_view.exe`), it is possible to handle different kinds of datas.

- Distance image: for each pixel ($176 \times 144 = 25344$) this image contains the depth value of the corresponding object. The distance is expressed in meters in a 16 bits spanned range, which spaces from 0 to 5 meters.
- Amplitude image: for each pixel ($176 \times 144 = 25344$) this image contains the amplitude value of the corresponding object, which is another time an array of 16 bits words. It can report the happening of saturation phenomenon for each pixel, and in this case the MSB is set to be 1.
- Confidence image: it is generated in the driver of the host PC using a combination of Distance and Amplitude measurements and their temporal variations. It represents a measure of probability that the distance measurement for each pixel is correct. Low confidence is typically due to low reflected signal or movement in the scene. The Confidence Map can be used to select regions containing measurements of high quality, reject measurements of low quality, or even to obtain a confidence measure for a measurement derived from a combination of many pixels. Confidence Map data is output from the SR4000 as an array of 16 bit words, of length (176 * 144 = 25344). The Confidence Map has a range of 0-0xFFFF, with greater values representing higher confidence.

For further considerations and acquisitions, it have been used a software (`ToolCalibrazione.exe` [14]) that exported data values in a handy format. For each acquired frame we have (`xxxx` stands for the incremental number of the measurements):

- `sr_a_xxxx.png` amplitude PNG image.
- `sr_c_xxxx.png` confidence PNG image.

- `sr_d_xxxx.txt` depth matrix in a simple text file; it contains the 176×144 depth values in a floating point format.

1.1.5 Canesta Range Camera

Another ToF sensor analysed in this work is the Canesta range camera. This sensor has been developed by Canesta Inc. (now acquired by Microsoft) in 2010. It presents some improvements against SR4k device.
It is a newer sensor, which presents first of all a RGB built-in camera. Its resolution is standard VGA, so it returns also a color information of the scene of 640×480 pixels. For the depth measurement technology side, it is very similar to Mesa SR4000 for a lot of aspects, in the sense that uses the same working principle, measuring the time of flight of a modulated sinusoidal signal. A very important improved feature is the dimension of the depth map, that is increased to 320×200. Unfortunately more deep informations around the implementation of this device are not available because of it is covered by trade secret.

1.1.6 Software Toolkit for Canesta camera

With the official supplied kit of Canesta (`SALSample.exe`), it is possible to handle different kind of data, very similar to the data handled by SR4k. The main difference is the possibility to directly export also RGB images of the frame. For canonical geometrical informations, the output format is the PGM format, while for color images the software uses PPM format. These formats are used because of they have some technical advantages: for example they are very suitable for multiple acquisitions, because even more images can share the same header, using less disk space because they are placed consecutively into the same files. Even if this solution uses less disk space, for several acquisitions it is an important issue, that can in some cases compromise buffer integrity of acquisition software, if some errors occur.
For further considerations the saved data of Canesta camera have been processed by a Matlab script to adapt the informations in a handy format. Also in this case for every acquired frame we have (`xxxx` stands for the incremental number of the measurements):

- `ca_a_xxxx.png` amplitude PNG image.
- `ca_c_xxxx.png` confidence PNG image.
- `ca_d_xxxx.txt` depth matrix in a simple text file; it contains the 320×200 depth values.

1.2 Structured Light Sensor

1.2.1 Structured Light technology

Other techniques frequently used to retrieve shape informations of the environment consider the shading on a surface as an important source of information about local surface orientation. When a surface is covered by a projection of light source, the surface normal changes across the object and the apparent brightness changes as a function of the angle between the local surface orientation and the incident illumination.
Upon this basic principle, structured light technology permits the shape estimation of an object[16].

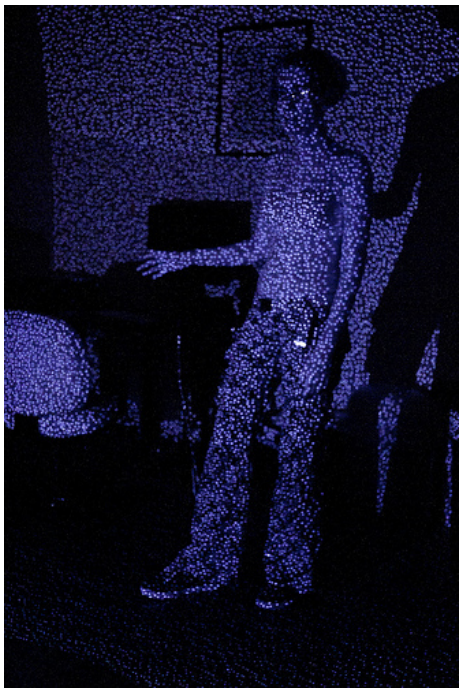


Figure 1.5: Structured light projection with dots.

1.2.2 Microsoft Kinect sensor

The Microsoft Kinect is a device released in late 2010 by Microsoft, designed for the gaming console Microsoft Xbox. Its main purpose is obviously videoludic, and it is a controller-free gaming and entertainment experience[3]. The innovative aspect of this controller is the fact that enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface of gestures and spoken commands. In fact its software technology enables advanced gesture recognition, facial

Figure 1.6: The Microsoft Kinect device.

recognition and voice recognition.
It contains inside a variety of sensors which permit the use of such natural interfaces. It is constituted by a little mechanic motor that enables some movements of the sensor and an array of microphones to capture sounds from the surrounding environment, but the most important components are a VGA camera with 640×480 resolution and a 3D depth sensor that produce at the output a post-processed depth map with 640×480 resolution[2].
The last one is the sensor on which the analysis has been mainly focused. Before starting to describe this interesting part of the Kinect, a little preamble must be done. It is a device developed with a proprietary paradigm, for which no detailed description is officially provided and even though a lot of reverse engineering has been done by some fans and researchers, more detailed technical information are not available. It is known that the depth sensor has been acquired by Microsoft from PrimeSense Ltd., a company skilled in natural-interface interaction between human and computer.
It is PrimeSense itself that publishes a little description about the component which enables 3D capabilities, asserting that it is based upon Light Coding^{TM}technology[4]. This technology is a 3D structured light sensor, constituted by two main component, well distinguishable between them; they are an IR emitter and an IR CMOS camera. The emitter projects a static pattern on the scene, constituted by a constellation of dots, which are captured by the IR camera that decode the constellation to retrieve a complete depth-map of the scene. The decoding step executes a sophisticated parallel computational algorithm to decipher the informations about the scene.
A very similar technology is described in another PrimeSense patent [15], and it can be used as a reference, while the real Light Coding patent is pending.
A fundamental characteristic of the sensor is that it performs by construction a post-processing step on raw-data acquisition. The Microsoft Kinect doesn't produce a dense depth map, but the output is relatively sparse, and for every depth measurement we have more than only one pixel. For this reason the 640×480 resolution is obtained with a particular interpolation

[2]More details about post-processing operation of the Kinect are explained soon and so the Kinect has not a native 640×480 resolution.

of the pixel of the IR image acquired by the CMOS camera. The algorithm of the interpolation procedure is unknown, for the already cited reasons of trade secret covering this new technology.
It is important to underline also that a formal error model has not been developed yet for the Microsoft Kinect 3D depth sensor.

1.2.3 Software Toolkit for Microsoft Kinect

To retrieve informations on the scene using the Microsoft Kinect device, a very user-friendly tool has been used. It is `RGBDemo` [5], which looks very similar to other tools used in case of ToF acquisitions. The version used for further analysis is 0.4.0, which supports a complete retrieving of following informations through the use of `libfreenect`[1]:

- Depth-map, which is available in two different format, a grayscale PNG image and a YML text version, with a resolution of 640×480.
- Intensity, available in grayscale PNG image format
- Color image, available in RGB color PNG image format.

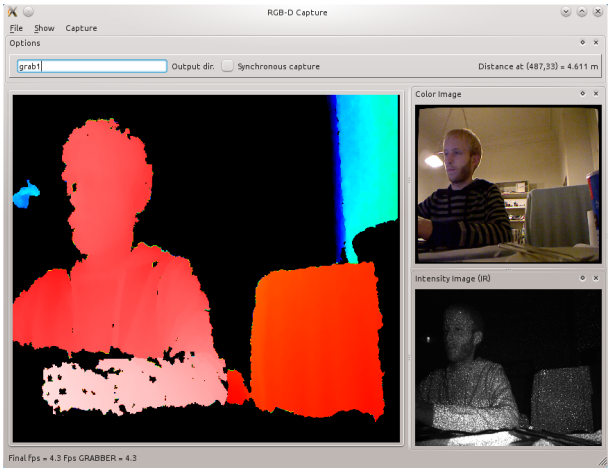


Figure 1.7: Main window for RGBDemo acquisition tool.

This tool supports a fast grabbing of multiple frames, but it has some issues[3] for much longer acquisition, stopping the process in a random way. This problem lightly limited the number of acquisitions that have been performed for further analysis.

[3]The issues were encountered in version 0.4.0. Further versions has not been tested.

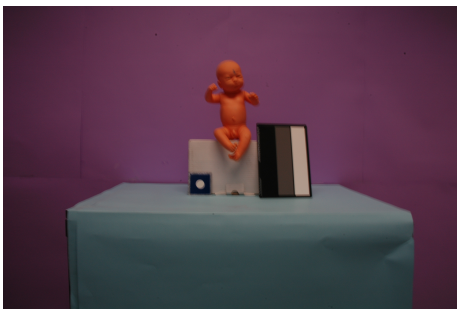
Chapter 2

Comparison between 3D Sensors

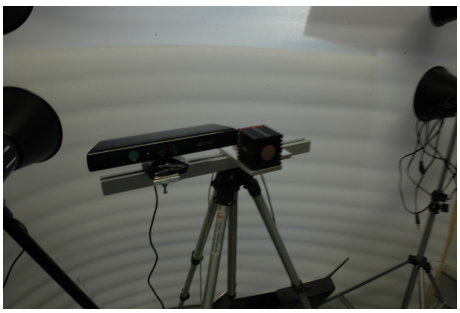
To get more knowledge on the behaviour of the sensors, it is necessary to achieve some data. After some preliminary considerations, a more deep and rigorous analysis will be performed and in order to qualitatively and quantitatively analyse all three devices, a set of experiments have been designed with the purpose to characterize all the features of ToF cameras and of Microsoft Kinect.
Before starting to present some measurements' results, it is important to declare the costs of the different devices considered. For ToF sensors both the hardware implementation and the state-of-the-art of the technology itself lead to highly increase the realization costs. Mesa SR4k has a price of about €7000, even if it is produced to be sold on the market. For the Canesta camera another type of observation must be done. In fact it is the result of a research prototype, for which a real price doesn't exist since it is not a product ready to the market. Even with previous statement, it is simple to understand the magnitude of the price on the basis of the SR4k one.
In the case of the Microsoft Kinect, previous considerations must be drastically changed. It is a device produced by one of the most important corporation of the IT world, and so any other further statement must take into account this fact. Moreover the target of this product is the mass market of video-games, and the price of the Kinect must respect some business constraints. The actual price for italian market is about €150.
Already with the last statement it is possible to understand how ToF devices and Kinect belong to different categories. This fact must be taken into account to make final conclusions.

2.1 Preliminary Acquisition

Upon the basis of the introduction made in Chapter 1, it is possible to prepare an experimental setup to highlight the characteristics of the sensors and consequently perform a deeper analysis.
A simple setup has been prepared and it is presented in Figure 2.1(a).



(a) Acquisition setup.



(b) Sensors used: Microsoft Kinect and Mesa SR4k.

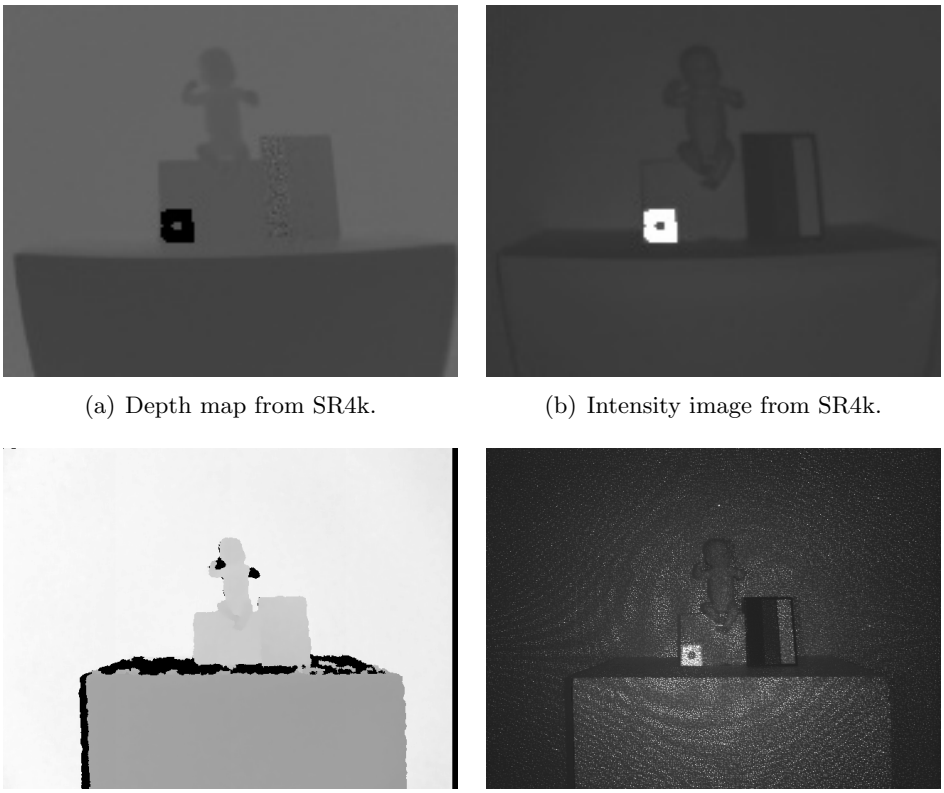
Figure 2.1: Preliminary setup.

This scene has been considered in order to qualitatively analyse different behaviours on different sensor, the ToF technology based one and on the other side the structured light technology-based one.
As you can see it contains several interesting elements that can stress the performances of both the cameras. These objects are:

- a gray-scale board with patches that have different reflectivity property
- a photogrammetry marker, characterized by an high reflectance in the near IR
- a baby doll which represents an articulated shape object in the scene
- a table with a very large slanted surface
- a wall on the background with a large flat surface

In Figure 2.2 we can see the different acquisitions with both ToF, i.e. Mesa SR4k, and structured light sensor, i.e. Microsoft Kinect (see Figure 2.1(b)).

It becomes more clear looking to Figure 2.2, how the two technology work in a different manner. In Figure 2.2(b) and 2.2(d) there are the intensity images of the two sensors, Mesa SR4k and Kinect ones respectively. The latter one is characterized by the constellation of dots projected by the IR emitter, mentioned in Chapter 1, while the intensity of Mesa SR4k doesn't show this kind of structure. Continuing with a preliminary analysis of the



(a) Depth map from SR4k.

(b) Intensity image from SR4k.

(c) Depth map from Kinect.

(d) Intensity image from Kinect.

Figure 2.2: Acquisitions on setup in Figure 2.1

intensity and depth images, some more aspects must be highlighted and after a short presentation of such phenomenons, in the next sections a more deep analysis will be done.

Grayscale Board

The gray-scale board positioned on the table is made by some different patches. In the case of SR4k it is highly visible on the Figure 2.2(a) the presence of a lot of Gaussian noise on the depth-map estimation. For the sensor of the Kinect, there are not issues on this object, and the depth estimation seems being lightly affected by any kind of error.

Photogrammetry Marker

The property of the photogrammetry marker causes a high saturation in depth estimation of the ToF, and it is characterized by a black depth color (that means no depth estimation) in Figure 2.2(a). Even Kinect seems to detect the high reflectance itself, in fact the intensity image reports the

acquired dots with an higher brightness in the zone of such marker, but in this case the depth estimation is not affected by any kind of artefact and the relative zone contain consistent value of depth.

Edge Identification

To correctly control the aspect of edge identification, it is necessary to zoom an image and make an overlapping between the depth and the intensity image. Figure 2.3 shows clearly this phenomenon. The blue shading is the

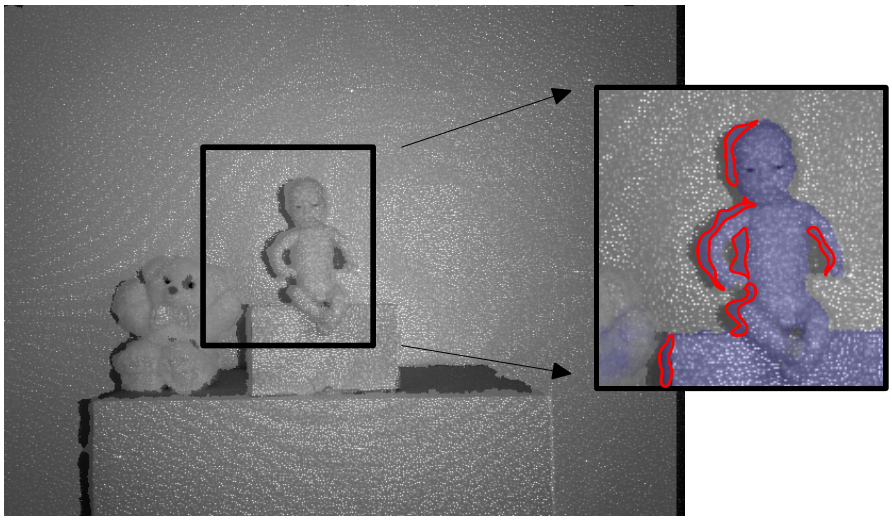


Figure 2.3: Depth map misalignement with Microsoft Kinect with respect to the acquired IR image.

shape in the depth image, which is overlapped onto the IR image. As it is highlighted by the red lines, there is an annoying misalignement phenomenon. In some cases it rise up to 10 pixels of dimensions, and it must be taken into account on further analysis and data processing.
The phenomenon occurs only with the Kinect sensor, while ToF cameras are not affected in general by this systematic error.

Shape Estimation

Figure 2.2 is characterized by a lot of complex shape, that are positioned very near in the space. While Mesa SR4k produce a regular depth map also in presence of articulated shapes, Kinect depth-map is affected by some holes, where the real values of depth are not retrieved. In particular in Figure 2.2(c), the baby arms are not retrieved at all. This consideration makes the Kinect very error-prone in such environmental conditions.

Depth–map Production

Continuing with the analysis of the depth-map appearance, some other artefacts occur in image. For the SR4k side, the main problem is represented by saturated points, in fact on correspondence of the marker, there are not values of depth. For the Kinect side another problem occurs in correspondence of the slanted surface of the table, and here there aren't available depth values. It is curious to highlight that where the ToF camera doesn't perform very well, the Kinect doesn't show any issue, and vice versa where the Kinect has problems of depth retrieving, the ToF produces a low noise depth map.

Occluded Points

The last aspect of this preliminary analysis is the occlusion of some part of the scene. While SR4k is not affected by this issue, with Kinect it occurs all but in every acquisition. In Figure 2.2(c) the phenomenon occurs on the left of the table.

2.1.1 Time–of–Flight

There are multiple causes that influence the measurement of ToF sensors.
The reflectance of the scene at the emitters wavelength can modify the value of the measure. If the surface is highly reflective (e.g. white coloured surface), the depth measures are more precise. On the contrary if the surface is poorly reflective (e.g. black coloured surface) the depth measures are less precise, and in particular they are inclined to retrieve a farther object.
The worst case of the reflective phenomenon is a high reflectance surface, (e.g. the photogrammetry marker in Figure 2.1(a)) that bring on the occur of the saturation phenomenon, and as stated in previous section, it leads to absence of depth information.
The background illumination B is also another important factor of error increasing in a measurement. For example, the sun radiation has of course a non-neglectable component in the near IR (e.g. $830[nm]$). The higher is the value of this background illumination, the higher value of component B, and so the less precise is the ToF measurement.
The geometry of the scene can also affect the measurement in a particular way that is explained here. For discontinuity in the shapes of the scene that are covered by a single pixel, e.g. part of this area may be relative to a closer object and another part to a farther object. The effect encountered is that the measured distance is somehow in between the distance of the farther and closer objects. The pixels affected by this phenomenon might be referred to edge pixels.

For the non-edge pixels it is possible to perform a canonical Signal-to-Noise-Ratio (SNR) analysis of the measurement errors. The general distribution of their measurement noise is approximately Gaussian, as described in [14] and as it is shown in following section. The standard distribution of such a Gaussian is directly proportional to the SNR of the considered pixel. Within $r(t)$, the useful signal component is $R \cdot s(t - \tau)$, and the noise component is $B + w(t)$. In order to estimate the values of the SNR the following considerations should be taken into account.
The higher is the reflectivity of the area the higher is the amplitude of the received signal R and the SNR (e.g. white surface has a high SNR and a low standard deviation of the Gaussian, while black surface has lower SNR and the Gaussian has a higher standard deviation). The further is the object from the sensor, the lower is R, and so the lower is the SNR. The higher is the background illumination, the higher is B, and the lower is the SNR.
The SNR analysis of the edge pixels is more complex because of multiple reflections that might occur. However it is interesting to notice that the measured distance can still be approximated by a Gaussian, with mean somehow in between the further and the closer surface parts and standard deviation with unknown characteristics. The mean location depends on the proportion between the area of the closer and the further surface.
Last but not the least, there is another effect that generally affects the measurements performed by a ToF range camera, i.e., a systematic offset in the distance measurement due to harmonic distortion [13]. However, this issue is internally corrected by the manufacturers, as in the case of the Mesa SR4k, and therefore it is not considered in this analysis.

2.1.2 Microsoft Kinect

For the Microsoft Kinect it is not possible to make the same consideration that have been stated before for ToF cameras, because a formal error model has not been developed yet. However it is possible to describe its various sources of errors and artefacts.
With respect to the ToF cameras analysis of the previous section, the distinction between edge and non-edge pixels for the Kinect camera needs to be slightly modified. For the previous highlighted phenomenon of edge misalignment, in these zones of the depth-map, the systematic error of depth misalignment can rise up to huge levels.
Moreover some random errors in the measurement of the depth values of all the pixel may occur, but the distribution of the measurements is not a Gaussian anymore as the distribution of the ToF measurements. The distribution of the error for such a depth sensor is spiky, for both the edge or the non-edge pixel. Probably this randomness is due to the noise of the IR CMOS camera and to some reflections artefacts that do not allow to the camera to correctly retrieve the reflected dots of the scene, or alternatively

by some internal post-processing calculation of the firmware of the Kinect made upon the raw data of the depth-map.
Other depth artefacts typical of the Kinect are the lack of measurements for a considerable amount of pixels. This is due by several causes. The first one is principally the spatial dislocation of the IR emitter with respect to the IR receiver, that is rather large, countable in about $7.5[cm]$. The generated effect is the occlusion of some parts of the scene, and the consequent absence of informations. Another cause can be considered in the acquisition of slanted surfaces, on which the prospective distortion affects too much the shape of the projection pattern, making impossible for the Kinect estimation algorithm to correctly interpret it. In the end also a very low reflective surface can produce a complete absorption of the light emitted by the projector causing a lack of depth measurements.

2.2 Quantitative Analysis

In the previous sections it has been presented a preliminary analysis of the behaviours of the camera that are based upon different working principle. In particular to test the data retrieved by Mesa SR4k and by the Microsoft Kinect a setup has been prepared.
Now a more methodical and rigorous analysis is done on all three devices presented in Chapter 1. In order to perform it, a specific template object has been designed and built with some advantageous characteristics. It is showed in Figure 2.4.

Figure 2.4: Template object.

This object has been designed with specific intentions. It is constituted by a flat panel, half is black-coloured and half is white-coloured. For everyone of those zones, there are two protruding cubes, one black and and the other white coloured. Figure 2.5 shows the specifications of such panel. This particular configuration has been chosen for the analysis of the acquisition errors in several situation:

1. Distribution of the measurements noise on flat surfaces as function of the target reflectivity (white and black surface).

2. Distribution of the measurements noise on surfaces characterized by depth discontinuities in every combination of the target reflectivity.

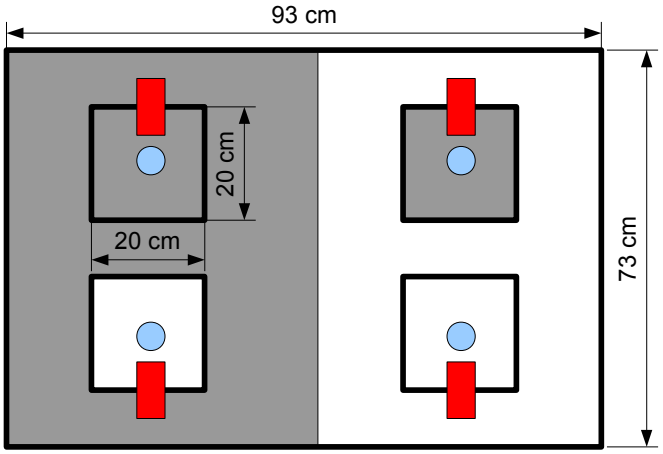


Figure 2.5: Template object specification. Cyan circles are single points, red rectangles represents edges points stripes

In order to quantify the first type of noise, i.e. the noise on flat surfaces as function of flat surface reflectivity, four points on the scene have been considered. Such points (shown by cyan circles in Figure 2.5) have been picked in the middle of the four cubes and represent the following situations:

1. $PointWW$: high reflectivity foreground (white cube) on high reflectivity background (white panel).

2. $PointWB$: high reflectivity foreground (white cube) on low reflectivity background (black panel).

3. $PointBW$: low reflectivity foreground (black cube) on high reflectivity background (white panel).

4. $PointBB$: low reflectivity foreground (black cube) on low reflectivity background (black panel).

On the contrary, in order to analyse the distribution of second type of noise, i.e., the noise on depth discontinuities, another set of points of the template has been considered. For each of the four cubes, a segment of five points that crosses a depth discontinuity, has been considered (the corresponding region is shown in red in Figure 2.5, while the points position with respect to the edge is represented in Figure 2.6. Such stripes of points reflects four possible situations[1]:

1. $EdgeWW_i$: high reflectivity foreground (white cube) on high reflectivity background (white panel).
2. $EdgeWB_i$: high reflectivity foreground (white cube) on low reflectivity background (black panel).
3. $EdgeBW_i$: low reflectivity foreground (black cube) on high reflectivity background (white panel).
4. $EdgeBB_i$: low reflectivity foreground (black cube) on low reflectivity background (black panel).

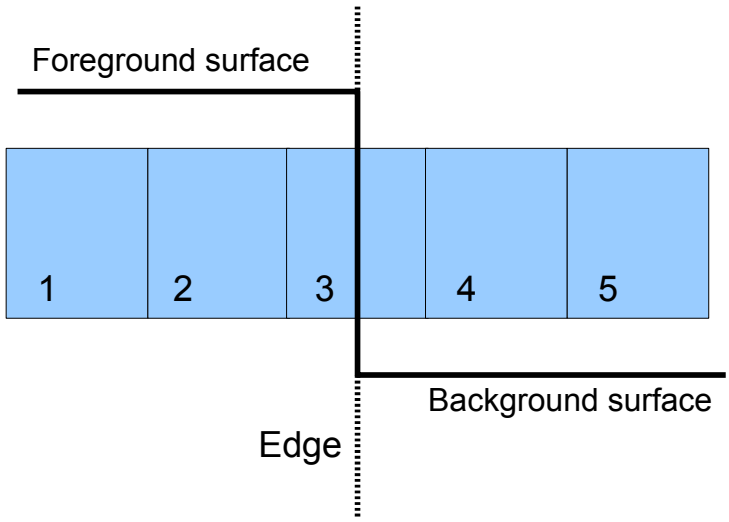


Figure 2.6: Alignment of the considered segment of pixels, along a depth discontinuity.

For each of these points a set of 10000 depth acquisitions have been performed with the Mesa SR4k and with the Canesta Time-of-Flight camera. A set of 1500 depth acquisitions has been performed with the Microsoft

[1]The symbol i can assume a value in the integer interval $[1, 5]$ as shown in Figure 2.6.

Kinect[2]. In all the acquisition the panel was at a distance of about 2 meters in a controlled environment without external illumination. This was made in order to avoid multiple reflection phenomenons, and to obtain a repeatable configuration. Nevertheless the distance from the set of acquisitions made with different sensors can change of about 10-20 cm, but this fact doesn't affect the exactness of the analysis which is rather focused on the distribution of the measurements, instead of absolute precision.
For each pixel several useful calculations have been done. Furthermore they have been calculated and drawn a histogram for every pixel, to highlight their distribution, and also they have been calculated the maximum value, the minimum value, the mean and the standard deviation with the well-known formulas

$$\mu_p = \frac{\sum_{i=1}^{N} z_{p_i}}{N} \tag{2.1}$$

$$\sigma_p = \sqrt{\frac{\sum_{i=1}^{N} (z_{p_i} - \mu_p)^2}{N-1}} \tag{2.2}$$

where z_p stands for depth value of pixel p considered (e.g. $PointWW$ or $EdgeBW_3$) and N is the maximum number of pixel (i.e. 10000 for ToF camera and 1500 for the Microsoft Kinect).

2.2.1 Implementation of algorithm analysis

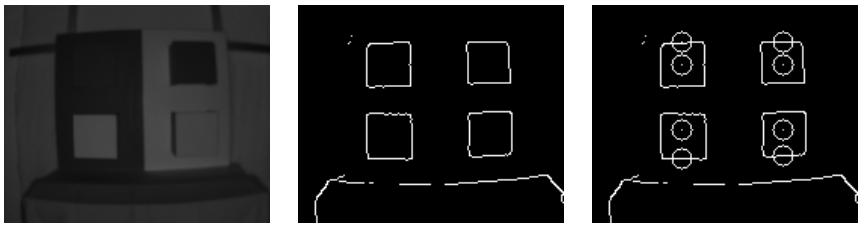
To perform the requested operations, a simple `C++` program has been implemented. It makes use of some standard `C++` libraries like `iostream`, or `string`, and the powerful `OpenCV 2.2` library [7] for the computer-vision computation part of the program.
The program is composed by the following steps:

1. Detection of the edges of the scene through the use of Canny algorithm applied on intensity image of the ToF cameras, and on depth image for the Microsoft Kinect[3]. See Figures 2.7(a), 2.8(a), and 2.9(a) that show the intensity images, while the Figures 2.7(b), 2.8(b) and 2.9(b) show the results of the edge detection.

2. Detection of the pixel considered and retrieving of their correct coordinates. The Figures 2.7(c), 2.8(c) and 2.9(c) show the considered pixels.

[2]The number of acquisition performed with the Kinect is limited because the acquisition library (i.e. RGBDemo [5]) had some problems with the acquisition of several consequent frames, making the acquisition process very annoying with random crashes of the application.

[3]The pixels considered for the Kinect were not affected by misalignment.



(a) Amplitude image from SR4k. (b) Canny computation for SR4k image. (c) Pixels detected for SR4k image.

Figure 2.7: Analysis for Mesa SR4k acquisitions. The Figures 2.7(a), 2.7(b) and 2.7(c) have all a 176×144 resolution

(a) Amplitude image from Canesta. (b) Canny computation for Canesta image. (c) Pixels detected for Canesta image.

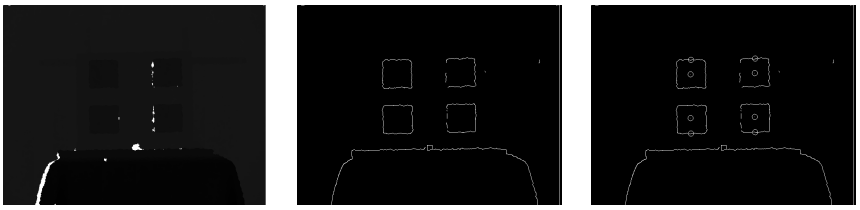
Figure 2.8: Analysis for Canesta acquisitions. The Figures 2.8(a), 2.8(b) and 2.8(c) have all a 320×200 resolution.

3. Loading of the depth data z_p, through the parsing of previously acquired data.
4. Computation of max, min, mean and standard deviation values.
5. Computation of the histograms of the depth data z_p.
6. Creation of the output of all the informations computed and acquired by the program.

Every single implementation of the procedure is slightly different on the basis of the sensor considered. This was due for the different kind of files format that store the depth values, or for the canny algorithm thresholds that had to be slightly differentiated to perform a correct edge detection in the scene.

2.2.2 Non-Edge Pixels analysis

The Figures 2.11, 2.12 and 2.13 show the distribution of measurements made by SR4k, Canesta and Kinect devices respectively, on the pixels in flat surfaces of the template object. It is highlighted by such plots, how the distribution of ToF sensors measurements are highly comparable with a Gaussian



(a) Depth image from the Kinect.

(b) Canny computation for the Kinect image.

(c) Pixels detected for the Kinect image.

Figure 2.9: Analysis for the Microsoft Kinect acquisitions. The Figures 2.9(a), 2.9(b) and 2.9(c) have all a 640×480 resolution.

distribution in every reflectance condition (e.g. black surface foreground on black background, or black surface foreground on white background).
On the other hand the Microsoft Kinect demonstrates a very strange behaviour. The distribution of its measurements is very spiky and the plots in Figure 2.13 show that the values acquired tend to agglomerate them in some sparse bins, and very far from each other. The presence of these outliers might be a problem for applications that are not robust with respect to such errors. This issue is probably due by internal processing of the Kinect software, also mentioned in previous section.
Concerning with standard deviation values, a complete comparison for the non-edge pixels is present in Figure 2.10. It is clear how all the three sensors have similar values of deviation from the mean value, even if the distribution of Kinect is not a Gaussian properly. Moreover it is worth to notice that for the ToF sensors, the black surfaces increase the deviation. This fact demonstrates how low reflectance conditions decrease the quality of a depth measurement.
The Table 2.1 reports some numerical values for measurements on flat points, in particular they are minimum, maximum, mean and standard deviation values.

2.2.3 Edge Pixels analysis

The Figures 2.16, 2.17 and 2.18 show the distribution of the measurements made by SR4k, Canesta and Kinect devices respectively in a particular edge[4]. It is clear how the two ToF sensors (i.e. Mesa SR4k and Canesta cameras) have also in this case a comparable behaviour, that is a Gaussian distribution of the measurements for all the pixels on an edge, withstanding the difficult measurement of the zones with discontinuities.
On the other hand, Microsoft Kinect repeats its very strange behaviour. The distribution of its measurements is very spiky, both in, for example,

[4]The various histograms show $EdgeBB_i$ measurements distribution.

Point p	Device	Min $[m]$	Max $[m]$	$\mu_p[m]$	$\sigma_p[m]$
$PointBB$	SR4k	1,6594	1,7006	1,6812	0,0056
	Canesta	2,0300	2,1170	2,0721	0,0116
	Kinect	1,6994	1,7173	1,7052	0,0043
$PointBW$	SR4k	1,6590	1,7097	1,6864	0,0058
	Canesta	2,0350	2,1200	2,0779	0,0122
	Kinect	1,6994	1,7173	1,7088	0,0023
$PointWB$	SR4k	1,6805	1,6987	1,6898	0,0024
	Canesta	2,0440	2,0780	2,0604	0,0045
	Kinect	1,7083	1,9426	1,7173	0,0076
$PointWW$	SR4k	1,6856	1,7061	1,6964	0,0024
	Canesta	2,0380	2,0730	2,0553	0,0049
	Kinect	1,6994	1,9544	1,7116	0,0076

Table 2.1: Values of important parameters acquired on flat surfaces.

$EdgeBB_1$, that is the pixel on the protruding flat surface, or in $EdgeBB_3$, that is exactly the edge pixel (remember the scheme in Figure 2.6). The plots show that the values acquired tend to agglomerate them also in these conditions, in some sparse and very far from each other bins. The presence of these outliers might be a problem for applications that are not robust with respect to such errors. This issue is probably due by internal processing of the Kinect software, also mentioned in previous section.
It is worth to notice also how in presence of depth discontinuities, the depths measured by SR4k and by Canesta are somehow in between the closer surface (cube) and the further surface (panel), where the distance reported changes smoothly from the points in the and the point in the panel. The depth measurements performed by the Kinect are more affected by noise, and they can present in some cases also some overshoots of about $4[cm]$ near edges, i.e. some points in the edges are measured closer that the cube or further than the panel. In Figure 2.14 they are reported the mean values of all the sensor for the pixels at the discontinuities, and all these phenomenons can be easily viewed. In particular the difficulty of Kinect to detect the correct edges is highlighted, i.e. the pixel 3 that should be exactly on the cube, sometime can be addressed with its real value of distance, to the background flat surface.
Concerning with standard deviations, the Figure 2.15 shows interesting topics. As it was expected, Microsoft Kinect presents a more noisy behaviour, in fact almost in every test case it has higher values of standard deviation. It can grow up to 20 times the standard deviation of the time of flight sensors, especially in the $EdgeWW_i$ set of points, where the brightness of the pixels

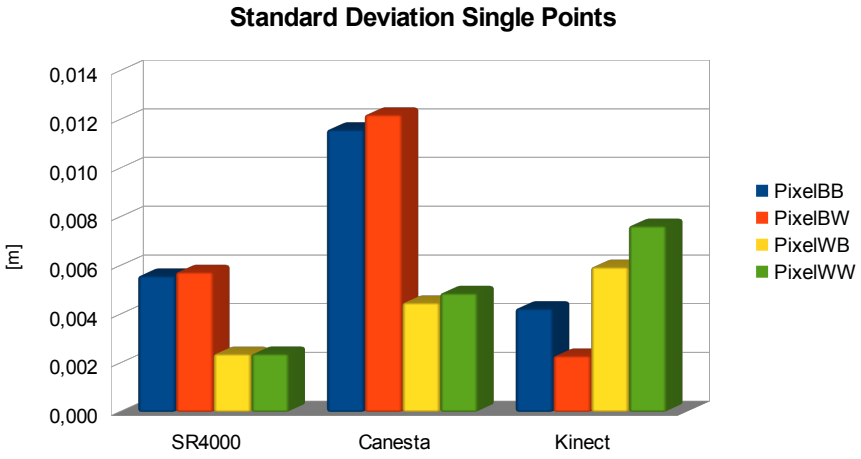


Figure 2.10: Standard deviation values of depth measures on flat surfaces.

acquired, probably[5] affects the detection process of IR CMOS camera.
For time-of-flight devices, the depth detection process is more stable and the measurements present in general lower values of standard deviation with respect of Microsoft Kinect. Both SR4k and Canesta have similar values, but in general Mesa range camera seems to be lesser affected by noise with a general lower value of deviation. This fact is mainly due for the production hardware process of Mesa and for its drivers and software that are more tested and stable: the Canesta camera is in fact a laboratory prototype and it is not fully optimized. Both ToF sensors have in every case a standard deviation value lesser of $1[cm]$ that can be considered a good value, related to a measure of about $2[m]$.
The Table 2.2 presents some numerical values for edge pixel $EdgeBB_i$. In particular they are minimum, maximum, mean and standard deviation values.

[5]It is used the term *probably* because the Microsoft Kinect decoding algorithm is completely unknown.

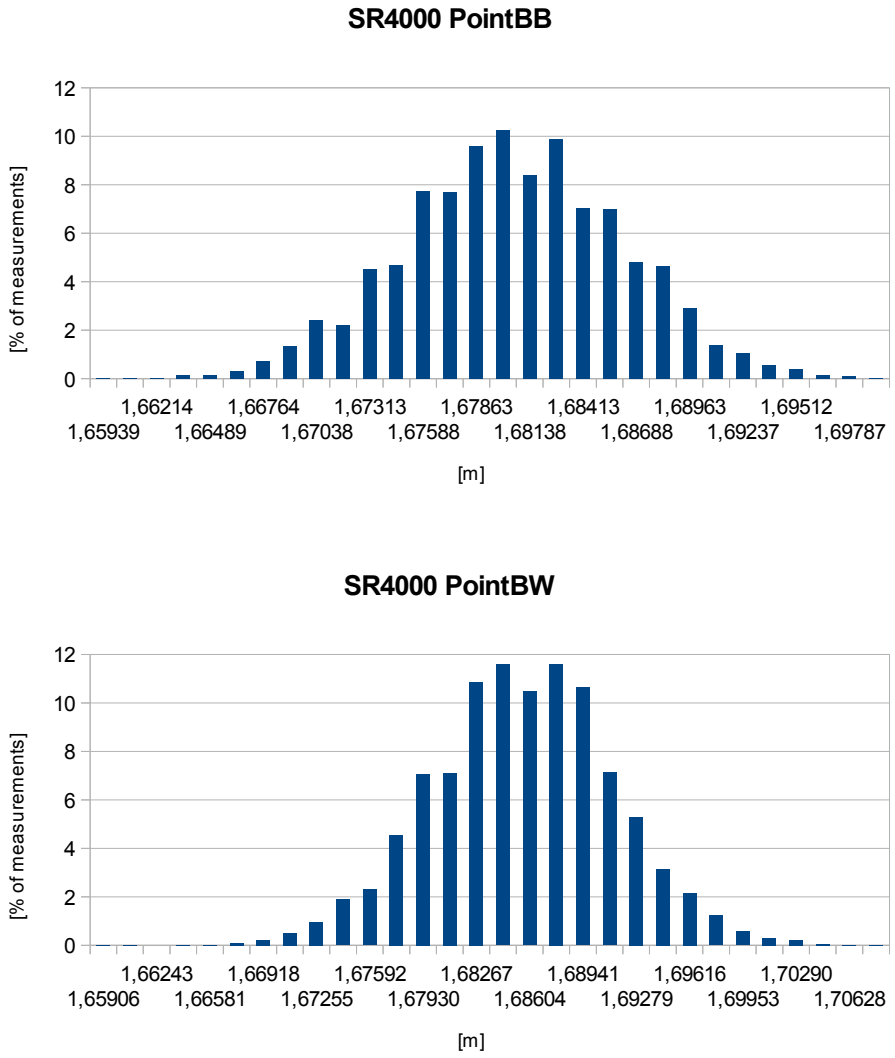


Figure 2.11: Distribution of measurements for Mesa SR4k of pixels on the flat surfaces.

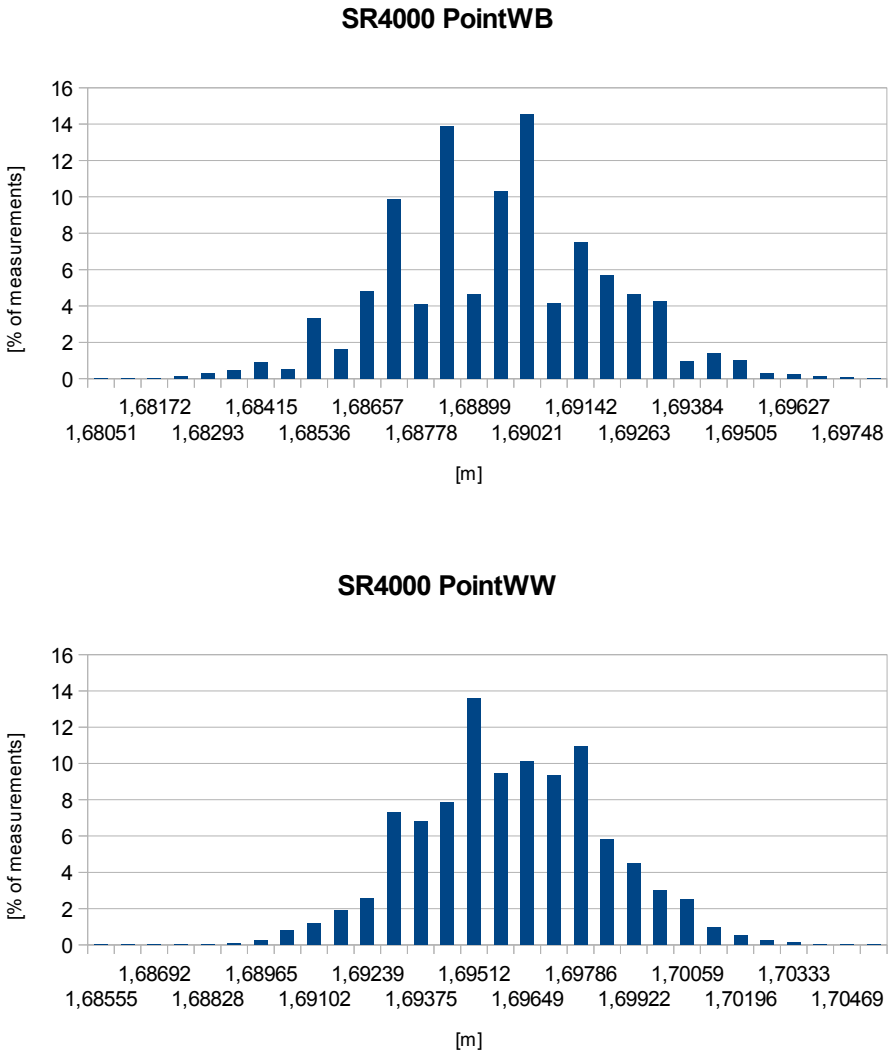


Figure 2.11: Distribution of measurements for Mesa SR4k of pixels on the flat surfaces.

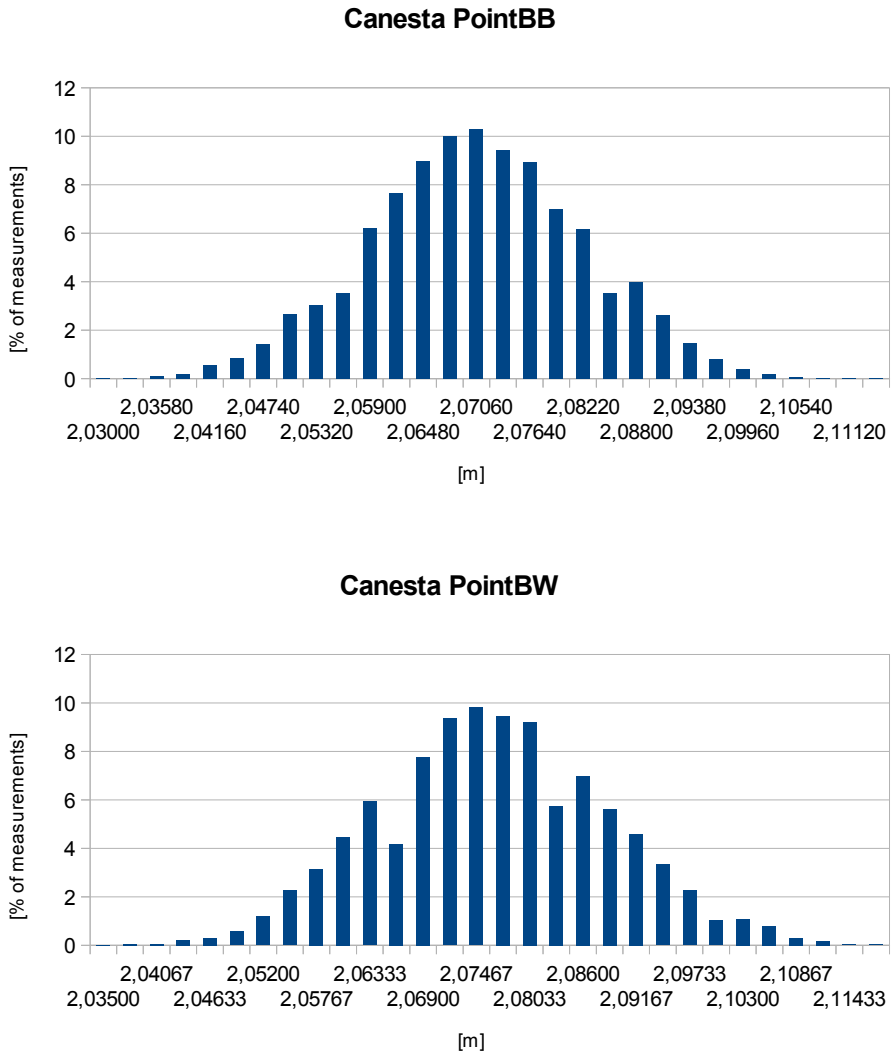


Figure 2.12: Distribution of measurements for Canesta range camera of pixels on the flat surfaces.

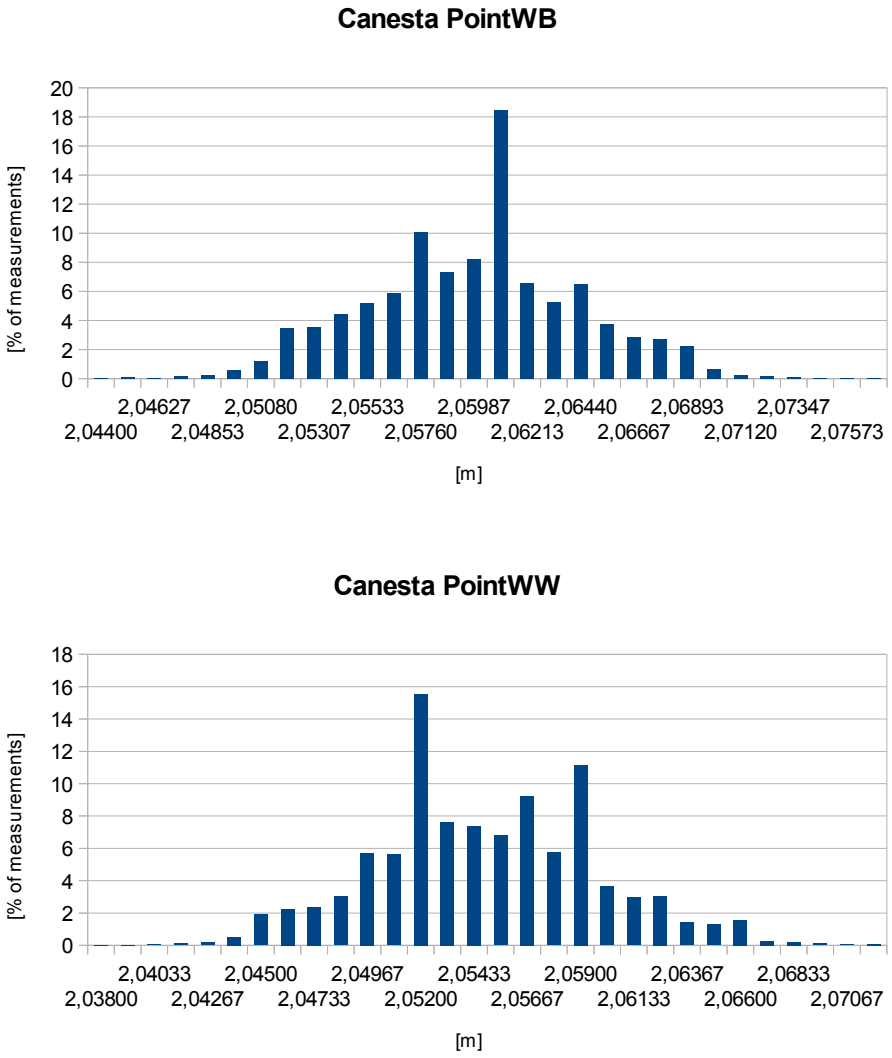


Figure 2.12: Distribution of measurements for Canesta range camera of pixels on the flat surfaces.

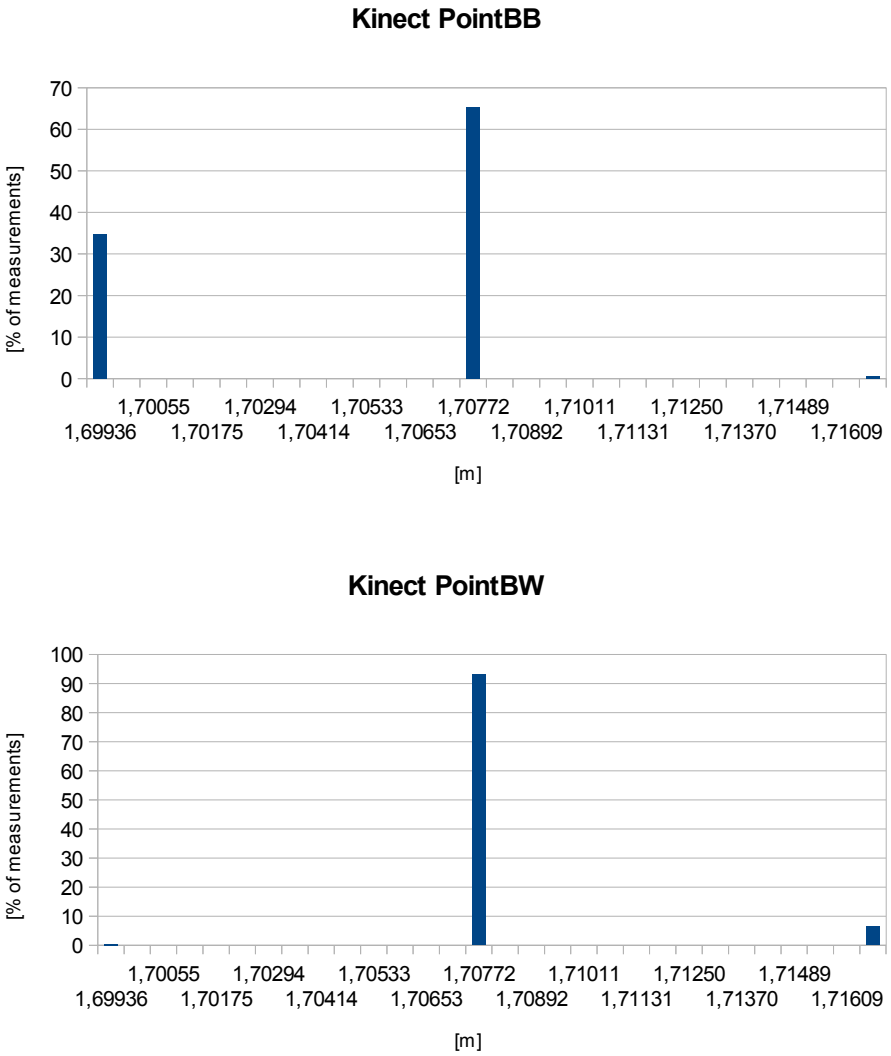


Figure 2.13: Distribution of measurements for Microsoft Kinect sensor of pixels on the flat surfaces.

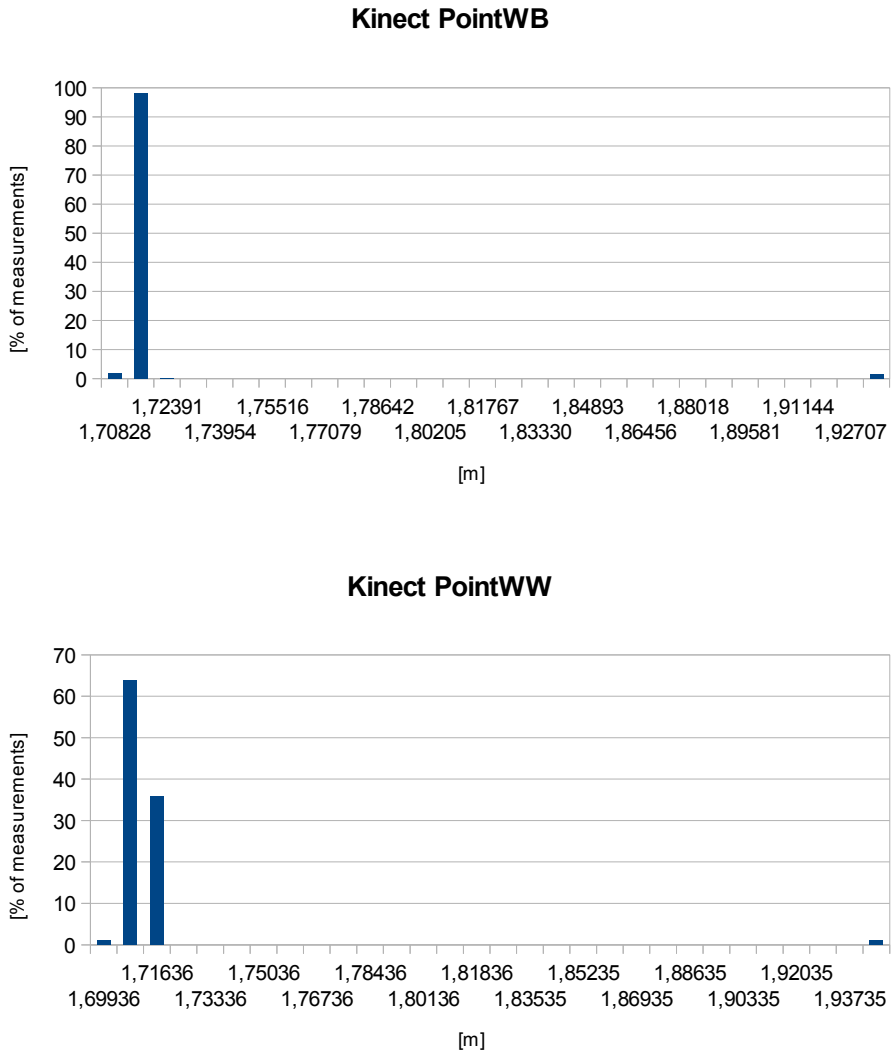


Figure 2.13: Distribution of measurements for Microsoft Kinect sensor of pixels on the flat surfaces.

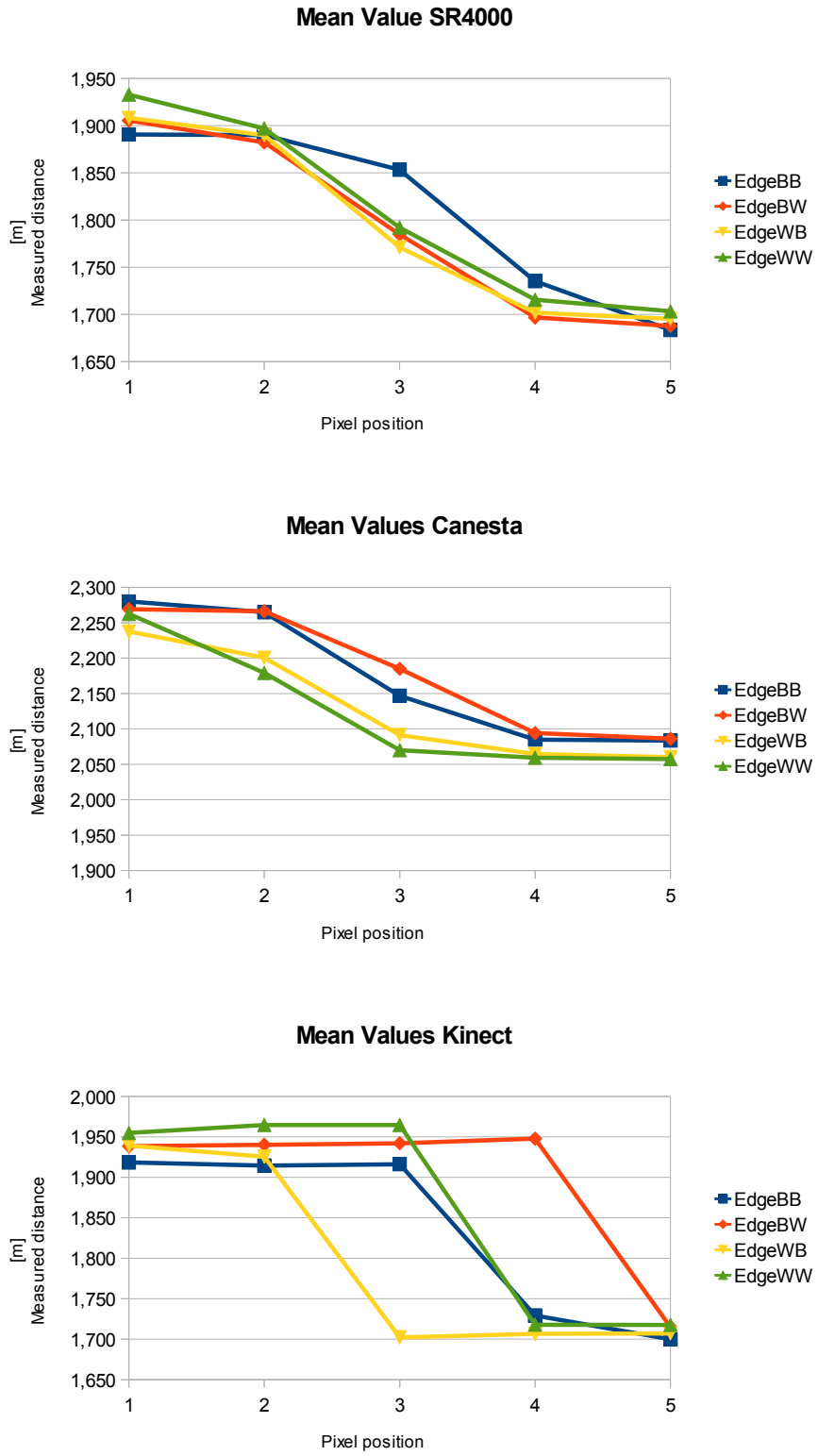


Figure 2.14: Means of depth measures in correspondence of an edge for the three sensors.

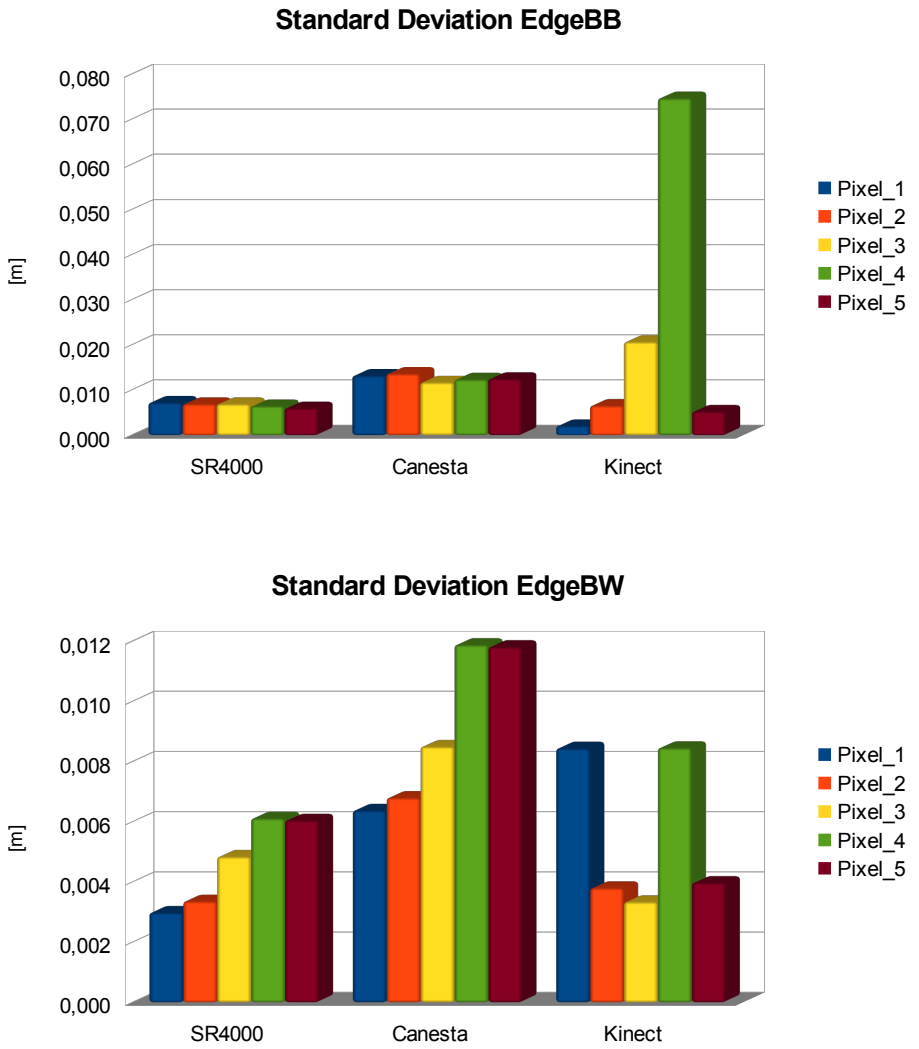


Figure 2.15: Standard deviation values of depth measures in correspondence an edge for the three sensors.

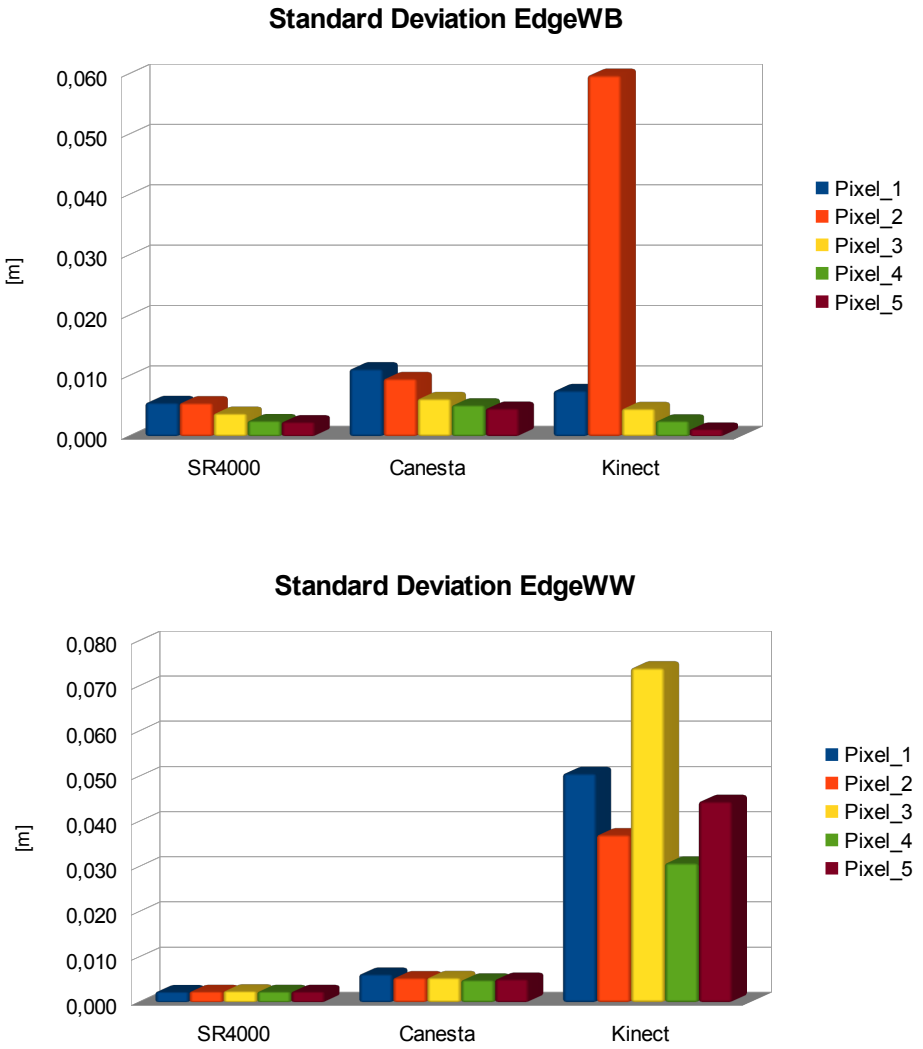
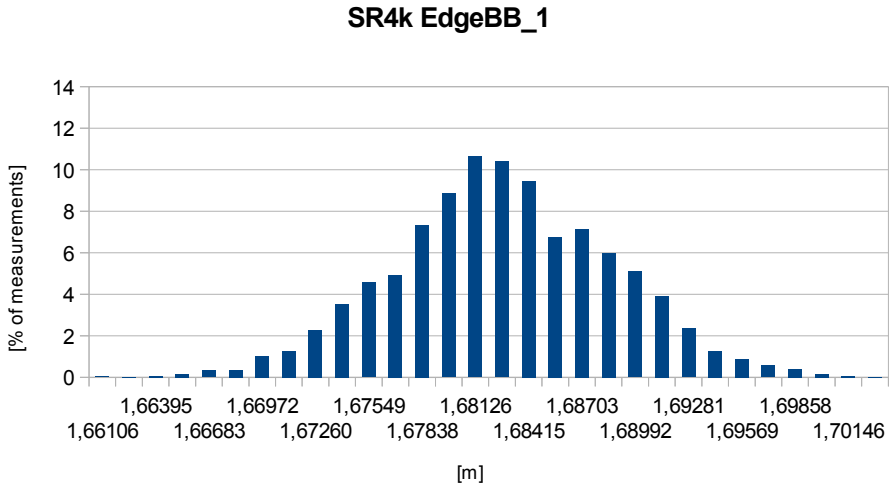
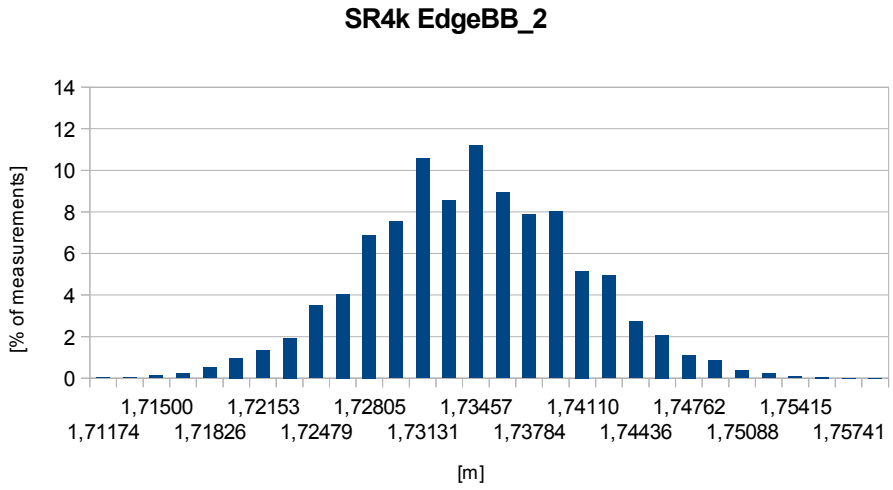


Figure 2.15: Standard deviation values of depth measures in correspondence an edge for the three sensors.





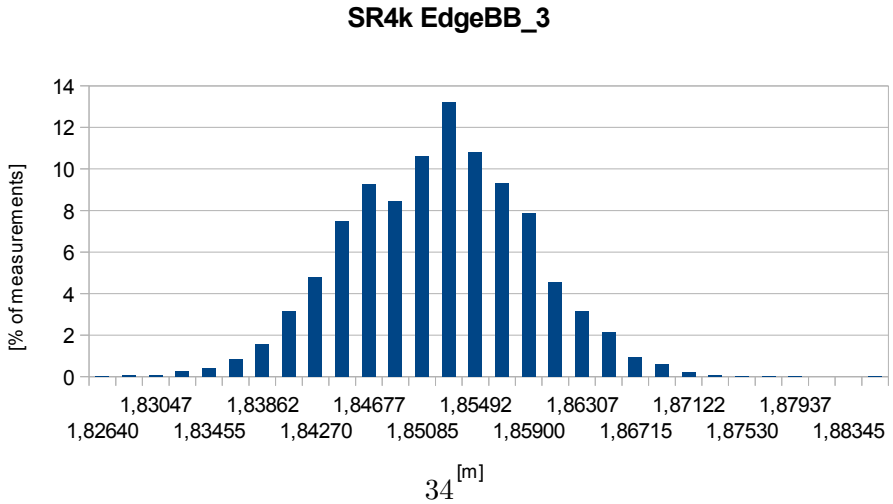


Figure 2.16: Distribution of measurements for Mesa SR4k of $EdgeBB_i$ pixels.

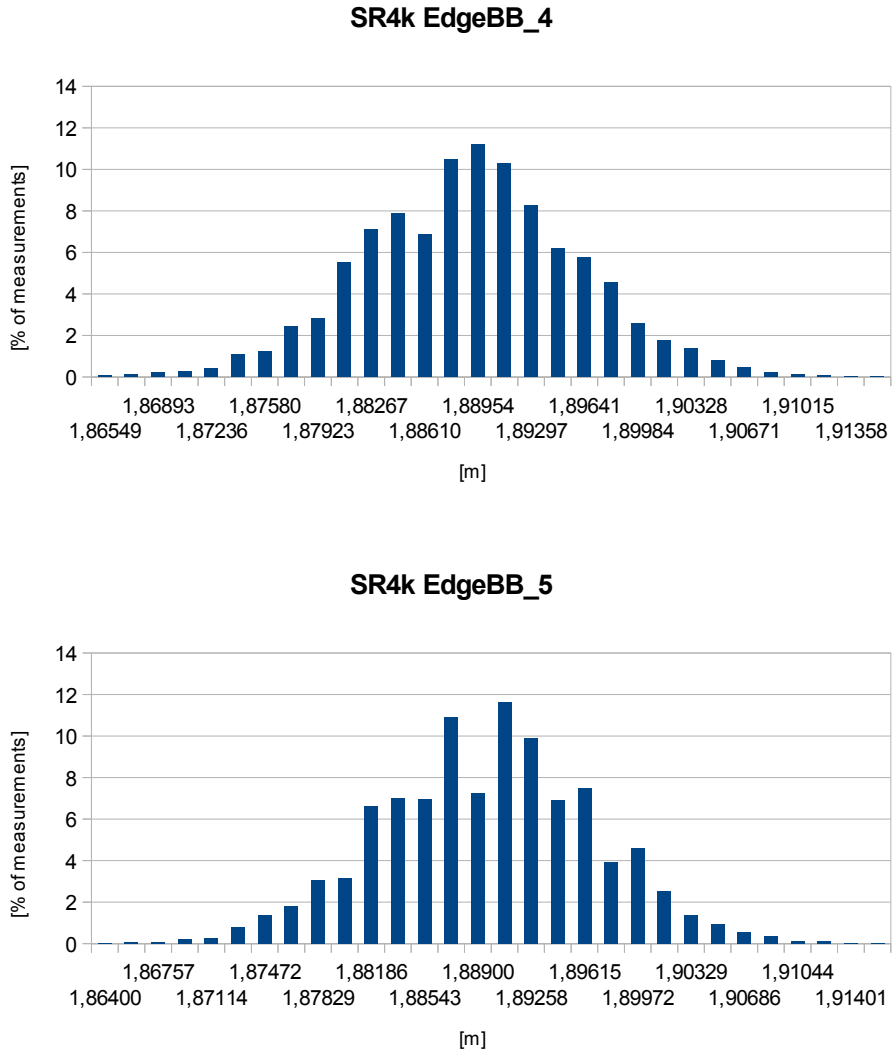
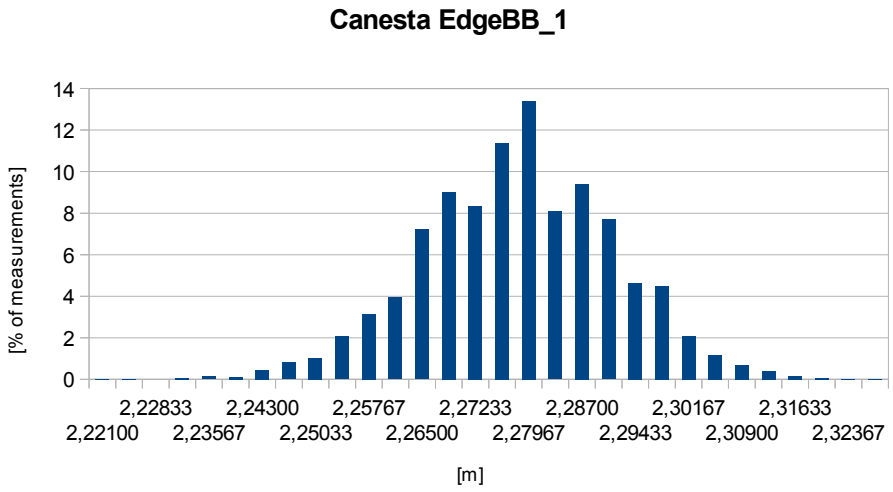
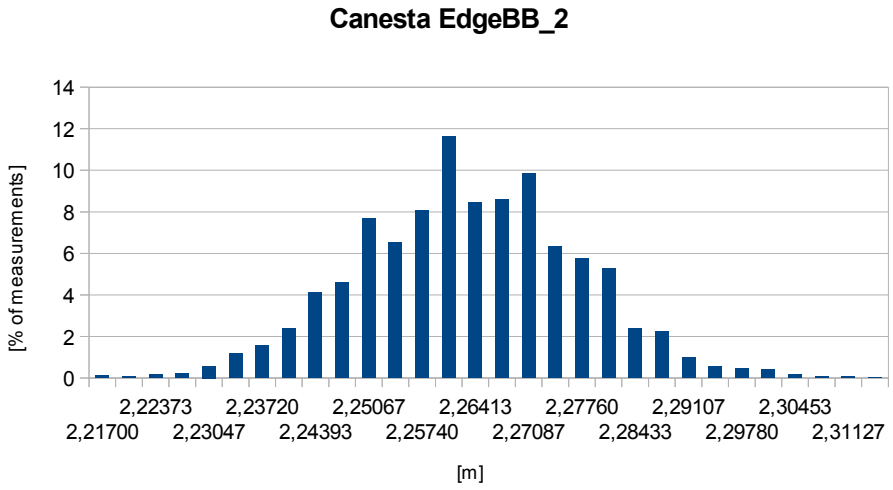


Figure 2.16: Distribution of measurements for Mesa SR4k of $EdgeBB_i$ pixels.





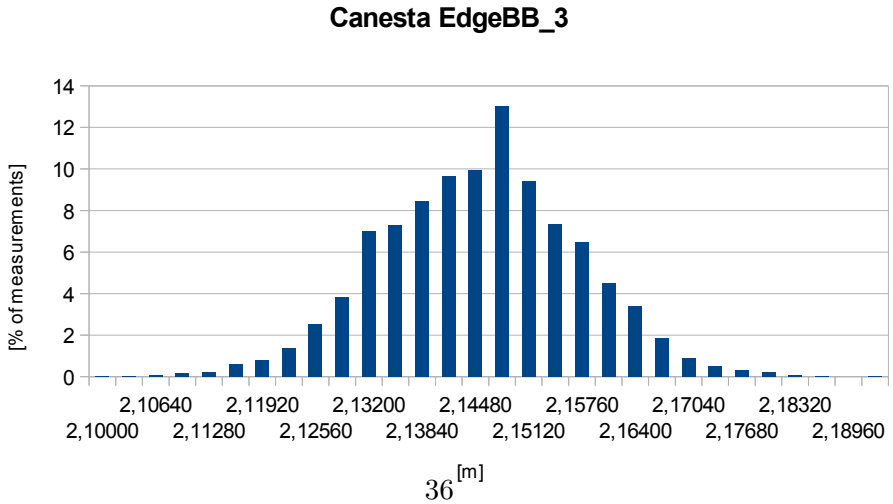


Figure 2.17: Distribution of measurements for Canesta of $EdgeBB_i$ pixels.

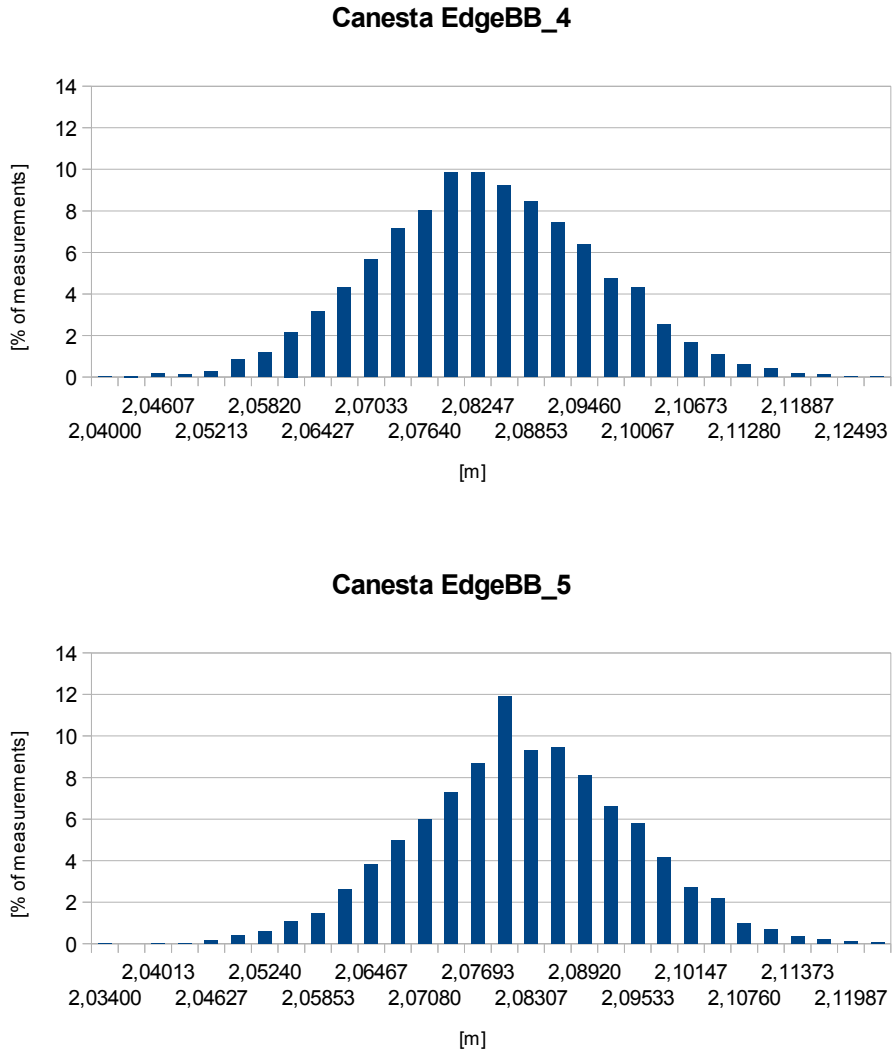
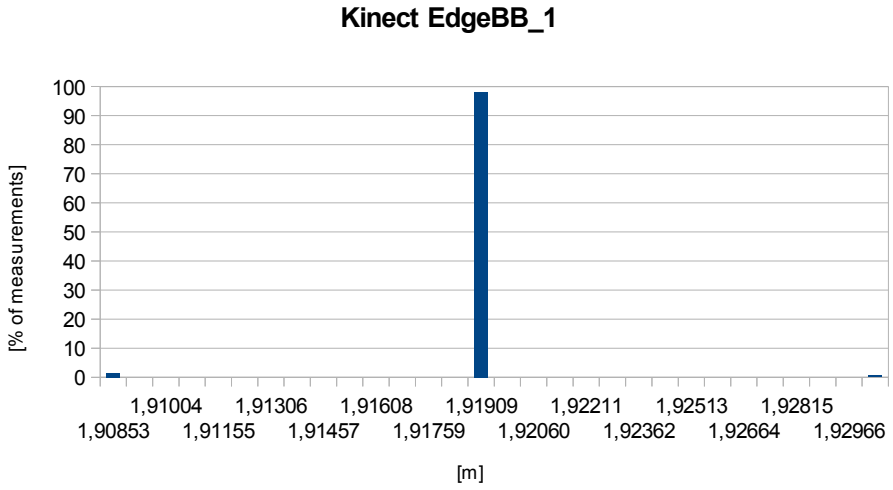
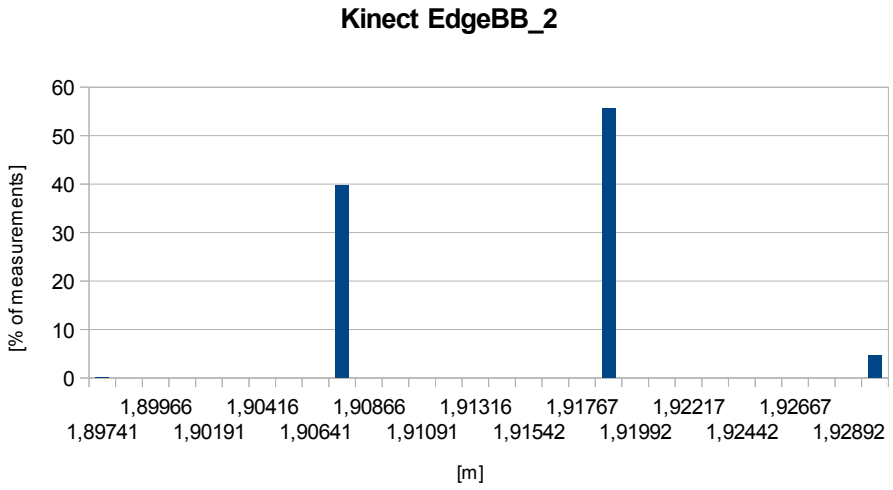


Figure 2.17: Distribution of measurements for Canesta of $EdgeBB_i$ pixels.





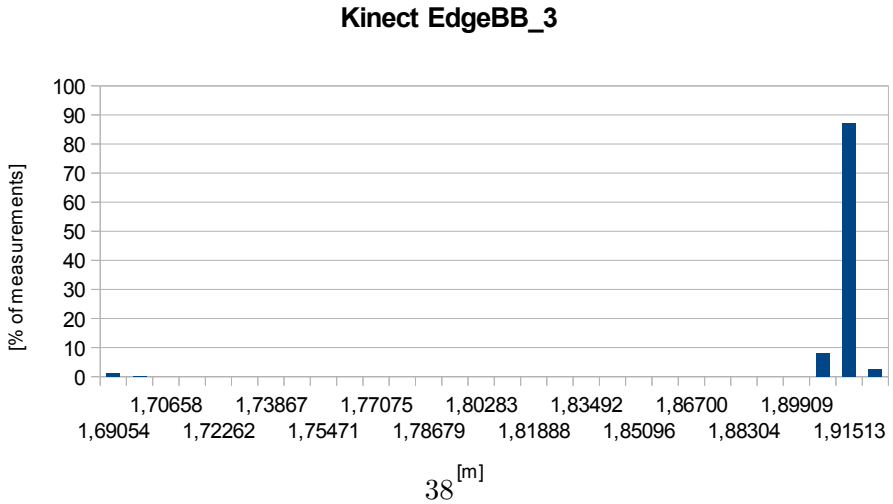


Figure 2.18: Distribution of measurements for the Kinect of $EdgeBB_i$ pixels.

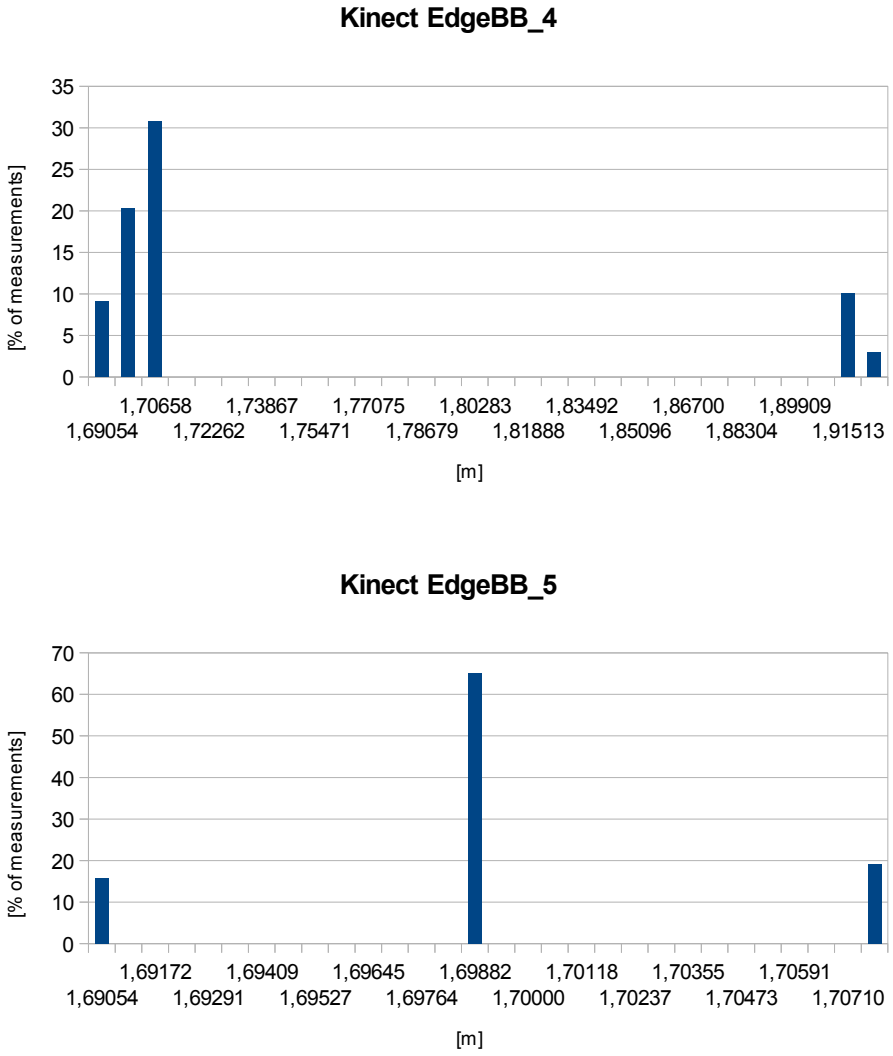


Figure 2.18: Distribution of measurements for the Kinect of $EdgeBB_i$ pixels.

Point p	Device	Min $[m]$	Max $[m]$	$\mu_p[m]$	$\sigma_p[m]$
$EdgeBB_1$	SR4k	1,8640	1,9176	1,8906	0,0072
	Canesta	2,2210	2,3310	2,2800	0,0132
	Kinect	1,9085	1,9312	1,9183	0,0020
$EdgeBB_2$	SR4k	1,8655	1,9170	1,8900	0,0069
	Canesta	2,2170	2,3180	2,2648	0,0136
	Kinect	1,8974	1,9312	1,9145	0,0065
$EdgeBB_3$	SR4k	1,8264	1,8875	1,8531	0,0068
	Canesta	2,1000	2,1960	2,1469	0,0117
	Kinect	1,6905	1,9312	1,9162	0,0207
$EdgeBB_4$	SR4k	1,7117	1,7607	1,7352	0,0064
	Canesta	2,0400	2,1310	2,0850	0.0123
	Kinect	1,6905	1,9312	1,7292	0,0747
$EdgeBB_5$	SR4k	1,6611	1,7044	1,6835	0,0060
	Canesta	2,0340	2,1260	2,0861	0,0125
	Kinect	1,6905	1,70828	1,6997	0,0052

Table 2.2: Values acquired of important parameters on discontinuity points of $EdgeBB_i$.

Chapter 3

Algorithms for Depth–Map Improving

After the analysis performed in previous chapters, it becomes clear how data retrieved by dynamic 3D sensors are affected by errors in a considerable way. They can have several kind of causes, but they can be in most cases modelled and consequently they can be also corrected.
From previous section it is highlighted in particular, how the Microsoft Kinect have lower, but comparable performances with respect to Time-of-Flight devices and it takes more sense trying to improve with a post–processing step the quality of the depth map. Previous considerations about the very low cost of this dynamic 3D sensor, give greater credence to the reasons of improving the quality, with the hope to obtain both a cheap and a less–affected by errors device.
This chapter of the thesis focuses on the preparation of a rigorous method that combine several techniques and algorithms to improve the quality of the depth map acquired by the Microsoft Kinect sensor. Most of these techniques can be easily adapted also to depth–maps acquired by other sensors.

3.1 Error analysis and planning of the methods.

It is worth to remember what kind of problems were decisive to encounter depth-map low quality.
First of all, Kinect suffers of bad edge estimation; this is mainly due for the misalignment of the depth-map with respect to real edges of the image, but also for simple errors of depth value assignment. The bad edge detection is also encountered in correspondence of articulated shapes, where in some cases the depth can also have considerable holes, i.e. unknown values of depth.
The phenomenon of depth absence is also noticed in correspondence of oc-

cluded point, and this problem is due for the distance between IR projector and IR camera of the Kinect device.
The first improving that is researched is of course a repair of the misalignment on the edges, consequently a complete filling of holes, both for articulated and occluded zones of the scene.
In second instance other considerations can be done. The process of quality improvement can also be orientated towards another important aspect of computer vision, that is the resolution. The devices considered until this moment (e.g. ToF Swiss Ranger 4000 and Canesta) are characterized by low values of resolutions. Depth-maps of order of magnitude of 176×144 or 320×200 are for some particular purposes, too much undersized, also considering the fact that actual resolutions for common cameras can grow up to several millions of pixels. Even Microsoft Kinect with an output resolution of 640×480, can be upscaled to more considerable resolution values.
To perform all these operations of improvement, a new acquisition setup has been prepared, and it is reported in Figure 3.1 as a scheme. They have been used a Microsoft Kinect sensor and a Nikon D70s color camera, shown in Figure 3.2.
In this stereo configuration, the Nikon color camera is the left-eye, while

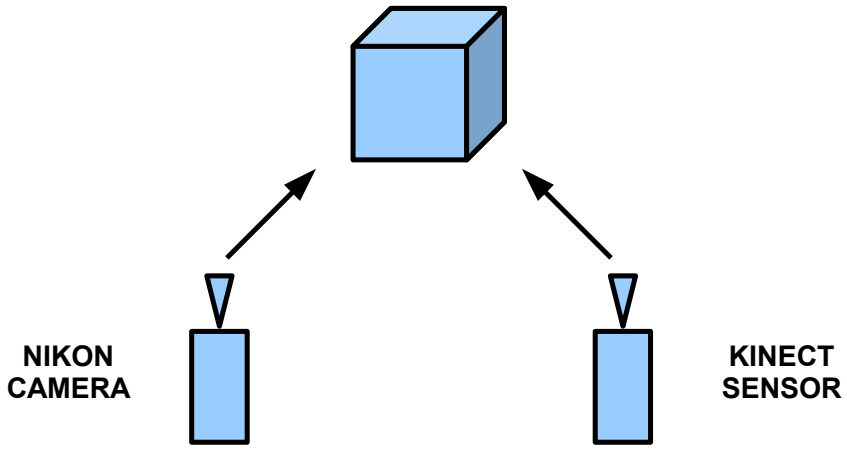


Figure 3.1: New setup for depth-map quality improving.

the right-eye is the Kinect. The set of acquired informations is populated with just two objects:

- Nikon JPEG image of resolution 3008×2000, that is used for colour information of the scene. The image is shown in Figure 3.3.

- Microsoft Kinect depth-map, with both gray scale PNG image and YML text formats, with a resolution of 640×480, utilized for distance information of the scene. The image is shown in Figure 3.4

Figure 3.2: Nikon D70s camera used for color information.

A set of algorithms will be designed and correctly correlated each other o perform a correction on depth-map errors; moreover both color information from the Nikon camera and other algorithms will be used to upgrade the resolution of the depth-map.
The complete pipeline of the methods that have been planned is presented here, while a more detailed analysis on every step will be explained in following sections:

1. Error Detection and Removal
2. Backprojection
3. High Resolution Interpolation

The implementation of the complete procedure has been done, also this time, in `C++` with the support of various standard libraries, and also with a massive use of already cited `OpenCV` computer vision libraries [7]. It had been chosen because it offers some powerful sets of data types, I/O functions and in particular images and matrix processing functions. The huge quantity of algorithms implemented are fully optimized. `OpenCV` is released under a BSD license, so it is free for both academic and commercial use, and it makes the code written able to be redistributed, modified, improved and also integrated in other projects. It is used around the world and a so populated community of developers maintains the code young and clean; moreover it can give help through the web-group and it is easy to find accurate informations about `OpenCV` framework.

Figure 3.3: Color image

3.2 Error Detection and Removal

The first operation that has to be performed is the detection of such pixels that we know to be affected by error. In particular, from previous considerations, it has been highlighted that we have three categories of such pixels:

- Pixels on the edges with misaligned depth.
- Pixels on bad edges estimation.
- Saturated and occluded pixels without depth value.

It is reported in Figure 3.4 the gray scale PNG image of the Kinect depth–map. It presents inside all of such types of pixels affected by error just mentioned in the list. First of all it is worth to remember that the intensity of every pixel of this image have a direct correspondence with its depth value, i.e. the higher the depth value the brighter the pixel, and in particular the black colour is used by the acquisition software to warn about the absence of depth information. So, to detect saturated and occluded points it is enough to consider black pixels of the depth map. Instead to detect the other two categories of pixels, it is necessary to make use of some more advanced computer vision algorithm. Before citing them, some considerations and assumptions must be done on how this error is distributed onto the space domain of the depth map.
The Figure 3.5 shows with a black line an hypothetical shape that has to be detected, for which the dynamic sensor has to retrieve the distance and with a blue line it represents the same shape in the depth map detected by the sensor, where they can occur in some zones the just presented issues.
It is clear how their mutual spatial arrangement is correlated and so it is reasonable to think that their distance cannot overcome an upper bound.

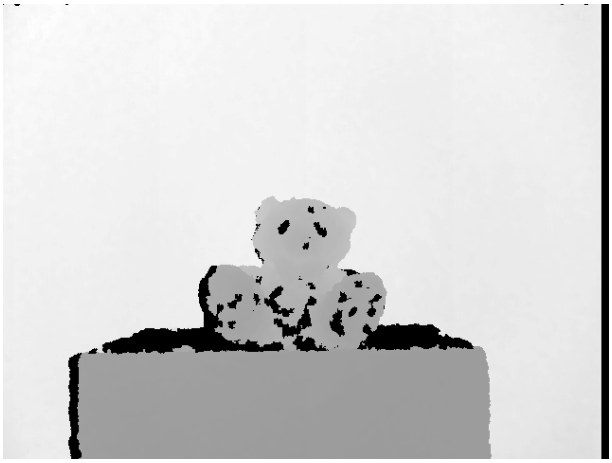


Figure 3.4: Depth PNG image of Kinect depth map.

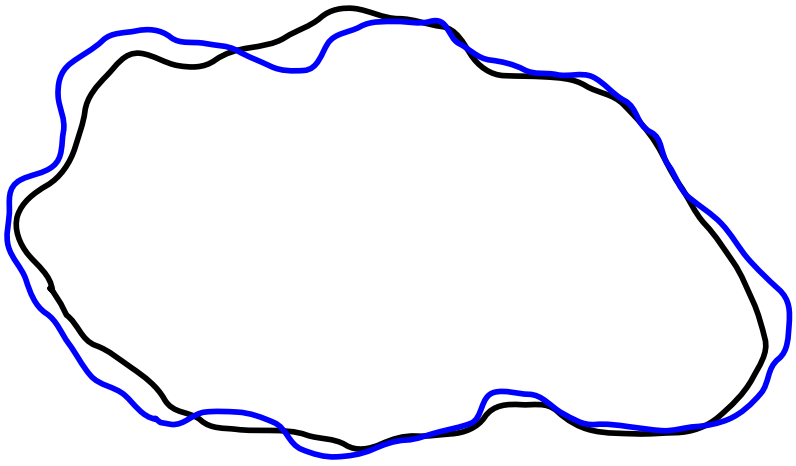


Figure 3.5: Real edge of an object with black line. Detected edge of an object by Kinect sensor with blue line. Ideal representation.

Even if this error has not been modelled with a formal theory, previous analysis and other particular investigations stated that this upper bound cannot overcome the size of 15 pixels, for further considered acquisitions.
The Figure 3.6 shows that it is possible to consider an enlargement of the blue line to catch a considerable amount of pixels with wrong depth value. Those pixels are covered by the area between blue dotted lines. The policy utilized for this error correction procedure completely prunes the corresponding depth values.

So for the implementation of this phase of error detection and removal, it

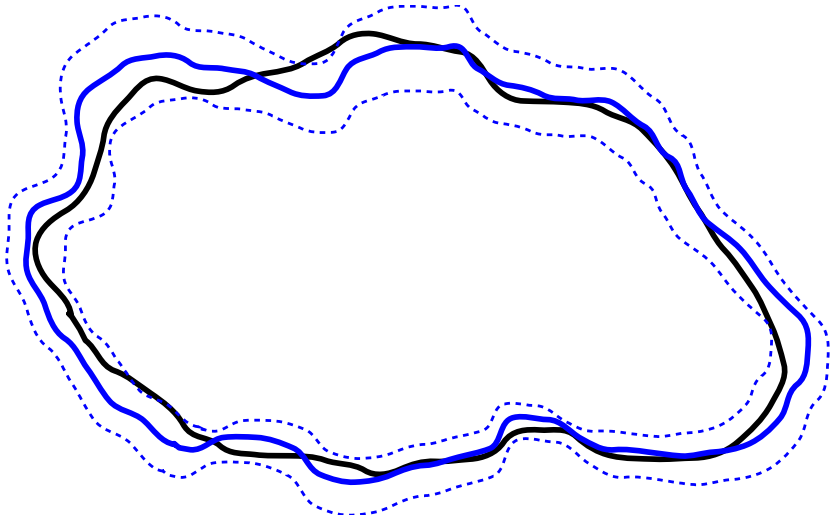


Figure 3.6: The zone between blue dotted lines is the subject for depth value pruning.

Algorithm 1 Edge Error Detection and Removal

Require: *depth,r*
  $edges \leftarrow$ ***canny***$(depth)$
  $dilatedEdges \leftarrow$ ***dilate***$(edges)$
  for $i = 0 \rightarrow r - 1$ **do**
    if $depth\ [i] == black$ **and** $dilatedEdges[i] == black$ **then**
      $dilatedEdges[i] \leftarrow white$
    end if
  end for

has been designed the Algorithm 1, that uses a very well-known in literature edge detector algorithm, i.e. the Canny edge detector [8], and an image morphological operator, i.e. the dilation algorithm [10]. In the text of the algorithm, r is the number of pixel of *depth* image, $depth[i]$ and $dilatedEdge[i]$ are the images where i is their progressive number of pixel, which can span

the integer interval $[0, r-1]$.
Through the use of the ***canny*** operator it is possible to detect the edges of the scene, that are the edges detected from the sensor of Kinect (see Figure 3.7), because ***canny*** operates upon the depth PNG image returned from the acquisition software. Upon this output it is performed the planned enlargement, i.e. the dilation, of the edges area with the second operator ***dilate***, for which depth values will not be taken in account (see Figure 3.8).
The last loop has the purpose to detect all the remaining areas for which no depth values is available, corresponding to black pixels into the depth map, adding a white pixel in the dilated image.
As result of all these operations we have a black and white (boolean) image (see Figure 3.9) that is a sort of mask:

- Black pixels correspond to good values in the depth-map.
- White pixels correspond to wrong and pruned values in the depth-map.

Figure 3.7: Edges detection: output image of canny algorithm.

3.2.1 Canny Edge Detector Algorithm

The challenge of edge detection is a frequent problem that has to be overcome. Qualitatively edges occur at boundaries regions of different color, intensity or texture. Under such conditions, a reasonable approach is to define an edge as a location of rapid intensity variation.
In literature there are a lot of well known procedures and algorithms that perform edge detection, but a particular refinement of existing techniques [10, 16, 7] has been considered, that is Canny algorithm.
In the Canny algorithm, the first derivatives are computed in x and y and then combined into four directional derivatives. The points where these directional derivatives are local maxima, are then candidates for assembling

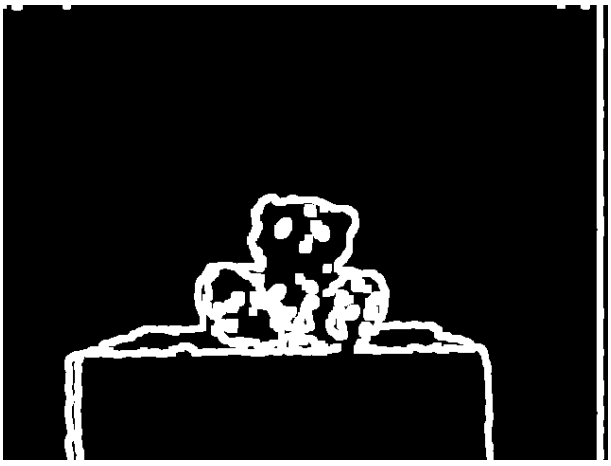


Figure 3.8: Edges dilation: output image of dilate algorithm.

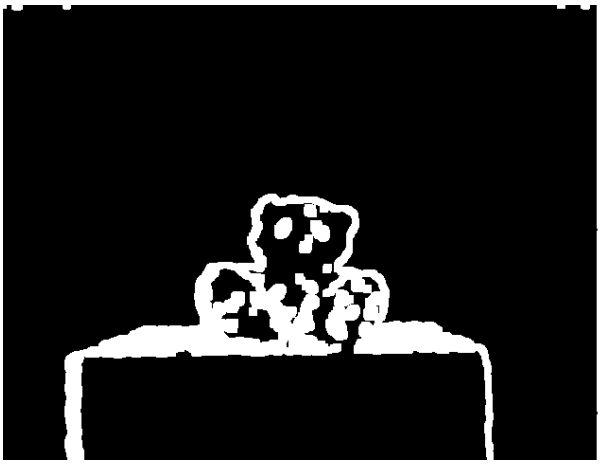


Figure 3.9: Final result: the area of unknown and pruned depth is marked with a white pixel.

into edges. However, the most significant new dimension to the Canny algorithm is that it tries to assemble the individual edge candidate pixels into contours. These contours are formed by applying an hysteresis threshold to the pixels. This means that there are two thresholds, an upper and a lower. If a pixel has a gradient larger than the upper threshold, then it is accepted as an edge pixel; if a pixel is below the lower threshold, it is rejected. If the pixels gradient is between the thresholds, then it will be accepted only if it is connected to a pixel that is above the high threshold.
`OpenCV` contains a useful function to operate such algorithm on an input image, that is

- `cvCanny(input,edges,lowThresh,highThresh)`

where `input` is the Figure 3.4 and `edges` is the Figure 3.7. For the particular implementation they have been used

- `lowThresh=49`
- `highThresh=50`

3.2.2 Dilation Algorithm

The Dilation algorithm belongs to the morphology category of image processing operators. This category contains a lot of as simple as useful algorithms that are very well described in the majority of literature [10, 16, 7]. Dilation is a convolution of some (gray scale or binary) image (or region of an image) A, with some kernel B. The kernel, which can be any shape or size, has a single defined anchor point, i.e. the central point. The kernel can be thought of as a template or mask, and its effect for dilation is that of a local maximum operator. As the kernel B is scanned over the image, the procedure computes the maximal pixel value overlapped by B and replaces the image pixel under the anchor point with that maximal value. This causes bright regions within an image to grow as diagrammed in Figure 3.10.
`OpenCV` contains a useful function to operate such algorithm on an input image, that is

- `cvDilate(input,dilation,kernel,iterations)`

where `input` is the image in Figure 3.7 and `dilation` is the image in Figure 3.8. The `kernel` is a parameter to change the dimension of the kernel B and `iterations` is of course a parameter to set the number of iterations of the kernel B over the input image A. For the particular implementation they have been used

- `kernel=DEFAULT`
- `iterations=3`

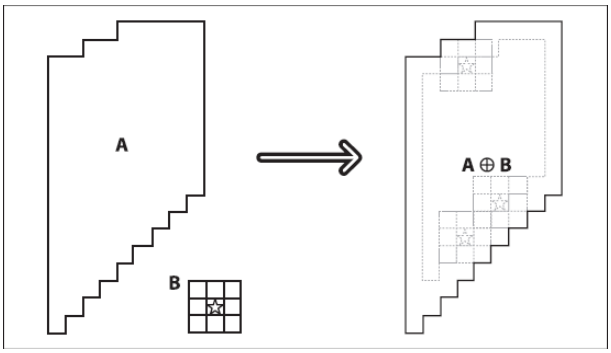


Figure 3.10: Morphological dilation: take the maximum of A under the kernel B.

3.3 Backprojection

Before being ready to use the high resolution informations of the Nikon camera of the stereo setup shown in Figure 3.1, it is necessary to find a correct way to correlate such informations with the Kinect ones. Starting from a couple of images of the same scene from different point of view, it is necessary to find the correspondences of every point between Kinect and Nikon reference systems. The solution for this kind of problem is well-known in literature and it is called stereo calibration[9, 18, 14].
For the particular stereo setup considered, a set of operations must be performed:

- Right camera calibration.
- Left camera calibration.
- Stereo calibration.

This set of consequent calibration steps allows to find the complete geometrical description of the information of the scene. With the first two types of calibration we can find a way to easily pass between 2D and 3D coordinates from a single camera point of view, both for Nikon and Kinect reference systems. The last calibration step allows to express the 3D reference systems of one device in function of the other one.
All these calibration steps impose to execute some particular calculations that made the assumption that every camera satisfies the hypothesis of a pinhole camera model[18, 9, 14].
In general the calibration of *extrinsic parameters* is the procedure that allow to retrieve a decription of the projection of the points that belong to the real world into the image produced by a single camera. The following expression is a matrix representation in homogeneous coordinates that models

the projection of the tridimensional points

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -fk_u & fk_u \cot(\theta) & u_0 & 0 \\ 0 & -fk_v/\sin(\theta) & u_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

In 3.1, $p = [u, v]^T$ are the 2D coordinates of the point in the image of the camera, while $P = [x, y, z]^T$ are the coordinates of the same points in a 3D reference system. The other intrinsic parameters $-fk_u$ and $-fk_v$ are the focal lengths of the camera, and they are expressed in pixels, θ is the angle between the axis u e v, and in the end $[u_0, v_0]$ are the coordinates of the principal point relative to the image reference frame.
The operation of *instrinsic parameters* calibration is the estimation of such parameters that allow the description of the projection of the points in the real world during the process of image making. This model is verified if the coordinates of the tridimensional space are expressed with respect to the reference system of the camera itself, that is when the reference system of the world corresponds with the reference system of the camera.
In case of multiple cameras setup, or in case of not correspondence between the world reference system and the camera reference system the model must be slightly changed.
The difference of the two reference systems can be calculated through the estimation of the parameters of a rototranslation matrix that can be written in general

$$G = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.2}$$

where R is a 3×3 rotation matrix of the camera reference system with respect to the world reference system, and $\mathbf{t}$ is the 3–component vector that explain its translation from the world reference system.
The complete transformation is explained by the following equation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cong \begin{bmatrix} -fk_u & fk_u \cot(\theta) & u_0 & 0 \\ 0 & -fk_v/\sin(\theta) & u_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.3}$$

where $P = [x_w, y_w, z_w]^T$ are the coordinates of the point in the world reference system.
The complete procedure of calibration has been performed through the use of the software Matlab Calibration Toolbox[6], that permits the calculation of all extrinsic and intrinsic parameters.
After the calibration processes, the conversion that has to be performed allows the rendering of Nikon coordinates starting from Kinect coordinates. The complete proceedings are

1. Conversion from Kinect 2D reference system to Kinect 3D reference system.

2. Conversion from Kinect 3D reference system to Nikon 3D reference system, that is considered to be the same of the world reference system.

3. Conversion from Nikon 3D reference system to Nikon 2D reference system.

Introducing a new point of view of the same scene, other considerations must be done. In particular it appears the problem of the occluded points that must be solved in order to correctly project all the right points onto the left reference system. Contextualizing, there can be some points that are visible from the visual of the Kinect, but that are not visible from the point of view of Nikon. This problem requires the utilization of a zBuffer. It is a well known support structure in computer graphics, and it serves to the management of image depth coordinates in tridimensional graphics, usually done in hardware, sometimes in software. It is one solution to the visibility problem, which is the problem of deciding which elements of a rendered scene are visible, and which are hidden.
For this particular setup it is used a static two-dimensional buffer of Nikon resolution dimensions, where for every pixel it is stored the minimum distance z_{min} that this point can have from the left point of view. The Figure 3.11 shows a simple representation of an occlusion phenomenon in a binocular vision of a synthetic scene.

To perform a correct backprojection of the points of the Kinect refer-

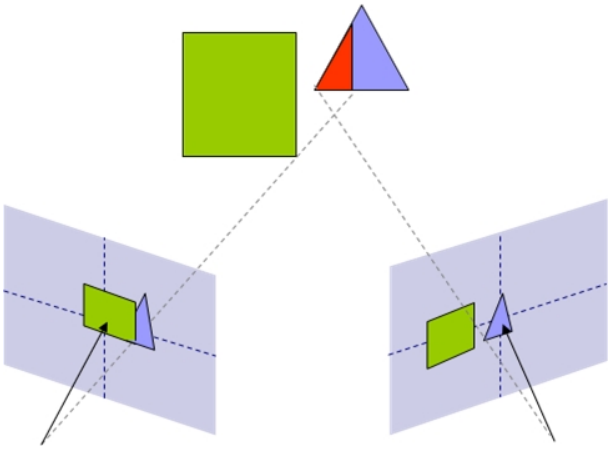


Figure 3.11: Red zone on triangle is an occluded area for the left visual.

ence system, it has been designed the Algorithm 2. The input of the algorithm are the *color* image, that is the high resolution Nikon image, and the *depthRight* image, that is the Kinect depth map. The algorithm continues

Algorithm 2 Backprojection

Require: *colorLeft,depthRight*
  $x \leftarrow getX\,(color)$
  $y \leftarrow getY\,(color)$
  $zBuffer \leftarrow$ ***zbuffer*** $(depthRight)$
  for $i = 0 \rightarrow n-1$ **do**
    $(u_i, v_i, z_i) \leftarrow$ ***backprojection*** $(depthRight[i])$
    if $u_i \in [0, x-1]$ **and** $v_i \in [0, y-1]$ **and** $z_i \leq zBuffer[i]$ **then**
      $depthLeft\,[i] \leftarrow z_i$
    end if
  end for

with the zBuffer and backprojection procedure, that are already cited and that will be explained soon. Into the last loop the if–statement is designed to check for every projected pixel of the depth map, if it falls inside the true coordinates of the Nikon image, and above all if the depth computed for left reference system is less or equal than the zBuffer value, just to avoid that the occluded points become visible.

3.3.1 zBuffer Algorithm and Radial Smoothing

The calculation of the zBuffer is done with the utilization of `OpenCV` matrix data structures, and with the support of matrix operators that are already implemented inside this framework.
Referencing the Figure 3.12, the algorithm considers every possible set of four close points, that is a rectangle of vertexes $A(x, y)$, $B(x+1, y)$, $C(x, y+1)$ and $D(x+1, y+1)$. For these point it is computed the backprojection with the algorithm that will be shown in next section, calculating also the depth value associated.

The projection will be in general a polygon in the reference system of the Nikon camera and in this polygon it is considered the set of two triangles (i.e. $A\widehat{B}C$ and $B\widehat{C}D$, or $A\widehat{B}D$ and $A\widehat{C}D$) that generate a lower value of polygon diagonal between the two possibles $\overline{AD}$ or $\overline{BC}$ segments. Considering the window spanned by the two points of coordinates $\alpha(xMin, yMin)$ and $\beta(xMax, yMax)$, for each pixel inside the window that belongs to almost one triangle, it is computed its smoothed depth z_i. A final check is computed and if z_i is lower than z_i^{min} of such pixel, the value z_i replaces z_i^{min} in the zBuffer matrix.
From these considerations it becomes clear that the zBuffer algorithm assumes the hypothesis that the depth map is completely available, but the starting point of zBuffer calculation is a depth map with some pruned depth values (remember the Figure 3.9). In order to compute completely the zBuffer, it is necessary to retrieve a filled version of the depth map.

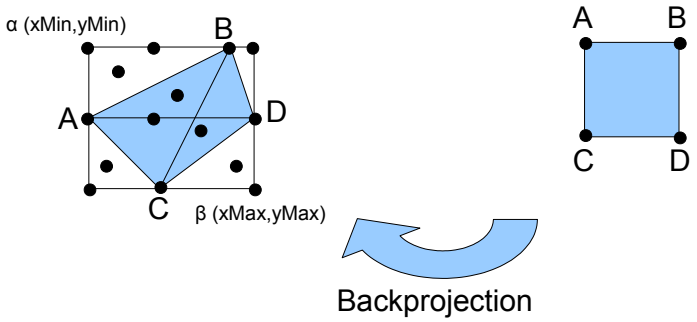


Figure 3.12: zBuffer backprojection scheme.

A simple solution to this problem is to compute, for the pruned pixels that constitutes the holes in Figure 3.9, a weighted mean of the depth of the pixels surrounding them and for which the depth value is still available. This solution is called here radial smoothing.
A concept of distance must be introduced in order to show the algorithm that calculate the depth for the pruned values. It is used the D_8 distance, often called *chessboard distance* [10], where the 8-neighbour pixels that form a square around the considered pixel have distance 1, where the square that surround these pixels have distance 2, and so on. In general given two points $p = (x, y)$ and $q = (s, t)$, the D_8 distance is

$$D_8(p, q) = max(|x - s|, |y - t|) \tag{3.4}$$

The Table 3.1 can explain in a better way this concept. For every pixel with

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Table 3.1: D_8 distance of pixels with respect of central pixel.

unavailable depth value z_u we consider a four zones division of the image space, that are the four cardinal directions, i.e. North, South, West and East. For each of these zones it is caught one available depth value in an iterative way getting z_n, z_s, z_w and z_e, searching for these depth values in a D_8 distance increasing manner. This can lead to a more balanced choice of depth values that compose the final weighting, in fact the procedure can take a value of depth from every distinct cardinal direction. The Figure 3.13

can show an example of such search into the North direction, and the other ones are specular.
The final depth has been designed to be a simple mean of these 4 values

$$z_u = \frac{z_n + z_s + z_w + z_e}{4} \tag{3.5}$$

In Figure 3.14 it is shown the result of radial smoothing operation.

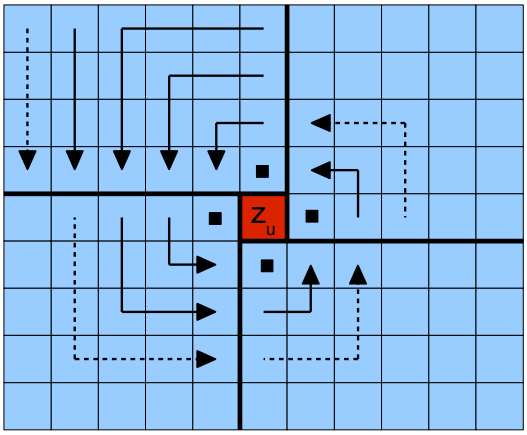


Figure 3.13: The space is divided into four cardinal zones. Iterative searching, with D_8 distance increasing manner to calculate unknown depth value z_u.

3.3.2 Backprojection Algorithm

The backprojection procedure has been already presented and in this section it will be contextualized for the particular setup designed, presented in previous sections and also in Figure 3.1.
Considering this binocular system, some conventions must be declared to explain better every calculation of the backprojection step.
The intrinsic parameters matrix for Kinect sensor is K_R and for the Nikon camera is K_L. The rotation matrix R and the translation vector $\bar{t}$ compose the estimated parameters of stereo calibration.
The first operation is to obtain the 3D coordinates $\left[{x_i}^R, {y_i}^R, {z_i}^R\right]^T$ of the point P_R in the Kinect reference system, starting from 2D coordinates $\left[{u_i}^R, {v_i}^R\right]^T$ of the same point p_R in the bidimensional reference system.

Figure 3.14: The result of radial smoothing operation for the pixels for which the depth is unknown.

It is done with the following transformation

$$ {z_i}^R \begin{bmatrix} {u_i}^R \\ {v_i}^R \\ 1 \end{bmatrix} = K_R \begin{bmatrix} {x_i}^R \\ {y_i}^R \\ {z_i}^R \end{bmatrix} \Longrightarrow P_R = \begin{bmatrix} {x_i}^R \\ {y_i}^R \\ {z_i}^R \end{bmatrix} = {K_R}^{-1} \begin{bmatrix} {u_i}^R \\ {v_i}^R \\ 1 \end{bmatrix} {z_i}^R \tag{3.6} $$

where ${z_i}^R$ is really the depth value retrieved from the Kinect sensor and that is stored into the depth map.
Secondly the consequent operation is the conversion of the reference system, passing to the Nikon 3D coordinates $\left[{x_i}^L, {y_i}^L, {z_i}^L\right]^T$ of the point P_L. For the particular calibration toolbox used, the rotation matrix R and translation vector $\bar{t}$ provided allow to perform the following conversion

$$ P_R = R \cdot P_L + \vec{t} \tag{3.7} $$

that is exactly the inverse rototranslation that is needed, so applying the inverse operation in both the members

$$ P_L = \begin{bmatrix} {x_i}^L \\ {y_i}^L \\ {z_i}^L \end{bmatrix} = R^{-1} \cdot \left(P_R - \bar{t}\,\right) \tag{3.8} $$

the wanted coordinates are calculated.
The final step is the calculation of the 2D coordinates $\left[{u_i}^L, {v_i}^L\right]^T$ of point p_L in the Nikon reference system with utilization of the left intrinsic matrix parameters K_L

$$ {z_i}^L \begin{bmatrix} {u_i}^L \\ {v_i}^L \\ 1 \end{bmatrix} = K_L \begin{bmatrix} {x_i}^L \\ {y_i}^L \\ {z_i}^L \end{bmatrix} \tag{3.9} $$

The result of these matrix multiplications allows to store in a *depthLeft* depth map the computed values of depth ${z_i}^L$ into the bidimensional coordinates $({u_i}^L, {v_i}^L)$. The depth map obtained is shown in Figure 3.15. It is worth to notice that the output of backprojection is a very sparse depth map, where black pixels have still unknown depth value.
To give an idea of the quantity of pixels projected in relations with the number of total pixels of high resolution image, if every Kinect depth pixel would fall into the Nikon coordinates, the sparse depth map would have a fraction of pixels with known depth of about

$$\frac{640 \times 480}{3008 \times 2000} = \frac{307200}{6016000} \cong 0,05 = 5\% \tag{3.10}$$

In general the real number of known depth pixels can be lesser, for the

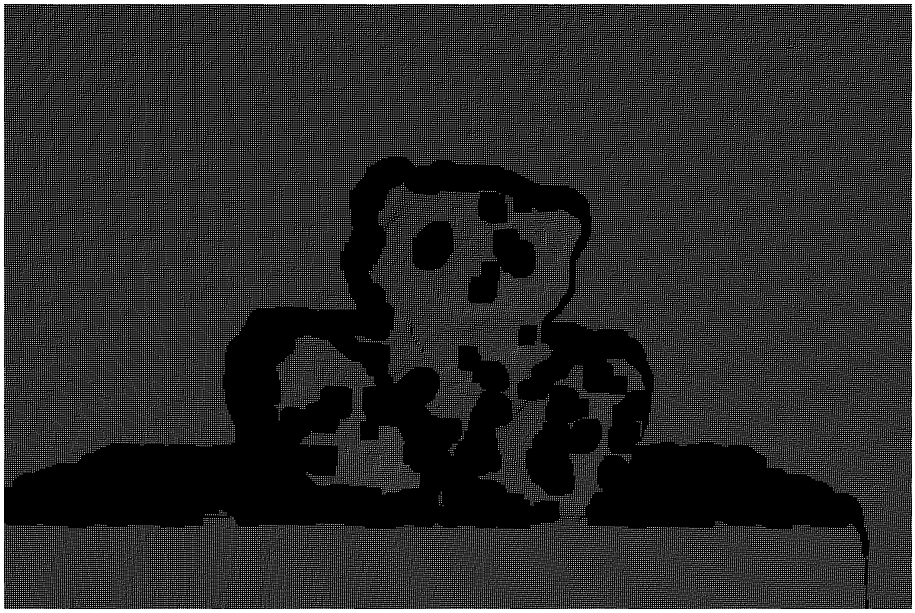


Figure 3.15: Result of backprojection into the Nikon 2D reference system. This sparse depth–map is the input of the interpolation process.

already described pruning operations.

3.4 High Resolution Interpolation

At this point of the complete procedure, the depth informations are ready to be interpolated to completely fill the target resolution that is presented with color image of the scene, shown in Figure 3.3.
Some considerations can be done on the actual situation. Referencing Figure 3.16, it is reasonable to think that pixels with no informations about their

depth value can get some kind of information from the surrounding pixels that have depth informations.

In literature are also available a lot of methods that are applicable to

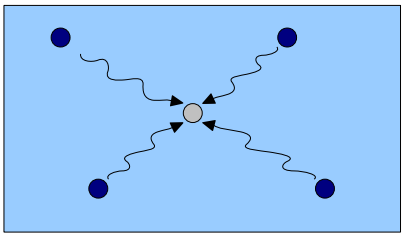


Figure 3.16: Grey circle are pixels with unknown depth value. It can take some informations from a considerable amount of blue circles, that are pixels with known depth value.

this situations and that give a paradigm to combine the information that is present in a local neighborhood. Even if for a different purpose, bilateral filtering [17, 12] wants to combine spatial closeness and range similarity of two pixels to calculate the smoothness of a target pixel. Taking into account those statements, the considerations that have done in this context are rooted in a so different knowledge field.

The propagation of depth information can be considered as the physical process of thermal energy transfer between two corps. An ideal process of thermal energy exchange is schemed in Figure 3.17. The factors that

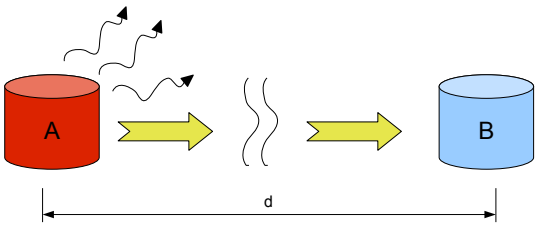


Figure 3.17: Thermal energy transfer between a source A and a recipient B.

influence such process can be grouped in following sets

- The energy intensity of the thermal source, or even better, the thermal gradient from the source A to the recipient B, so the *energy distance*.

- The distance between the two corps A and B, so the *spatial distance*.
- The *conduction* properties of the objects and of the space between the two corps A and B.

It is natural to agree with previous statements, and also the everyday experience can help to understand these concepts:

- The more the temperature difference between two corps, the more heat is transferred.
- The more the vicinity between two corps, the more heat is transferred.
- The more the conductivity of the space between two corps, the more heat is transferred.

To fall within the context of these considerations, there are some theoretical parameters that can be immediately compared to the previous statements and that can represent the subjects for the interpolation process.
Firstly it is trivial to associate the spatial distance between two pixels to their euclidean distance, secondly to interpret the energy distance it is used the color information of the two pixels and, in the end, they are taken into account also the morphology of the shapes in the space between the compared pixels to contextualize the concept of thermal conduction. To do this it is computed a Laplacian of Gaussian (LoG) filter on color image. This tern of factors can concur to a complete estimating of depth of a target pixel in a way that is better explained by following sentences:

- The higher the spatial distance, the lower the contribute of the associated weight.
- The higher the color difference, the lower the contribute of the associated weight.
- The higher the module of the gradient between the compared pixels, the lower the contribute of the associated weight.

3.4.1 Trilateral Filtering

The following procedure has been designed on the basis of the observations made in previous section. The preliminary part is the preparation of Laplacian of Gaussian computation of the scene for the third component of the trilateral filter, that requires this preprocessing. The first step is the searching of available informations in a surrounding zone of unknown pixel p_i in the sparse depth map, with the creation of an adaptive window. The consequent step is the computation of the weights for the trilateral calculation, and in the end the last step of depth estimation itself.

Laplacian of Gaussian

To take in account the morphology of the objects acquired, it is not sufficient to consider the edges of the scenes, but a more detailed information has to be considered. In particular the gradient of the image of the scene can be used to acquire morphological informations of the scene.
It is introduced the Laplacian operator computes on pixel p

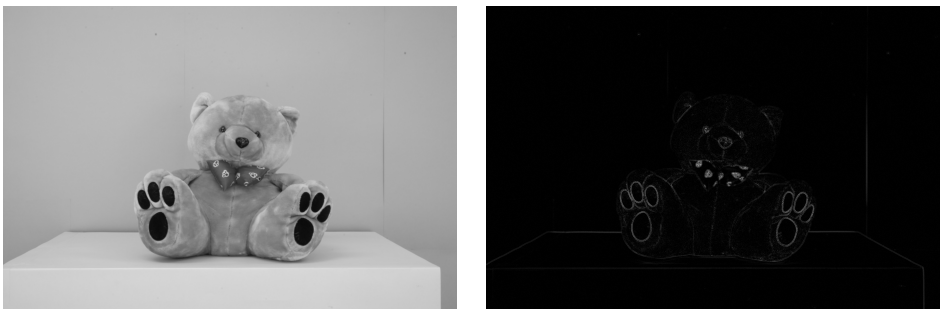
$$\nabla^2 p = \frac{\vartheta^2 p}{\vartheta x^2} + \frac{\vartheta^2 p}{\vartheta y^2} \tag{3.11}$$

Because the Laplacian operator can be defined in terms of second derivatives, a common application is to detect blobs or edges. This means that a single point or any small zone (smaller than the aperture) that is surrounded by higher values will tend to maximize this function. Conversely, a point or small blob that is surrounded by lower values will tend to maximize the negative of this function.
The computation of the second derivative measurement on the image is very sensitive to noise. To solve this problem, the image is often Gaussian smoothed before applying the Laplacian filter with following function

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.12}$$

This pre-processing step reduces the high frequency noise components prior to the differentiation step.In literature the consequent use of Gaussian filter and Laplacian operator is often called LoG, Laplacian of Gaussian[16].

To implement these filters, they have been used the following `OpenCV`



(a) Gaussian pre-processing filter.

(b) Laplacian operator on filtered gray scale image.

Figure 3.18: Application of Laplacian of Gaussian filter on color image.

functions:

- `cvSmooth(input,gaussian, CV_GAUSSIAN,xKernel,yKernel,xSigma,ySigma)`

- `cvLaplace(gaussian,laplace,kernel)`

where `input` of `cvSmooth` function is a gray scale image obtained from Nikon color image, `gaussian` is the Figure 3.18(a), and `laplace` is the Figure 3.18(b). The parameters `xKernel` and `yKernel`, `xSigma` and `ySigma`, and also `kernel` of Laplacian function, have been changed from time to time to perform different tests, but frequent values are

- `xKernel=yKernel=7`

- `xSigma=ySigma=2`

Adaptive Window

Let P_j be the pixel considered for the depth calculation. Let W_j the square window that cover the area surrounding P_j, with dimensions $k \times k$. Let n be the number of pixel with known depth, overlapped by W_j. If k would be constant, and it could not change its value, the procedure should be affected by almost one of following issues. A too much high k values can lead to a number of calculations that can be higher than requested, because a great number of good pixel n is encountered into the scan of the window W_j, leading to very long computation time. On the contrary a too much low value of k can lead the number of good pixels detected to be zero. This can mean in some cases a depth map with some zones for which the depth cannot be estimated, e.g. in the zones where pixels have been pruned in previous steps, because W_j doesn't overlap any known depth value.
A solution for this problem is the utilization of an adaptive window, that can change its dimensions. The paradigm utilized follows the argument of previous radial smoothing algorithm. In fact the size of W_j can grow up until n reaches a selected threshold t of good pixels detected. In the implementation the threshold is counted upon four components, i.e. t_{North}, t_{South}, t_{West} and t_{East}, that have to take into account the distribution of the spatial location of detected pixels. The Figure 3.19 reports a scheme of the window centered in pixel P_j. The external green crown C_l represents the area of possible subsequent increase, that is effectuated if and only if the number of good pixels into grey area of actual window does not overcome the lower bound t selected. The value l corresponds to the chessboard-distance of pixels that will constitute the following increment crown C_l.
This solution permits both to obtain a spatial balanced contribute of good pixels, that doesn't depend on the magnitude of the holes in the depth map and also a efficient research of known pixels. The procedure of adaptive window calculation is reported in Algorithm 3.

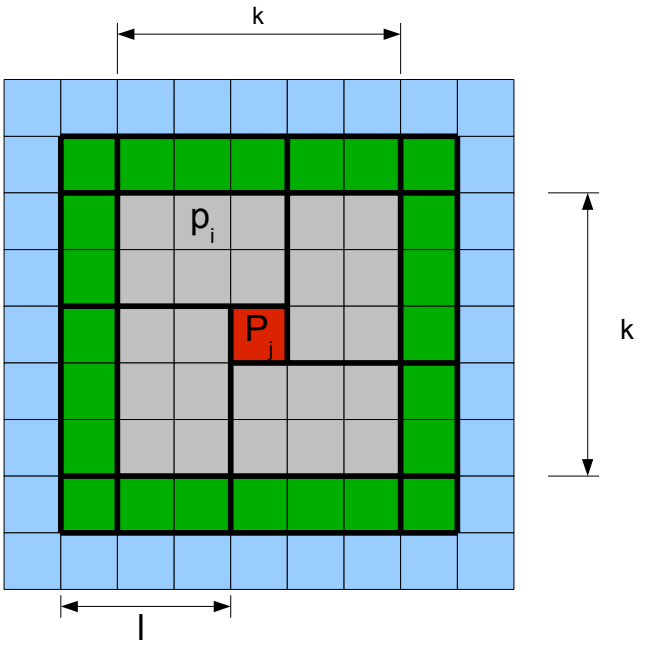


Figure 3.19: Adaptive window W_j. Grey pixels represent actual window, while the green pixels make the crown C_l added to actual window if $n < t$.

Algorithm 3 Adaptive Window

Require: *sparseDepth*, t, P_j
  $l \leftarrow 1$
  $n \leftarrow 0$
  while $n < t$ **do**
    Consider the crown C_l *centered in* P_j
    $c \leftarrow getNumberOfPixel(C_l)$
    for $i = 0 \rightarrow c - 1$ **do**
      if $C_l[i] \neq black$ **then**
        $n \leftarrow n + 1$
      end if
    end for
    $l \leftarrow l + 1$
  end while
  $kernel \leftarrow \bigcup_{i=1}^{l-1} C_i$
  $k^2 \leftarrow getNumberOfPixel(kernel)$
  return k

Weights Computation

At this point the procedure can start some computations from the known pixels that are covered by the window W_j.
For every good pixel p_i the algorithm has to compute

1. The spatial distance with the target pixel, $d_s(p_i, P_j)$.
2. The color distance with the target pixel, $d_c(p_i, P_j)$.
3. The maximum value of the module of the gradient value in the route between p_i and P_j, $maxGrad(p_i, P_j)$.

To calculate the euclidean distance it is used the canonical formula for two dimensions distance

$$d_s(p_i, P_j) = \sqrt{(x_p - x_P)^2 + (y_p - y_P)^2} \tag{3.13}$$

on the basis of the cartesian coordinates of the two points.
For the color distance it has to be introduced a particular color space. Its characteristics have to support a fast and simple measurement of distance, and also its value must correspond in some way to a perceived dissimilarity of colours compared. CIE-Lab color space satisfies all these hypothesis because it is based on a large body of psychophysical data concerning color-matching experiments of human observers[17]. So, also in this case the distance is computed with the well-known formula of euclidean distance, but this time on the tridimensional space of CIE-Lab parameters

$$d_c(p_i, P_j) = \sqrt{(L_p - L_P)^2 + (a_p - a_P)^2 + (b_p - b_P)^2} \tag{3.14}$$

The last component is represented by the maximum value of module of the gradient that is present in the route from p_i to P_j. See Figure 3.20 as reference.
The algorithm has to find the values of gradient, starting from pixel p_i. It calculates the four D_4 distances of neighbour pixels with respect to the pixel P_j [16]. For the particular pixel p_l that minimize the value of $d_s(p_l, P_j)$ it is checked the gradient that have been previously calculated by LoG operator, and it is saved the maximum value of gradient encountered. The procedure is repeated until the p_l becomes P_j, i.e. P_j is reached.
The complete procedure of distances computation is reported in Algorithm 4.

Trilateral filter computation

For all n known pixels p_i in W_j, the weights have been computed and they can concur to calculate the target value of depth $\widehat{d_j}$ for the pixel P_j.

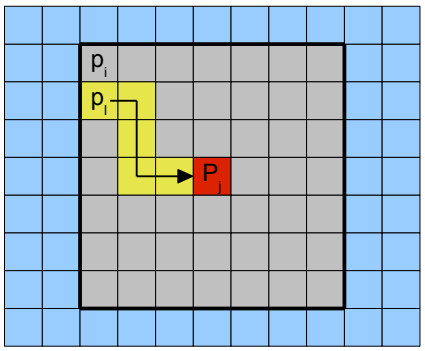


Figure 3.20: Gradient algorithm crosses the minimum path, represented by yellow pixels, saving the maximum of module of the values.

Algorithm 4 Distances Computation

Require: *sparseDepth, color, gradient, k*
  for $i = 0 \rightarrow k^2 - 1$ **do**
    $t \leftarrow 0$
    if $kernel[i] \neq black$ **then**
      $depth[t] \leftarrow kernel[i]$
      $d_s[t] \leftarrow d_s(p_i, P_j)$
      $d_c[t] \leftarrow d_c(p_i, P_j)$
      $grad[t] \leftarrow 0$
      $p_l \leftarrow p_i$
      while $d_s(p_l, P_j) == 0$ **do**
        $grad[t] \leftarrow max\ (grad[t],\ grad(p_l, P_j))$
        $p_l \leftarrow argmin_l\{D_4\ distances\ of\ p_l\ neighbours\}$
      end while
      $t \leftarrow t + 1$
    end if
  end for
  return $depth[],\ d_s[],\ d_c[],\ grad[t]$

Taking inspiration from the bilateral filtering paper[17], it is used a exponential function with gaussian morphology, for the calculation of every weighting component.
In particular the depth $\widehat{d_j}$ is estimated through the use of following function:

$$w(p_i, P_j) = e^{-\left(\frac{d_s(p_i,P_j)}{\sigma_s}\right)^2} e^{-\left(\frac{d_c(p_i,P_j)}{\sigma_c}\right)^2} e^{-\left(\frac{maxGrad(p_i,P_j)}{\sigma_g}\right)^2} \tag{3.15}$$

that is used as weight at the numerator, and as normalization factor at the denominator, into the depth estimation equation

$$\widehat{d_j} = \frac{\sum_{i=1}^{n} d_i \cdot w(p_i, P_j)}{\sum_{i=1}^{n} w(p_i, P_j)} \tag{3.16}$$

3.5 Computational complexity analysis

Before presenting some results on test cases and comparing them each other, it is important to make an analysis on computational complexity of the algorithm presented in previous section.
The analysis takes in consideration only the most innovative part of the complete procedure, that is the trilateral filter, leaving to the reader the analysis of other algorithms (e.g. canny edge detector or dilation) to their references.
Let's consider some parameters:

- m is the number of total unknown pixels into the high resolution image.
- t is the threshold (lower bound) selected for number of known pixels overlapped by the window W_j.
- k is the dimension of square adaptive window W_j, such that the threshold t is exceeded.
- $\frac{r}{R}$ is the ratio between the number of pixels r of the source depth map at low resolution, and the number of pixels R of the target depth map at high resolution. The assumption is that $R > r$.

Search of known pixel in W_j

For every unknown pixel, the algorithm has to find into the window W_j the number of known pixels. If the lower bound t is not reached, at the next loop the counter will search only into the external crown, to maintain a lower

complexity. For these reasons the research step requires $k \times k$ comparisons, that leads to a computational complexity of

$$O(k^2). \tag{3.17}$$

Calculation of weights

The consequent step is made by the calculation of the tern of weights. For how concern spatial and color distance, it is simple to understand that for every known pixel they are performed 6 operations for spatial distance (see Equation 3.13), and they are performed 9 operations for color distance (see Equation 3.14), which lead to $15 \times t$ total operations, and to $O(t)$ computational complexity. For the gradient factor, the consideration has to be lightly changed. For every known pixel, it is computed the shortest path between p_i and P_j, and for every crossed pixel it is made one comparison. The maximum length of such route is $k/2 + k/2$, that leads to a complexity $O(tk)$.
Summing for the three components, the calculation of distances requires

$$O(t) + O(t) + O(tk) = O(tk). \tag{3.18}$$

Depth Estimation

The last step of trilateral filtering is constituted by the estimation of unknown depth through the use of Equation 3.16, which counts a number of constant operations for every unknown pixel t of W_j, leading to a complexity computation of

$$O(t). \tag{3.19}$$

Total Accounting

The total sum of three contributes for trilateral filter is

$$O(k^2) + O(tk) + O(t) = O(k^2) + O(tk). \tag{3.20}$$

It is important to remember that these operations are repeated for every window, and so for every unknown pixel m. In general for every unknown pixel the dimension k of the window is not the same for every step. From a theoretical point of view, the size of W_j can grow up indefinitely but the assumption is that this fact cannot happen in reality. In fact k is directly correlated with t and with the ratio r/R in such way

$$k^2 \frac{r}{R} = t \quad \Longrightarrow \quad \overline{k} = \sqrt{\frac{tR}{r}} \tag{3.21}$$

This equation leads to the value $\overline{k}$, assuming a uniform distribution of the known pixel on the surface of high resolution depth map. This value can be

considered for the analysis of computational complexity. In particular we can rewrite using $k = \overline{k}$

$$m\left[O(k^2) + O(tk)\right] = O(mk^2) + O(mtk). \tag{3.22}$$

It is reasonable to think that t is a constant, compared with m, so previous explanation can be slightly changed:

$$O(mk^2) + O(mtk) = O(mk^2) + O(mk) = O(mk^2) \tag{3.23}$$

For previous assertions, the computational complexity highly depends from the size of window W_j, that is a direct consequence of r/R ratio itself. When $\frac{r}{R} \cong 0$, i.e. the high resolution $R \gg r$, k can have a very high value, on the contrary if $\frac{r}{R} \cong 1$, i.e. the target resolution is not so high, the size of k doesn't grow up in a considerable way.
Concluding, on real utilization of this algorithm, it is reasonable to find a good trade off between the upscaling factor of the depth map resolution and the computational time length.

Chapter 4

Results

It is important to make a better explanation of the implementation of the trilateral filter algorithm.
In particular, referencing with Function 3.15 that is the weighting function used for the depth value estimation, they must be taken into account the possible values of functions $d_s(p_i, P_j)$, $d_c(p_i, P_j)$ and $maxGrad(p_i, P_j)$.
The first distance $d_s(p_i, P_j)$ can be at most the diagonal of high resolution image, and in the considered case this value can be $\sqrt{3008^2 + 2000^2} \cong 3612$.
A slightly different consideration has to be done for $d_c(p_i, P_j)$; this distance is computed over the three 8–bit spaced components L–a–b, leading to $\sqrt{3 \times 255^2} \cong 441$ maximum distance value. In the end, $maxGrad(p_i, P_j)$ is retrieved over a float image, that is the result of LoG computation, and its maximum values is 1. So the three components of weighting function can assume the values in following ranges:

- $d_s(p_i, P_j) \in [0, 3612]$
- $d_c(p_i, P_j) \in [0, 441]$
- $maxGrad(p_i, P_j) \in [0, 1]$

This clarification has been done because they have been performed different tests with several values of σ_s, σ_c and σ_g to stress the performances of the trilateral filter algorithm. The Table 4.1 contains a fast comparison between σ values.

σ_s	σ_c	σ_g
361,2	44,1	0,1
721,4	88,2	0,2
180,6	220,5	0,5
3612	441	1

Table 4.1: Parameters of weighting function. For each each row different values of parameters lead to same weights magnitude.

4.1 Experimental results

Before showing a complete analysis of results, it is important to recall what were the objectives for which this algorithm has been designed.
The most important scope is of course the improving of edges accuracy in the depth–map, fixing the misalignment between real and acquired edge and in a second instance also an estimation of depth into the holes of saturated zones. Both of these improvements have to be realized within an operation of resolution upscaling.
In Figure 4.1 they are reported and compared each other source depth–map with $r = 640 \times 480$ and target depth–map with $R = 3008 \times 2000$, with their respective aspect ratios preserved. The upscaling factor is countable in about $R/r \cong 19,58$.

The image of high resolution depth–map is shown in Figure 4.2. For this

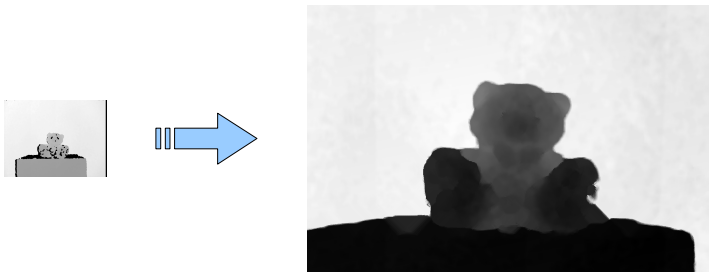


Figure 4.1: Upscaling of source depth–map is performed. Mutual aspect ratios are preserved in this Figure.

depth–map, the 255 gray scale values are mapped into the range of minimum and maximum values of distance. So the black pixels are the closer values of depth, while the white pixels are the further values.

It is worth to notice that the resolution upscaling is performed with a good final result. The Figure 4.2 presents a complete filling of the depth–

Figure 4.2: High resolution depth–map. It is the output of the trilateral filter algorithm.

map, considering the huge absence of informations in the input of trilateral smoothing (see the image in Figure 3.15). Concerning with edge estimation the algorithm shows a very good behaviour too, restoring the researched edge alignment in almost the totality of shapes. As it is shown in Figure 4.3, the overlapping of color image upon the depth–map image is very faithful with the shape of the object acquired.

On the contrary for big holes, the algorithm doesn't achieve the desired

Figure 4.3: Edge alignment is restored on output depth–map. Blue line is the edge in the color image.

result. In particular for the slanted surface of the table the estimation has big area where depth–map has a low accuracy. The phenomenon is shown in Figure 4.4.

This fact is certainly due for the threshold t setted to a particular value

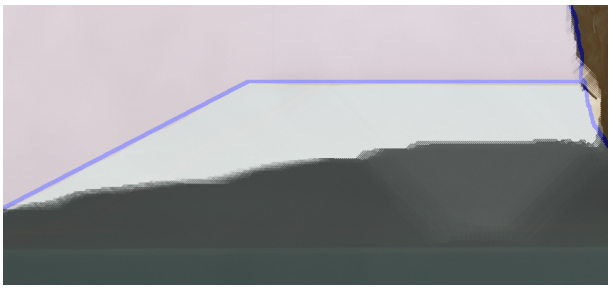


Figure 4.4: Depth estimation is affected by error on very large zones without depth values. Blue line is the edge in the color image.

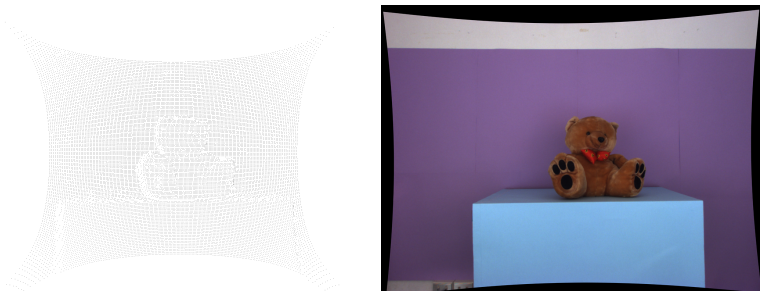
that was not sufficient to retrieve depth values such that the surface is correctly estimated. In fact increasing t better estimation is achieved, but at the same time also computational time grows up in considerable way.
There are other small imperfections that are encountered all over the depth–

map in correspondence of depth discontinuities. This phenomenon is mainly due for the operation of fusion between the sparse depth–map with the estimated depth–map. They are combined in a way that all the known pixel are copied into the output image and the missing values are replaced by the estimated ones. This operation of mixing raw values with smoothed values create the not so natural presence of close white and black pixels[1].
To justify this fact it is important to remember that the trilateral filter works upon strange conditions. In literature every type of filter, smoothing or morphological ones, processes the input image (or in a more theoretical way, the input matrix) with the hypothesis that in all over the input they are available data on which it computes the output value: very often the kernel (i.e. the window) parses all the surface of the input, and the data overlapped by the kernel are retrieved in a complete homogeneous distribution. For the filter implemented in this work, this assumption is never verified and in most cases the depth acquisition made by Kinect produce vaste zones with absence of information. These zones can be in some cases (e.g. the slanted surfaces) too big, and they can lead to bad estimation by the trilateral filter.
To support this statement the trilateral filter has been applied on another sparse depth–map with more uniform distribution of values on high resolution space, generated by a Time-of-Flight device, i.e. Mesa SR4k. As it was already showed in previous chapters, in general ToF devices do not suffer of high absence of depth detection, and in particular articulated shapes and slanted surfaces do not decrease the quality of their depth–maps. It is worth to notice that this test has been performed with $r = 176 \times 144$ and with $R = 1032 \times 778$, so an even more higher upscaling factor $R/r \cong 31,67$ than Kinect case. The sparse depth–map is shown in Figure 4.5(a) and the color image of the same scene is presented in Figure 4.5(b).

The trilateral filter has been applied on this input data and the results obtained is shown in the Figure 4.6.

This time the depth map obtained is not affected by any kind of alignment error, and in particular also in the zone of slanted surface of the table, the depth obtained is consistent with the shape of the scene, moreover the algorithm has better time performances.
They are shown several tests of trilateral filter processing in following Figures. It is not performed a deep analysis on every depth map produced, but they are listed to permit a better understanding of the results on even more cases.
In Figure 4.16 and 4.17, it can be viewed the application of depth map upscaling on a human subject, to give an idea of the results on such utilization fields.

[1] An eventual post–processing step with smoothing filter can be done to overcome these little imperfections.



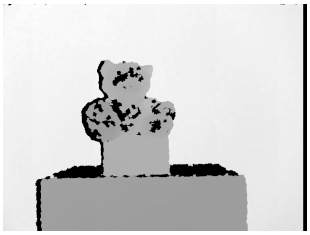
(a) Sparse depth map, generated by ToF device.

(b) Color of the scene, generated by a color camera.

Figure 4.5: Color and sparse depth map for Time-of-Flight trilateral filter test. It has been applied an undistortion operation in the calibration process.

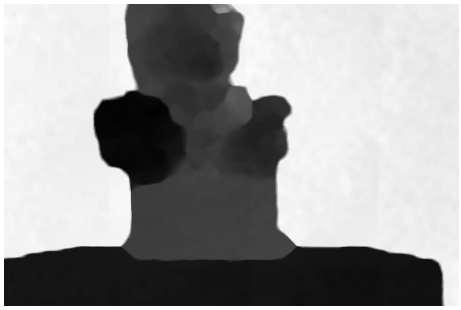
Figure 4.6: High resolution depth map obtained on ToF depth data.

They are shown low resolution depth map produced by the Kinect, high resolution color image acquired by the Nikon camera and in the end the result of the trilateral filter computation with the parameters $\sigma_d = 360$, $\sigma_c = 44$ and $\sigma_g = 0,1$, for every scene.



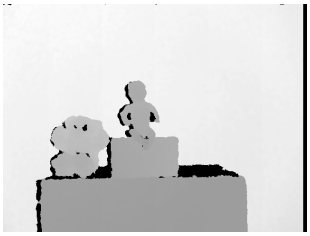
(a) Kinect depth map.

(b) Nikon color image.



(c) High resolution depth map, generated by trilateral filter.

Figure 4.7

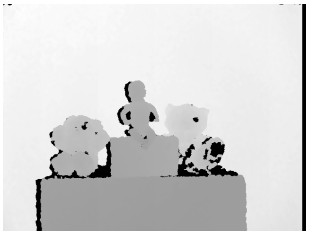


(a) Kinect depth map.

(b) Nikon color image.

(c) High resolution depth map, generated by trilateral filter.

Figure 4.8

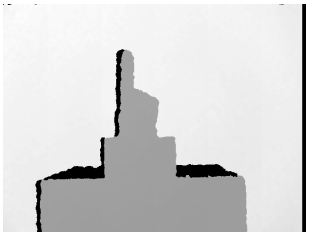


(a) Kinect depth map.

(b) Nikon color image.

(c) High resolution depth map, generated by trilateral filter.

Figure 4.9

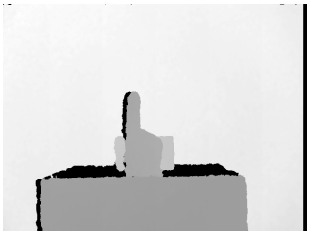


(a) Kinect depth map.

(b) Nikon color image.

(c) High resolution depth map, generated by trilateral filter.

Figure 4.10

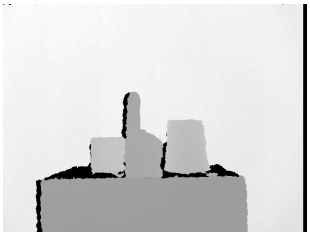


(a) Kinect depth map.

(b) Nikon color image.

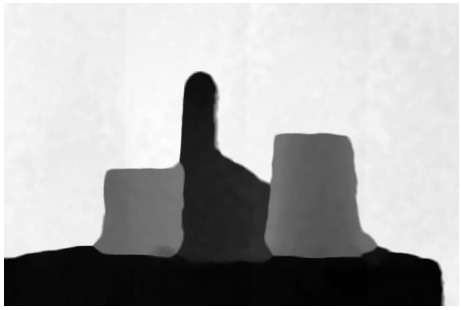
(c) High resolution depth map, generated by trilateral filter.

Figure 4.11



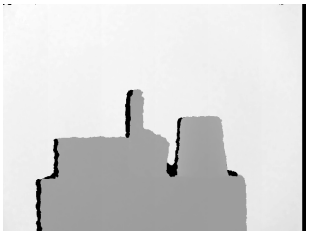
(a) Kinect depth map.

(b) Nikon color image.



(c) High resolution depth map, generated by trilateral filter.

Figure 4.12

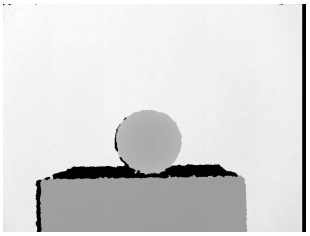


(a) Kinect depth map.

(b) Nikon color image.

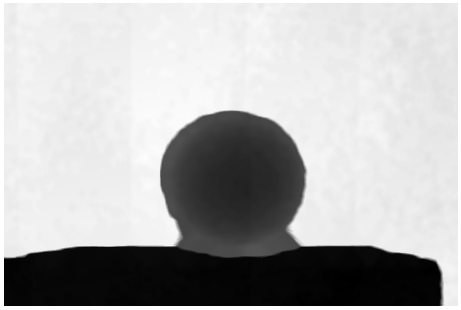
(c) High resolution depth map, generated by trilateral filter.

Figure 4.13



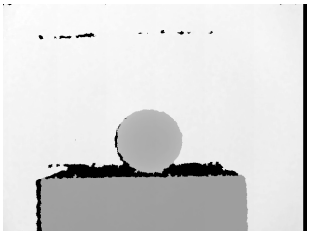
(a) Kinect depth map.

(b) Nikon color image.



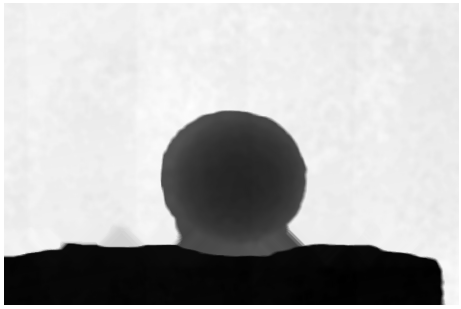
(c) High resolution depth map, generated by trilateral filter.

Figure 4.14



(a) Kinect depth map.

(b) Nikon color image.



(c) High resolution depth map, generated by trilateral filter.

Figure 4.15

(a) Kinect depth map.

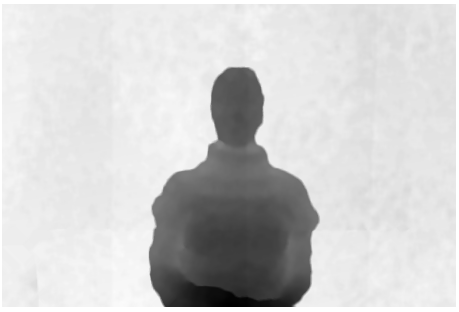
(b) Nikon color image.

(c) High resolution depth map, generated by trilateral filter.

Figure 4.16

(a) Kinect depth map.

(b) Nikon color image.



(c) High resolution depth map, generated by trilateral filter.

Figure 4.17

Conclusions and further improvements

In this thesis they have been accurately analysed the performance of both Time–of–Flight and Microsoft Kinect 3D dynamic sensor. The comparison is articulated on the basis of rigorous acquisitions and a deep characterization is made upon the huge set of measurements performed with Mesa SR4k, Canesta and Microsoft Kinect devices.
Even if the Kinect performances are globally lower the ToF ones, its affordability justifies the aim to improve the quality of depth maps retrieved by this new 3D sensor. The designed pipeline of work has the purpose to remove artefacts and to detect errors, moreover it wants to significantly increase the resolution of such depth–map using a smart variant of bilateral filter, adding the utilization of shapes information for the calculation of a Trilateral Filter.
The effectuated tests highlighted the robustness of the algorithm against edge errors fixing, but reported also too high computation time of the implementation, to fill with accuracy the saturated zones of the depth map. How demonstrates the application of the algorithm on a more uniform sparse depth map, the concept of trilateral filter weighting is both efficient and innovative.
Moreover the computational complexity can be reduced for the worst cases that have been encountered, because a big part of the procedure can be parallelized, e.g. the distance measurements, and another one can be implemented on GPU, e.g. backprojection and z–Buffer procedures, also to achieve real–time computation of the trilateral filter.
Finally, the technique presented can be integrated in almost every computer vision software toolkit, that is designed for the geometrical acquisition of the environment, through the use of a 3D dynamic sensor.

Appendix A

Code Documentation

In this appendix it is contained the documentation for the most important implemented functions of Trilateral Filter project. The documentation has been created with `Doxygen`; it shows all the function declarations of methods used into the implementation of Trilateral Filter.
The complete latex pdf file has been attached in following pages.

Chapter 1

File Documentation

1.1 Backprojection.h File Reference

It contains backprojection functions.

Functions

- void kinect2D3D (int u_r, int v_r, double z_r, CvMat ∗p_r)

  It calculates 3D Kinect coordinates p_r from 2D coordinates u_r and v_r and depth value z_r.
- void kinect3Dnikon3D (CvMat ∗p_r, CvMat ∗p_l)

  It calculates 3D Nikon coordinates p_l from 3D Kinect coordinates p_r.
- void nikon3D2D (CvMat ∗p_l, CvMat ∗leftPoint)

  It calculates 2D Nikon coordinates leftPoint from 3D Nikon coordinates p_l.
- bool pointInTriangle (CvMat ∗point, CvMat ∗a, CvMat ∗b, CvMat ∗c)

  Check if point belongs to a b c triangle.

1.1.1 Detailed Description

It contains backprojection functions. Methods implemented can pass from 2D Kinect reference system, to 3D Kinect, to 3D Nikon, to 2D Nikon reference system. All the intrinsic and extrinsic parametrs matrix are hard-coded. The CvMat matrix are CV_-64FC1.

1.1.2 Function Documentation

1.1.2.1 void kinect2D3D (int *u_r*, int *v_r*, double *z_r*, CvMat ∗ *p_r*)

It calculates 3D Kinect coordinates *p_r* from 2D coordinates *u_r* and *v_r* and depth value *z_r*.

Parameters

u_r	Coordinate x of Kinect point.
v_r	Coordinate y of Kinect point.
z_r	Depth coordinate of (*u_r* , *v_r*) point.
p_r	3D coordinate in Kinect reference system.

1.1.2.2 void kinect3Dnikon3D (CvMat * *p_r*, CvMat * *p_l*)

It calculates 3D Nikon coordinates *p_l* from 3D Kinect coordinates *p_r*.

Parameters

p_r	3D coordinates in Kinect reference system.
p_l	3D coordinates in Nikon reference system.

1.1.2.3 void nikon3D2D (CvMat * *p_l*, CvMat * *leftPoint*)

It calculates 2D Nikon coordinates *leftPoint* from 3D Nikon coordinates *p_l*.

Parameters

p_l	3D coordinates in Nikon reference system.
leftPoint	2D coordinates in Nikon reference system.

1.1.2.4 bool pointInTriangle (CvMat * *point*, CvMat * *a*, CvMat * *b*, CvMat * *c*)

Check if *point* belongs to *a b c* triangle.

Parameters

point	3D coordinates of the point that has to be checked.
a	3D coordinates of first triangle vertex.
b	3D coordinates of second triangle vertex.
c	3D coordinates of third triangle vertex.

Returns

Return *true* if *point* belongs to *a b c* triangle, *false* otherwhise.

1.2 printInfo.h File Reference

A support printing library for OpenCV objects.

Functions

- void printImageInfo (IplImage *img, string filename)

Print to filename.txt some img parameters.

- void printDoubleCvMat (CvMat ∗mat)

  Print to std::cout some parameters of a double mat.

- void printIntCvMat (CvMat ∗mat)

  Print to std::cout some parameters of an int mat.

1.2.1 Detailed Description

A support printing library for OpenCV objects. Methods implemented can print to file or std:cout IplImage and CvMat parameters. See method documentation for more detailed descrpition.

1.2.2 Function Documentation

1.2.2.1 void printDoubleCvMat (CvMat ∗ *mat*)

Print to *std::cout* some parameters of a *double mat*.

Parameters

mat	CvMat object for which are printed the *double* parameters.

1.2.2.2 void printImageInfo (IplImage ∗ *img*, string *filename*)

Print to filename.txt some *img* parameters.

Parameters

img	Pointer to IplImage object.
filename	Name of the txt saved.

1.2.2.3 void printIntCvMat (CvMat ∗ *mat*)

Print to *std::cout* some parameters of an *int mat*.

Parameters

mat	CvMat object for which are printed the *int* parameters.

1.3 RadialSmoothing.h File Reference

It contains radial smoothing functions.

Functions

- void radialSmoothing (IplImage ∗src, IplImage ∗dst, IplImage ∗mask)

  It calculates in dst a smoothing of values of src where mask is white.
- void radialSmoothingCvMat (CvMat ∗src, CvMat ∗dst, IplImage ∗mask)

  It calculates in dst a smoothing of values of src where mask is white.

1.3.1 Detailed Description

It contains radial smoothing functions. Methods implemented can calculate RadialSmoothing operator on both IplImage or CvMat objects.

1.3.2 Function Documentation

1.3.2.1 void radialSmoothing (IplImage ∗ *src*, IplImage ∗ *dst*, IplImage ∗ *mask*)

It calculates in *dst* a smoothing of values of *src* where *mask* is white.

Parameters

src	Pointer to IplImage where values are taken. It must be 1 channel grayscale IPL_DEPTH_8U.
dst	Pointer to IplImage where smoothed pixels are saved. It must be 1 channel grayscale IPL_DEPTH_8U.
mask	Pointer to boolean IplImage. White pixels are holes. It must be 1 channel grayscale IPL_DEPTH_8U.

1.3.2.2 void radialSmoothingCvMat (CvMat ∗ *src*, CvMat ∗ *dst*, IplImage ∗ *mask*)

It calculates in *dst* a smoothing of values of *src* where *mask* is white.

Parameters

src	Pointer to CvMat where values are taken. It must be of type CV_32FC1.
dst	Pointer to CvMat where smoothed pixels are saved. It must be of type CV_32FC1.
mask	Pointer to boolean IplImage. White pixels are holes. It must be 1 channel grayscale IPL_DEPTH_8U.

1.4 TrilateralFiltering.h File Reference

It contains all version of Trilateral Filtering methods.

Functions

- void trilateralFiltering (CvMat ∗depthMat, IplImage ∗colorMat, IplImage ∗laplaceMat, int halfKernel, CvMat ∗depthFinal)

  It calculates in depthFinal matrix a trilateral smoothing of values of depthMat using a fixed window with dimension 2∗ halfKernel +1.
- void trilateralFilteringVariable (CvMat ∗depthMat, IplImage ∗colorMat, IplImage ∗laplaceMat, int threshold, CvMat ∗depthFinal)

  It calculates in depthFinal matrix a trilateral smoothing of values of depthMat using a variable window.

1.4.1 Detailed Description

It contains all version of Trilateral Filtering methods. Methods implemented can calculate Trilateral Filtering. There are fixed and variable window versions.

1.4.2 Function Documentation

1.4.2.1 void trilateralFiltering (CvMat ∗ *depthMat,* IplImage ∗ *colorMat,* IplImage ∗ *laplaceMat,* int *halfKernel,* CvMat ∗ *depthFinal*)

It calculates in *depthFinal* matrix a trilateral smoothing of values of *depthMat* using a fixed window with dimension 2∗ *halfKernel* +1.

Parameters

depthMat	Pointer to CvMat where good pixel values are taken to calculate the tern of weights. It must be CV_32FC1.
colorMat	Pointer to IplImage where CIE-Lab color information are taken. It must be in CIE-Lab color space, 3 channels IPL_DEPTH_8U.
laplaceMat	Pointer to CvMat where gradient value are taken. It must be 1 channel IPL_DEPTH_32F.
halfkernel	The dimension of the kernel are (2∗ *halfkernel* +1) ∗ (2∗ *halfkernel* +1).
depthFinal	Pointer to CvMat where computed values of trilateral filter is saved. It must be CV_32FC1.

1.4.2.2 void trilateralFilteringVariable (CvMat ∗ *depthMat,* IplImage ∗ *colorMat,* IplImage ∗ *laplaceMat,* int *threshold,* CvMat ∗ *depthFinal*)

It calculates in *depthFinal* matrix a trilateral smoothing of values of *depthMat* using a variable window.

Parameters

depthMat	Pointer to CvMat where good pixel values are taken to calculate the tern of weights. It must be CV_32FC1.
colorMat	Pointer to IplImage where CIE-Lab color information are taken. It must be in CIE-Lab color space, 3 channels IPL_DEPTH_8U.

laplaceMat	Pointer to CvMat where gradient value are taken. It must be 1 channel IPL_-DEPTH_32F.
threshold	Minimum number of good pixels in the trilateral filter window.
depthFinal	Pointer to CvMat where computed values of trilateral filter is saved. It must be CV_32FC1.

Thanks

Before everything I would like to thank Prof. Zanuttigh, Prof. Cortelazzo and Ing. Carlo Dal Mutto, for the knowledge that they shared with me, and for the opportunity to work in Computer Vision field, one of the most fascinating of all the information engineering. In particular I thank Carlo, for the assistance, for the help and all the precious hints during the preparation of this thesis in LTTM laboratory.
A big thank to all my family that sustained me during the university years, and a big thank also to my friends for the physical and psychological help outside the walls of the department, and inside the rectangle of football training.
Last but not least important, thanks to Valentina for putting up with me during my exams and for her Love.

Bibliography

[1] Libfreenect Software Library. http://openkinect.org.

[2] Mesa Imaging AG. "http://www.mesa-imaging.ch".

[3] Microsoft Kinect. http://www.xbox.com/it-IT/kinect.

[4] PrimeSense. http://www.primesense.com/.

[5] RGBDemo software. http://nicolas.burrus.name/index.php/Research/Kinect.

[6] Jean-Yves Bouguet. Matlab Calibration Toolbox. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[7] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media, 2008. http://opencv.willowgarage.com/wiki/.

[8] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 8:679–698, 1986.

[9] Andrea Fusiello. Visione computazionale. Appunti delle lezioni, 2009.

[10] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3rd edition, 2007.

[11] Timo Kahlmann and Hilmar Ingensand. Calibration and development for increased accuracy of 3D range imaging cameras. *Journal of Applied Geodesy*, 2, 2008.

[12] J. Kopf, M. F. Coehn, D. Lischinski, and M. Uyttendaele. Joint Bilateral Upsampling. *ACM Transactions on Graphics*, 26(3), 2007.

[13] Robert Lange. 3D Time-of-flight Distance Measurement with Custom Solid-state Image Sensors in CMOS/CCD–technology, 2000.

[14] Carlo Dal Mutto. Ricostruzione 3D tramite fusione di dati stereo e ToF. Univesity of Padova, 2009.

[15] PrimeSense. Depth mapping using multi-beam, 2010.

[16] Richard Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2010.

[17] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the 1998 IEEE International Conference on Computer Vision.*

[18] Jana Kosecka S. Shankar Sastry Yi Ma, Stefano Soatto. *An Invitation to 3D Vision.* Springer, 2003.