Faculty of Civil and Industrial Engineering
Department of Civil and Environmental Engineering
Ph.D. course in Infrastructures and Transport
Curriculum in Infrastructures and Geomatics
XXIX cycle - 2013-2016

3D modeling by low-cost range cameras: methods and potentialities

Candidate: Roberta Ravanelli
Matricola: 1208903

Advisor: Mattia Giovanni Crespi
Co-advisor: Andrea Nascetti

Contents

List of Figures

List of Tables

Abstract

Nowadays the demand of 3D models for the documentation and visualization of objects and environments is continually increasing. However, the traditional 3D modeling techniques and systems (*i.e.* photogrammetry and laser scanners) can be very expensive and/or onerous, as they often need qualified technicians and specific post-processing phases. Thus, it is important to find new instruments, able to provide low-cost 3D data in real time and in a user-friendly way.

Range cameras seem one of the most promising tools to achieve this goal: they are low-cost 3D scanners, able to easily collect dense point clouds at high frame rate, in a short range (few meters) from the imaged objects.

Such sensors, though, still remain a relatively new 3D measurement technology, not yet exhaustively studied. Thus it is essential to assess the metric quality of the depth data retrieved by these devices.

This thesis is precisely included in this background: the aim is to evaluate the potentialities of range cameras for geomatic applications and to provide useful indications for their practical use.

Therefore the three most popular and/or promising low-cost range cameras, namely the Microsoft Kinect v1, the Micorsoft Kinect v2 and the Occipital Structure Sensor, were firstly characterized from a geomatic point of view in order to assess the metric quality of the depth data retrieved by them.

These investigations showed that such sensors present a depth precision and a depth accuracy in the range of some millimeters to few centimeters, depending both on the operational principle adopted by the single device (Structured Light or Time of Flight) and on the depth itself.

On this basis, two different models were identified for precision and accuracy vs. depth: parabolic for the Structured Light (the Kinect v1 and the Structure Sensor) and linear for Time of Flight (the Kinect v2) sensors, respectively. Then the effectiveness of such accuracy models was demonstrated to be globally compliant with the found precision models for all of the three sensors.

Furthermore, the proposed calibration model was validated for the Structure Sensor: with calibration, the overall RMSE, decreased from 27 to 16 mm.

Finally four case studies were carried out in order to evaluate:

- the performances of the Kinect v2 sensor for monitoring oscillatory motions (relevant for structural and/or industrial monitoring), demonstrating a good ability

of the system to detect movements and displacements;

- the integration feasibility of Kinect v2 with a classical stereo system, highlighting the need of an integration of range cameras into 3D classical photogrammetric systems especially to overpass limitations due to acquisition completeness;
- the potentialities of the Structure Sensor for the 3D surveying of indoor environments, showing a more than sufficient accuracy for most applications;
- the potentialities of the Structure Sensor to document archaeological small finds, where metric accuracy seems to be rather good while textured models shows some misalignments.

In conclusion, although the experimental results demonstrated that range cameras have the capability to give good and encouraging results, the performances of traditional 3D modeling techniques in terms of accuracy and precision are still superior and must be preferred when the accuracy requirements are restrictive.

But for a very wide and continuously increasing range of applications, when the required accuracy can be at the level from few millimeters (very close-range) to few centimeters, then range cameras can be a valuable alternative, especially when non expert users are involved. Furthermore, the technology on which these sensors are based is continually evolving, driven also by the new generation of AR/VR

reality kits, and certainly also their geometric performances will soon improve.

Chapter 1

Introduction

Today 3D modeling is a subject of great interest in very different fields such as industry, robotics, medicine, cultural heritage, civil engineering, architecture, where the demand of 3D models for the documentation and visualization of objects and environments is continually increasing. However, the traditional 3D modeling techniques and systems (*i.e.* photogrammetry and laser scanners) can be very expensive and/or onerous, as they often need qualified technicians and specific post-processing phases.

Thus, it is important to find new instruments, able to provide low-cost 3D data in real time and in a user-friendly way, at least for some applications, which may also be suitable to be developed in the frame of volunteered geographic information (VGI) generation, whereas a reasonable (if not low) cost is an important feature.

Range cameras seem one of the most promising tools to achieve this goal: they are active imaging sensors, low-cost and easy to use, able to natively measure the distances of several points at high frame

rate (30 - 60 Hz).

Thanks to these characteristics, nowadays this technology can play an important role in close-range 3D modeling: range cameras can be used as 3D scanners to easily collect dense point clouds practically in real time. In addition such sensors are continually evolving and they will be soon integrated in consumer grade smart devices, allowing their use along with other sensors and becoming available to a wider and not expert audience.

Range cameras, though, still remain a relatively new 3D measurement instrument, not yet exhaustively studied. Thus it is essential to assess the metric quality of the depth data retrieved by these devices.

This work is precisely included in this background: the aim is to evaluate the potentialities of range cameras for geomatic applications and to provide useful indications for their practical use.

In particular this thesis proposes specific models to represent random and systematic errors of depth measurements (dependent on the operational principle adopted by the single sensor and on the distance from the captured object) for the considered range cameras, thus describing their precision and accuracy, and proves the effectiveness of such models for the calibration of these sensors. Furthermore some investigations about the registration process of depth and color images are also described. Finally some case studies are presented in order to provide some insights into the practical usage of range cameras in different fields.

This thesis is therefore structured in the following chapters:

- **chapter 2**: a brief overview of the most used techniques and systems for close-range 3D modeling is given; basics and func-

tioning of range cameras are recalled, focusing on both Time of Flight (ToF) and Structured Light (SL) operational principles;

- **chapter 3**: the main features of the investigated sensors are firstly introduced; then their geomatic characterization is described, together with the models identified to represent their random and systematic errors, which are proven adequate for calibration; some investigations regarding the registration process of depth and color images are also presented;

- **chapter 4**: four case studies illustrating the practical use of range cameras are discussed;

- **chapter 5**: some conclusions are outlined, together with potential prospects for future investigations;

- **appendix A**: the libraries available to retrieve the depth data from range cameras are described; further algorithms and software for 3D model reconstruction are also presented;

- **appendix B**: the applications implemented to carry out the research are shortly illustrated, after a brief overview of the IT facilities used.

Chapter 2

Close-range 3D modeling and range cameras

Three dimensional (3D) modeling is an intensive and durable research problem in 3D graphics, computer vision and photogrammetry.

It consists in the complete process that, starting from data collection, generates a three dimensional mathematical representation of the geometry (shape and dimensions) of an object/environment. The obtained numerical description of the object is then stored in a digital form and can be interactively visualized as 3D virtual model on the screen of a computer.

Today many tools are available to produce 3D models of objects and scenes. Hereafter a brief overview of the most used technique and systems for close-range applications is given. The available literature is wide and continuously increasing, and the interested reader can refer to it for 3D modeling methods and techniques already existing before/other than range camera.

The most general classification of 3D object measurement and reconstruction techniques can be divided into *contact* methods (for example, using coordinate measuring machines, callipers, rulers and/or bearings) and *non-contact* methods (X-ray, SAR, photogrammetry, laser scanning). Nowadays the generation of a 3D model is mainly achieved using non-contact *optical* systems based on electromagnetic waves, in particular using *passive* or *active* sensors (Fig. 2) [91].

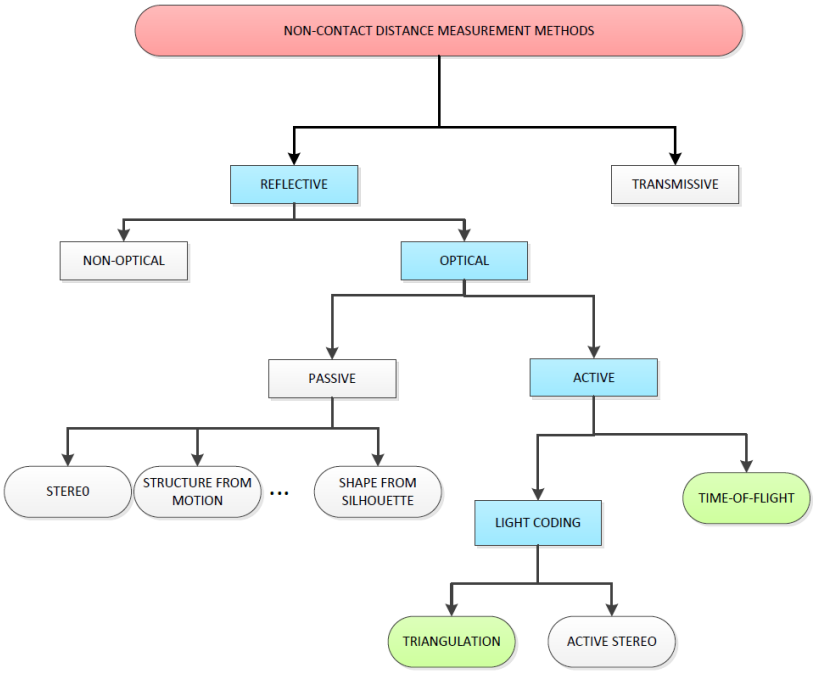


Fig. 2.1: Three-dimensional acquisition systems for object measurement using non-contact methods based on light waves [34].

Passive range sensing refers to 3D distance measurement by way of radiation (typically, but not necessarily, in the visible spectrum) already present in the scene; photogrammetry is a classical example

of this family of methods. Active sensing refers, instead, to 3D distance measurement obtained by projecting in the scene some form of radiation as made, for instance, by range cameras and laser scanners [34].

Active sensors directly provide depth data containing the 3D coordinates necessary for the mesh generation phase. Passive sensors provide images that need further processing to derive the 3D object coordinates [91].

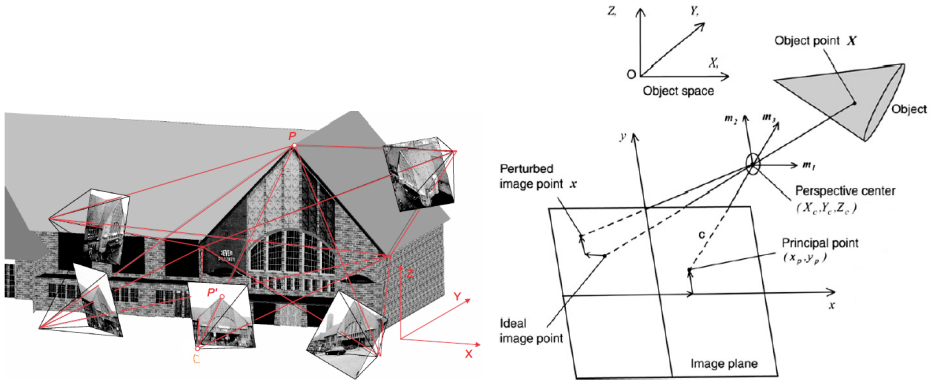
2.1 Close-range 3D modeling before range cameras

3D modeling of close-range objects is traditionally achieved with a standard topographic/photogrammetric survey or, more recently, with laser scanners. Today range cameras are a relative new technology but they can represent a very promising alternative for many kinds of close-range surveying and 3D modeling applications. In the following sections a brief description of the photogrammetric technique and the laser scanner systems is given, with a particular focus on their most challenging limits. As regards range cameras, their functioning will be illustrated in greater detail in Sec. 2.2. For now it sufficient to say that, to a certain extent, they are very similar to laser scanners, less accurate and capable of acquiring less points, but at the same time cheaper and easier to use.

2.1.1 Photogrammetry

Photogrammetry is a survey technique able to derive the shape and the position of physical objects and environments through the measurement and the interpretation of photo images. In particular, for close-range surveys, the so-called *terrestrial photogrammetry* technique is adopted, where the photo images are captured with cameras located on the earth surface. They may be handheld, mounted on tripods, or suspended from towers or other specially designed mounts.

Specifically, to extract metric data from two-dimensional (2D) images, photogrammetry adopts central projection imaging as its fundamental mathematical model (Fig. 2.2a) [70].



(a) (b)

Fig. 2.2: (a) Principle of photogrammetric measurement [70]; (b) perspective projection from the 3D object space to the 2D image plane [69].

Shape and position of an object are thus determined by reconstructing bundles of rays in which, for each camera, every image point P', together with the corresponding perspective center C, defines the

spatial direction of the ray to the corresponding object point P. Provided the imaging geometry within the camera and the location of the imaging system in object space are known, then every image ray can be defined in 3D object space. From the intersection of at least two corresponding (homologous) spatially separated image rays, an object point can be located in three dimensions. In stereo-photogrammetry two images are used to achieve this. In multi-image photogrammetry the number of images involved is, in principle, unlimited [70].

The interior orientation parameters describe the internal geometric model of a camera, represented as a pinhole camera for which the most important reference location is the perspective center C, through which all image rays pass. The interior orientation parameters define the position of the perspective center C relative to the image coordinate system, namely the reference system fixed in the camera, as well as departures from the ideal central projection (image distortion) [70]. They include the camera principal distance c and the photogrammetric principal-point location (x_p, y_p). The principal distance, which equals the camera focal length for a camera focused at infinity, is the perpendicular distance from the perspective center to the image plane, whereas the photogrammetric principal-point is where a perpendicular line from the perspective center intersects the image plane [69]. Due to lens distortion and refraction of the medium where electromagnetic waves propagate from the imaged object to the camera, however, perturbation to the imaging process leads to departure from collinearity that can be represented by the shifts δx and δy of the image point from its 'ideal' position on the image plane [69].

The exterior orientation parameters specify the spatial position and orientation of the camera in the global coordinate system in which the

object is placed. The exterior orientation is described by the coordinates of the perspective center (X_C, Y_C, Z_C) in the global system and the three suitably defined angles (ω, ϕ, k), expressing the rotation of the image coordinate system with respect to the global system. The exterior orientation parameters are calculated indirectly, after measuring image coordinates of well identified object points,namely the Ground Control Points (GCP), with fixed and known global coordinates [70].

In particular the collinearity equations provide the perspective projection relationship between the 3D coordinates (X, Y, Z) in the object space and the corresponding 2D coordinates in the image plane (x, y) [69]:

$$x - x_p - \delta x = -c\frac{r_{11}(X - X_c) + r_{12}(Y - Yc) + r_{13}(Z - Z_c)}{r_{31}(X - X_c) + r_{32}(Y - Y_c) + r_{33}(Z - Z_c)} \quad (2.1a)$$

$$y - y_p - \delta y = -c\frac{r_{21}(X - X_c) + r_{22}(Y - Yc) + r_{23}(Z - Z_c)}{r_{31}(X - X_c) + r_{32}(Y - Y_c) + r_{33}(Z - Z_c)} \quad (2.1b)$$

where $r_{ij}(i, j = 1, 2, 3)$ are the elements of the rotation matrix $R = R(k)R(\phi)R(\omega) = [r_{ij}]$ that are functions of the Euler orientation angles (ω, ϕ, k).

Therefore the collinearity equations allow to compute the 3D coordinates of the object captured in the images, but only after having determined the camera interior and exterior orientation parameters, as well as the lens distortion parameters.

The classical close-range photogrammetric workflow thus consists of several steps, including: image acquisition (images of the same view have to be captured at least from two different points of view), camera calibration (computation of the interior and distortion parameters) and orientation (computation of the exterior parameters), image point

measurements (matching: identification of the homologous points), 3D point cloud generation (computation of the 3D coordinates of the object), surface generation and texturing. Today reliable software packages are available allowing the management of the complete scene modeling process. After the tie point measurement, that can be manual, semi-automated or automated, and bundle adjustment phases, these software allow to calibrate and orientate the sensor, to compute the 3D object point coordinates, as well as to generate the wireframe or textured 3D models [92].

Anyway recovering a complete, detailed, accurate and realistic 3D model from images is still a difficult task, in particular for large and complex sites and in case of uncalibrated or widely separated images are used. The wrong recovery of the parameters may indeed lead to inaccurate and deformed results [91]. Finally the processing time can be very long, especially when the number of the captured images is noticeable.

For all these reasons, photogrammetry remains a specialized discipline that requires extensive knowledge and great experience in order to obtain high-quality and complete 3D models; moreover, it is also shows limits in performing real-time applications.

2.1.2 Laser scanners

Recently, advances in laser scanner technology have created much interest in the utilization of terrestrial laser scanning (TLS) for close-range 3D modeling [87]. Laser scanners are indeed very expensive tools (typically between 50000 € and 200000 €), but able to accurately measure the positions of millions of 3D points (mm level accuracy) with a very

high point density in a short time (up to 1 million points per second). These features make them a valuable alternative or complementary technique for classical topographical measurements based on total station or digital photogrammetry [103]. Different commercial solutions are actually available on the market, based on triangulation, time-of-flight, continuous wave or reflectivity measurements [92], [22], [24].

Although laser scanners can obtain high levels of geometric detail with high degree of accuracy, intensive work, experience and time are still needed for data acquisition and processing [87, 24]. Indeed, to obtain a complete 3D model, laser scanners must capture the object in different positions and a crucial registration process of the acquisitions from different points is thus required in order to generate the entire 3D model. Furthermore they are bulky and heavy tools, features that limit their flexibility of use. Meanwhile, laser scanners cannot by themselves acquire textural information. Integration and registration with digital camera images is necessary [24, 87]. This increases not only the cost, but also the complexity of the data processing pipeline [87].

2.2 Range cameras

Range cameras are active imaging sensors able to natively measure the distances of several points at high frame rate (30 - 60 Hz).

At every acquisition, they produce the so called *depth map* of the scene, an image in which each pixel contains its own distance from a specific reference, normally associated to the sensor itself. Starting from this depth map, range cameras generate a dense *point cloud* of the environment scanned, a collection of an elevate number of 3D

coordinates in a given reference system.

Thus it is straightforward to understand how this technology plays an increasing role in close-range 3D modeling: these devices can be used as 3D scanners to easily capture the 3D geometry (shape and dimensions in metric units) of a scene practically in real time.

In particular the depth sensor is the heart of the technology, since it is designed specifically to capture the depth data. Although its function is equivalent for all the range cameras, it is worth noting that the underlying working principle can be different, depending on the technique adopted by the specific range camera.

Indeed, to generate the depth map, range cameras can use two different operational principles: it is possible to distinguish between the Time of Flight (ToF) range cameras and the Structured Light (SL) range cameras.

ToF range cameras generate the depth map by measuring the time of flight taken by an electromagnetic wave to travel from the sensor itself to the object and back. For this family of devices the depth sensor consists of a matricial collection of emitters and receivers.

Differently, SL range cameras emit a bi-dimensional light pattern directly on the surface of the object to be measured and generate the depth map by evaluating the deformation between the pattern emitted and the one back-projected by the object itself. In this case the depth sensor consists of a projector and a frequency-matched camera.

Hereafter these two working principles are just shortly recalled, and the interested reader can find more details within [34], [50], [52] and [93]. In particular, for each of the two categories, the principal error sources of the most used sensors, namely the continuous wave ToF range cameras and the infrared SL range cameras, are described.

2.2.1 Time of Flight range cameras

ToF range cameras illuminate the environment with a modulated radiation (e.g. light) and observe the component reflected by the scene onto the receiver [56]. Typically, the illumination unit is a solid-state laser or a LED operating in the near-infrared (NIR) range (about 850 nm), invisible to the human eyes, and the receiver is an imaging sensor responding to the same spectrum, designed to convert the photonic energy to electrical current [67].

By measuring the flight time of the signal, ToF range cameras estimate the distance d to the objects of the scene framed through the following well known equation:

$$d = \frac{c\,\tau}{2} \tag{2.2}$$

where c denotes the speed of light and τ is the time of flight.

In particular, at least two different methods can be adopted in order to evaluate the time of flight: the direct method and the indirect method (see Fig. 2.3).

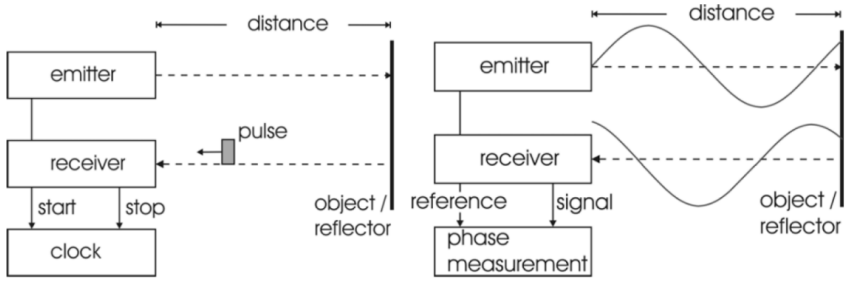


Fig. 2.3: Time of flight methods for distance measurement. The direct and the indirect method [56].

In the first case, the light emitted is modulated by a single *pulse* and the difference between the departure and arrival times of the pulsed signal is *directly* measured by high precision clock circuits.

In the latter, the light emitted is modulated by a *continuous-wave* source and the time of flight is measured *indirectly* by evaluating the phase shift between the reference signal generated by the emitter and the one reflected by the target and received by the detector.

It is therefore possible to identify two families of ToF range cameras: the pulsed ToF range cameras and the continuous wave ToF range cameras. The former ones, based on the direct ToF measurement, can measure distances up to 1500 m, whereas the latter ones, based on the phase shift measurement, usually show a more limited working range (up to 10 m), but they have higher accuracy and thus they are more suitable for close-range 3D modeling applications [83].

2.2.1.1 Pulsed ToF range cameras

Pulsed modulation can be achieved by integrating photoelectrons from the light of the reflected signal, or by starting a fast counter at the first detection of the reflection [67].

In the first approach, the light source illuminates the scene for a short period (Δt), and the reflected energy is sampled at every pixel in parallel, using two out-of-phase windows, C_1 and C_2, with the same Δt. The electrical charges accumulated during these samples, Q_1 and Q_2, are measured and used to compute the distance through the following formula [67] (see Fig. 2.4):

$$d = \frac{1}{2}\, c\, \Delta t \left(\frac{Q_2}{Q_1 + Q_2} \right) \tag{2.3}$$

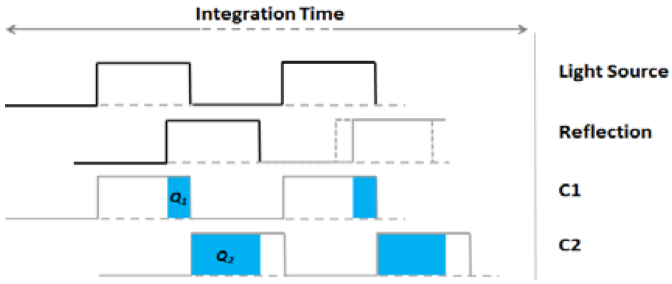


Fig. 2.4: Pulsed method by integrating photoelectrons from the reflected light [67].

Concerning the time counting method, it is more complex from an hardware point of view, as it requires fast electronics and a high precision clock to derive an accurate distance measurement using the following relation:

$$d = \frac{c}{2} \ (t_{STOP} - t_{START}) \tag{2.4}$$

where c is the speed of light, t_{START} is the starting time of the light pulse synchronization signal and t_{STOP} is the arrival time of the reflected pulse in the detector. This equation clearly shows that the distance accuracy depends on the accuracy of the time of flight measurement: in order to obtain an accuracy of 1 mm, it is necessary to measure a pulse of 6.6 picoseconds in duration. Single-photon avalanche diodes (SPADs) are one of the few detectors capable of capturing individual photons with the level of accuracy needed, even if it is nearly impossible to achieve this standard in silicon at room temperature [29].

2.2.1.2 Continuous wave ToF range cameras

Most of the commercially available ToF range cameras, such as the Microsoft Kinect v2, use the phase shift measurement principle and

thus the continuous wave modulation technique.

In particular, the emitter of the continuous wave ToF range cameras illuminates the scene with a NIR optical signal $s_E(t)$ of amplitude A_E modulated by a sinusoid of frequency f_{mod} [34]:

$$s_E(t) = A_E \left(1 + \sin(2\pi f_{mod} t)\right) \tag{2.5}$$

The signal $s_E(t)$ is then reflected by the object surface and travels back towards the sensor receiver, ideally co-positioned with the emitter. The signal round trip together with the non-instantaneous propagation of NIR radiation generate the phase delay $\Delta\varphi$ that intrinsically contains the distance information. The signal received is also shifted in amplitude of an offset B_R due to the presence of additional background radiation at the NIR wavelength of the emitted signal. Furthermore, its amplitude A_R is attenuated because of all the optical losses associated to the reflection such as energy absorption and free-path propagation attenuation [26], [34]. On the basis of all these considerations, the signal reaching the receiver $s_R(t)$ can be expressed as follows (see Fig. 2.5):

$$s_R(t) = A_R \left(1 + \sin(2\pi f_{mod} t + \Delta\varphi)\right) + B_R \tag{2.6}$$

The quantity A_R (from now on denoted with A) is the amplitude of the useful signal. The quantity $A_R + B_R$ (from now on denoted with B) is called intensity or offset, and it is the average of the received signal. According to this notation, Eq. 2.6 can be rewritten as a harmonic function describing a photon flux [34], [99]:

$$s_R(t) = A \sin(2\pi f_{mod} t + \Delta\varphi) + B \tag{2.7}$$

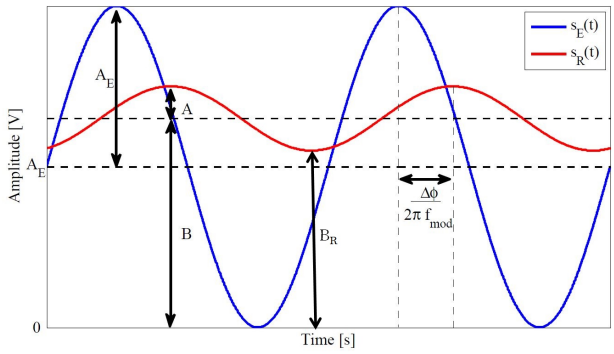


Fig. 2.5: The emitted signal $s_E(t)$ and received signal $s_R(t)$ [34].

Eq. 2.7 shows three unknowns: A, B and $\Delta\varphi$, where A and B are IR radiation amplitudes and therefore measured in volt [V] and $\Delta\varphi$ is a pure number representing a phase value. The most important unknown is the phase shift $\Delta\varphi$ since continuous wave ToF cameras infer distance d from it, whereas A and B are important for signal-to-noise ratio (SNR) considerations [34]. These quantities can be estimated by demodulating the incoming signal through a technique commonly known as *four-bucket* sampling [42] (see Fig. 2.6), according to which the received signal is sampled at four sample points of quarter phase interval of the modulated source frequency [75]:

$$S_{Ri} = S_R(t_i) = A_i \sin(2\pi f_{mod} t_i + \Delta\varphi) + B, \quad t_i = i \cdot \frac{\pi}{2\omega}, \quad i = 0, 1, 2, 3 \tag{2.8}$$

Anyway, since phase lock-in TOF sensors are digital devices, instantaneous measurement or ideal sampling of the received signal is not possible. In practice a measurement is made by integrating over a time period Δt. The sampling does not effect the phase measurement

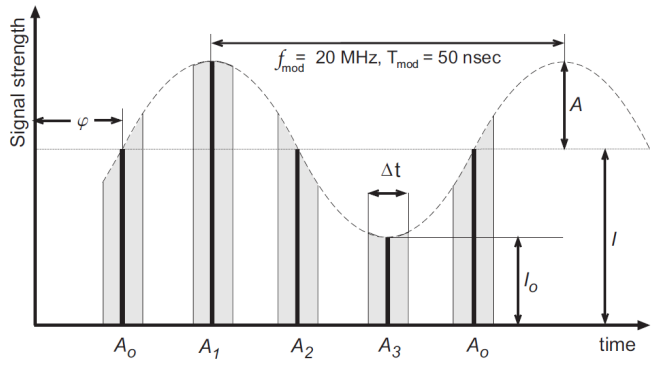


Fig. 2.6: A signal with a modulation frequency f_{mod} of 20Mhz and a background offset I_0 is sampled four times A_0, A_1, A_2, A_3 with a short integration time Δt to calculate amplitude A, phase shift $\Delta\varphi$ and intensity I[75].

as long as the integration time Δt is less than the modulation period of the sampled signal, but, however, it attenuates the amplitude. Therefore each sample corresponds to the integration of the photo-generated charges over a fraction of the modulation period [75].

Out of the four samples A_0, A_1, A_2 and A_3, the phase shift $\Delta\varphi$, the amplitude A and the offset B can be computed as:

$$\Delta\varphi = \arctan\left(\frac{A_3 - A_1}{A_0 - A_2}\right) \tag{2.9}$$

$$A = \frac{\sqrt{(A_3 - A_1)^2 + (A_0 - A_2)^2}}{2} \tag{2.10}$$

$$B = \frac{A_1 + A_2 + A_3 + A_4}{4} \tag{2.11}$$

Another approach adopted by continuous wave range camera to es-

timate the $\Delta\varphi$ is instead based on the on-chip correlation (or mixing) of the incident optical signal s_R, coming from the modulated NIR illumination and reflected by the scene, with a reference signal s_G emitted by the generator, possibly with an internal phase offset τ [61]:

$$C(\tau) = s_R \otimes s_G = \lim_{T\to\infty} \int_{-T/2}^{T/2} s_R(t) \cdot s_G(t+\tau)\, dt \tag{2.12}$$

Also in this case a sinusoidal modulation of constant frequency f_{mod} is used; in particular the s_G signal can be expressed in the following way:

$$s_G(t) = \cos(2\pi f_{mod}\ t) \tag{2.13}$$

With some trigonometric calculus it is therefore possible to explicit the correlation function C as a harmonic function too [61]:

$$C(\tau) = \frac{A}{2}\ \cos(f_{mod}\ \tau + \Delta\varphi) + I \tag{2.14}$$

where the amplitude of the incident optical signal A, the intensity offset I due to the background illumination and the phase shift $\Delta\varphi$ proportional to the object distance are once again the unknowns to be estimated, in this case by demodulating the correlation function C.

Again this can be achieved through the *four-bucket* technique, thus four sequential phase images Q_i (also called correlation images) with different phase offset τ [61] are obtained by sampling the correlation function C four times per modulation period with each sample shifted by 90 degrees [26]:

$$Q_i = C(\tau_i) = C\left(i \cdot \frac{\pi}{2}\right), \quad i = 0, 1, 2, 3 \tag{2.15}$$

The unknowns are computed applying the Eq. 2.9, Eq. 2.10 and Eq. 2.11 to the samples of the correlation function.

Anyway, once the phase shift $\Delta\varphi$ has been computed, the corresponding distance d can be calculated through the following equation:

$$d = \frac{\lambda_{mod}}{2}\frac{\Delta\varphi}{2\pi} = \frac{c}{2f_{mod}}\frac{\Delta\varphi}{2\pi} \tag{2.16}$$

where λ_{mod} is the modulation wavelength, c is the speed of light and the quantity $\frac{c}{2f_{mod}}$ is the ambiguity distance, that is the maximum distance that can be measured without ambiguity. In fact, since this measurement is based on phase, which wraps around every 2π, an ambiguity effect can occur.

To extend the measurable distance, the modulation frequency could be decreased, but at the cost of a reduced resolution. Instead of accepting this compromise, advanced ToF systems deploy multi-frequency techniques to increase the distance without reducing the modulation frequency [67]. Multi-frequency techniques work by using one or more modulation frequencies, as it is standard within the electromagnetic distance measurement for a while [94].

2.2.1.2.1 Error sources

Real continuous wave ToF range cameras are rather more complex than ideal systems. First of all, the theoretically required sinusoidal signal is not achievable in practice, since it is obtained through a low-pass filter on the squared wave-forms emitted by LEDs. Therefore the measured distance shows a systematic error, also called *wiggling effect* [61], which introduces an harmonic distortion which depends on the mea-

sured distance [34]. Secondly, they use standard optics to focus the reflected active light onto the chip. Thus, classical intrinsic calibration is required to compensate effects like shifted optical centers and lateral distortion [61]. In particular, ToF range cameras can be considered as an array of range finders [93] in which each pixel is associated to a finite area of the observed scene. The ideal situation in which a single pixel of the sensor measures exclusively the distance of the corresponding object point is indeed valid only if the scanned area has an almost constant reflectivity, that is absence of *reflectivity discontinuity*. If not, or to make matters worse whether the area related to the considered point crosses not only a reflectivity discontinuity but also a *depth discontinuity*, the resulting depth estimate presents severe artefacts [34]. In this case the mixing process results in a super imposed signal caused by the light reflected from different depths, leading to wrong distance values [98]. The pixels associated to such depth estimates are commonly called *flying pixels* and can be regarded as outliers in the depth measurement [34].

In addition to this problem, there is also the multipath propagation related to the *scattering*, i.e. reflection in multiple directions. Although the incident direction is the more likely path for the back reflected ray, the presence of other rays cannot be neglected, especially the specular ray to the incident one. In fact the light may additionally travel indirect paths, i.e. being scattered by highly reflective objects in the scene or within the lens systems or the housing of the camera itself [34]. These multiple responses of the active light are superimposed in each pixel, leading to an altered signal and thus a wrong distance. In the context of computer graphics this effect is known as global illumination [98]. Since multi-path is a scene-dependent error, it is very

hard to model [34].

Furthermore ToF range cameras need a pre-heating time before they can acquire stationary depth measurements. This kind of sensors are indeed made of semiconductor materials, highly responsive to temperature changes. In the first minutes after the start-up, the device internal temperature can increase (or decrease, if cooling is available) and then should eventually stabilize [83]. Thus the retrieved distances vary during the warm-up time and the sensor may take several minutes before measuring stable depth values [30], [83] [98], [63].

Finally ToF cameras can suffer from ambient background light that can lead to over saturation in case of too long exposure times in relation to the object distance and/or reflectivity [98]. Anyway most of the current ToF range cameras support Suppression of Background Intensity (SBI), thanks to which the intensity mainly reflects the incident active light and outdoor applications are facilitated [61]. In fact, since this kind of devices are active sensors using the infrared spectrum, all the infrared light sources, such as sun light or other active IR devices, can potentially interfere with their correct functioning. For the same reason these sensors can show difficulties to reconstruct surfaces that do not perfectly reflect the incident NIR light, as, for example dark, shiny or transparent objects.

2.2.2 Structured Light range cameras

SL range cameras are active triangulation systems that exploit the structured light technique to capture the 3D geometry of the scene.

A structured light device is similar to a classical stereoscopic sensor with a camera replaced by a light source, commonly a laser or a special

slide projector, as shown in Fig. 2.7.

In its simplest implementation, now almost obsolete, the projector is a laser beam projecting a single dot onto the scene. Since only one point is projected, the correspondence is direct, but, however, scanning along both axes is required [41].

A second solution consists of using a projector that emits a stripe (plane) of light and a camera placed at an angle with respect to the projector. At each point in time, the camera obtains 3D positions for points along a 2D contour traced out on the object by the plane of light. In order to obtain a full depth map, it is necessary to sweep the stripe along the surface (as is done by many commercial single-stripe laser range) by physically moving the projector. Correspondence solving is quite simple: the 3D positions of points on the object are determined from the intersection between the camera ray and the plane of light produced by the illumination source, this ensures a single and unambiguous matching [95].

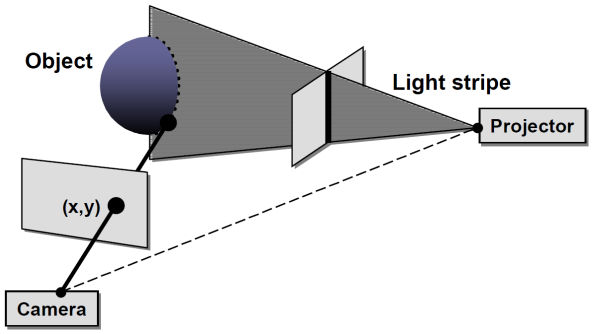


Fig. 2.7: Schematic layout of a single-camera, single-stripe-source triangulation system [95].

In order to avoid a time consuming mechanical scanning, the last

and most used solution consists in projecting a bidimensional pattern, such as a multi-stripe pattern, a grid or multiple dots. For this kind of patterns, a correspondence problem occurs, but it is solved by coding the pattern, in a way that each token of light is easily distinguishable from the others [41]. In this way the 3D position of each surface point is identified by searching correspondences between points in the camera image and points in the a priori known pattern emitted by the projector (pattern decoding). Therefore there is no need for geometrical constraints, as instead it happens in passive triangulation systems, where the problem is solved by searching the same object point along epipolar lines (geometric constraint) of two (or more) images [46] (see Fig. 2.8).

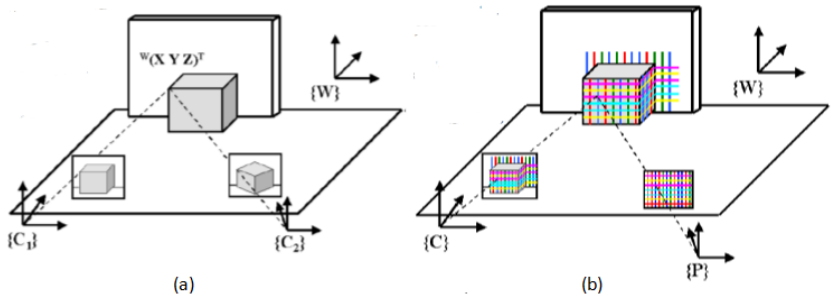


Fig. 2.8: Comparison between a passive triangulation system (a) and the structured light system (b) [46].

In general the term structured light mainly refers to the third method, which is the one implemented by SL range cameras. It extracts the 3D data by evaluating the distortion of the 2D spatially varying intensity pattern generated by the projector and captured by an imaging sensor. If the scene is a planar surface and thus without any 3D surface variation, the pattern shown in the acquired image is

very similar to that of the projected structured-light pattern. However, when the scene surface is not planar, the geometric shape of the surface distorts the projected structured-light pattern as seen from the camera [45].

In the specific case of SL range cameras, an infrared speckle pattern is used: an infrared laser projector emits a pattern of thousands of invisible infrared dots on the surface of the object/s to be modelled and a frequency-matched infrared camera records how the environment deforms the pattern, thereby obtaining the 3D geometry (shape and dimensions in metric units) of the objects.

2.2.2.1 Active triangulation

Triangulation is one of the oldest and most used ranging technique. In the form of *stereo vision* along with the *depth from focus* system, it is indeed at the core of human depth perception. Triangulation is based on a geometrical approach: the point to be measured is a vertex of a triangle whose two remaining vertexes are known parts of the measurement system. For passive triangulation, the two remaining points of the triangle are two imaging devices, in case of active systems they consist, instead, of a light source and a camera. In particular, the target distance can be determined by measuring the angles of the triangle or the triangulation base [64]. From this moment on, the discussion will be focused on the active triangulation devices.

In the first case, the target distance d can be computed by measuring the viewing angles α and θ with respect to the baseline b between

the light projector and the camera (Fig. 2.9) [45]:

$$d = b\frac{\sin(\theta)}{\sin(\alpha + \theta)} \tag{2.17}$$

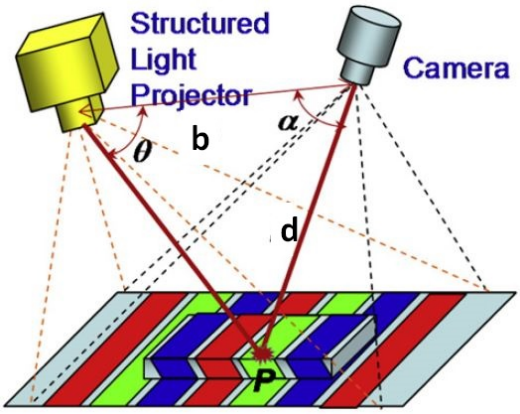


Fig. 2.9: Triangulation by angles measurement [45].

Rather than measuring angles directly, the second method exploits the similarity of two triangles, as Fig. 2.10 shows. The first triangle is constituted by the target point, the projector and the camera whereas the second one is fully defined by the optical axis of the imaging device, the focal length f of the system and the detector image plane [64]

By knowing the baseline b between the light source and the imaging device, the depth z of the target can be determined as follows:

$$z = \frac{f\ b}{\delta} \tag{2.18}$$

where the disparity δ denotes the shift between the horizontal position of the object point pixel in the camera image plane and in the emitted pattern, respectively. In other words the projection/acquisition process

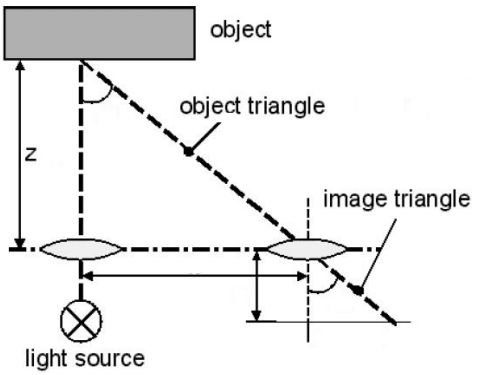


Fig. 2.10: Triangulation by disparity measurement [64].

introduces an horizontal shift δ proportional to the inverse of the depth z. Disparity shift δ is the most important quantity and since it carries the 3D geometry information relative to the considered scene, it needs to be carefully estimated [34] by finding accurately the tie points (also called conjugate points), namely the projection of the same object point in the camera image plane and the projector pattern plane.

In particular SL range cameras adopt a procedure called *matricial active triangulation*, which computes the disparity for every pixel of the depth map. In this case the major difficulty is keeping the correspondence problem as simple as for a single point. This issue can be handled by designing the pattern emitted by the projector through the light coding strategies described in the next paragraph.

2.2.2.2 Structured Light coding strategies

To identify conjugate points in order to apply triangulation, each pixel of the emitted pattern needs to be associated to a *code-word*, i.e. a specific local configuration of the projected pattern [34]. In this way it

is possible to establish a direct mapping from the code-words detected in the camera image to the corresponding coordinates of the pixel in the projected pattern. However, what really is important it is to select code-words that are highly and effectively decodable. This is why the pattern design is fundamental to the correct operation of structured light devices.

2.2.2.2.1 Code-Words

The patterns of structured light sensors are specially designed to assign code-words to pattern pixels. The codewords are numbers, which are mapped in the pattern by using grey levels, color or geometrical representations. The larger the number of points that must be coded, the larger the codewords are and, therefore, the mapping of such codewords to a pattern is more difficult [97]. Furthermore the more the code-words are different the more robust is the coding against disturbances and self-interferences [34]. Anyway, given a certain cardinality of possible code-words, the smaller is the number of used code-words the greater is the difference between them.

A code-words alphabet can be implemented through a light projector considering that it can produce $n_p = 2^{bit(s)}$ different illumination values called *pattern primitives* [34]. For example, a binary black /white projector has $n_p = 2^1$, whereas for a 8-bit gray-scale or RGB projector n_p is respectively 2^8 and 2^{24}. Then, this alphabet is used to build proper local distribution patterns – namely code-words – for each pixel p_P of the projector using the illumination values of the pixels in a window n_W around the given pixel. Therefore, the number of possible pattern configurations is n_p^{nW} and from this set N configurations need

to be chosen as code-words [34].

The pattern projected onto the scene and then acquired by the camera derives from the fusion of code-words relative to all the pixels of the projected pattern. However, there are several factors transforming it and introducing artefacts which must be taken into consideration [34]. For example *perspective* distortion could map not accurately neighbouring pixels because of difference in depth z. Secondly, there are *color distribution* and *reflectivity* distortions. The appearance of the pixels on the camera depends on the scene reflectance, which is strongly related to color properties. Strong absorption because of low reflectance can distort high intensity pixels making them appear much darker. This is a very important issue, since it might completely change the projected code-words [34]. Furthermore, also *external illumination* influences the acquired color because the light – artificial one or sunlight – falling on the scene surfaces acts as a noise source added to the information of the signal emitted [34]. In addition, the 3D geometry of the scene with its *occlusions* might determine the possibility that some pixels are not acquired by the camera, causing a not biunivocal association. Those pixels have to be identified and discarded in order to avoid wrong correspondences. Lastly, projector and camera *non-idealities* and noise should be taken into account.

Now it is more evident as the above listed transformations or distortions can do an acquired code-word very different from the projected one and, to make matters worse, due to occlusions some pixels of the acquired image may not correspond to any pixel of the projected pattern [34]. It is luckily possible to mitigate these potentially disruptive effects during the correspondences estimation process. To do this two decisions are fundamental:

- what code-word assign to each pixel p_P of the projected pattern, namely the pattern to be projected on a window centred at p_P;
- what code-word assign to each pixel p_C of the camera image, equivalently how to detect the code-word most similar to the local pattern distribution around p_C.

2.2.2.2.2 Coding Schemes

The key for triangulation-based 3D imaging is the technique used to differentiate a single projected light spot from the acquired image under a 2D projection pattern. Various schemes have been proposed for this purpose, and this section will provide an overview of various methods based on the structured-light illumination [45]. There are principally four possibilities, the three main ones illustrated in Figure 2.11:

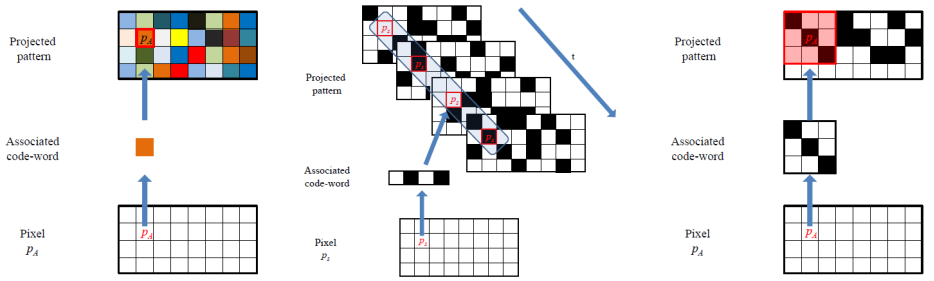


Fig. 2.11: The three main coding schemes: a) direct coding; b) time-multiplexing coding; c) spatial-multiplexing coding [34].

1. *Direct coding* represents the case in which the entire code-word is contained in a unique pixel ($n_W = 1$): the code-word is the

pattern value at the pixel itself (gray or color level). Only one image of the object under the pattern illumination is thus needed to compute the full frame of the depth image [97] and the maximum code-words cardinality is n_P;

2. *Time–multiplexing coding* is characterized by a sequence of T patterns projected at T subsequent times. The code-word for a given pattern pixel is formed by the sequence of illuminance values for that pixel across the sequence of the T projected patterns and thus there may be up to ${n_P}^T$ code-words [34];

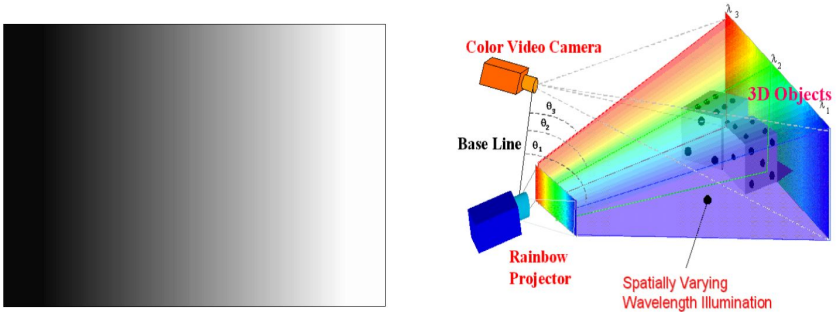
3. *Spatial–multiplexing coding* for which the code-word that identifies a certain point of the pattern is obtained from a spatial neighborhood of the points around it [97], precisely a window of n_W pixels centred around p_P. This option might have up to ${n_P}^{n_W}$ code-words. Note that neighbouring pixels share parts of their code-words, thus making their coding interdependent [34]. This technique concentrates all the coding scheme in a unique pattern (single shot pattern);

4. *Hybrid techniques* based on the combination of time–multiplexing with spatial–multiplexing methods: they project several patterns, but at the same time they also consider the information of spatial neighborhood in the decoding process.

In particular, two groups of methods belong to the category of direct coding strategies:

1. codification based on grey levels: a spectrum of grey levels is used to encode the pattern pixels [27];

2. color based codification: techniques that exploit a large spectrum of colors, such as the Rainbow 3D camera [44] which projects a spatially varying wavelength illumination onto the object surface.



(a) (b)

Fig. 2.12: Examples of direct codification patterns: (a) direct codification based on grey levels [27]; (b) Rainbow 3D camera [44].

As regards the time–multiplexing methods, they include:

1. techniques based on *binary codes*: black and white stripes are projected to form a sequence of projection patterns, such that each point on the object surface shows a unique binary code, different from the codes of the other points. In general, N patterns can code 2^N stripes [46];

2. techniques based on *Gray Code* encoding: this method works in a similar way as the binary encoding previously described, however it ensures that there is only one bit difference between consecutive patterns [53], [89];

3. techniques based on *n-ary codes*: in order to effectively reduce the number of the patterns needed to obtain a high resolution

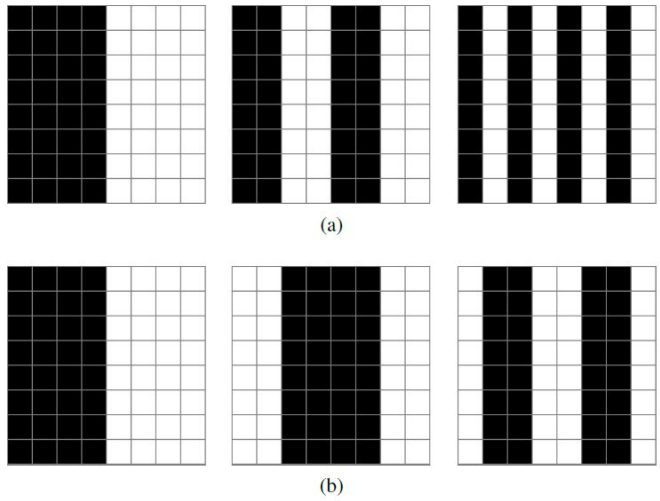


Fig. 2.13: (a) Binary encoding of the columns for an 8×8 pixels area. (b) Gray encoding corresponding to the same columns of the area in (a) [53].

depth map, a basis of n primitives is adopted to generate the code-words, *i.e.* n distinct levels of intensity (instead of only two in the binary code) are used to encode the pattern stripes [45], [97]. In this case, N patterns can code n^N stripes, more than the 2^N patterns needed by ordinary binary/Gray encoding techniques (Fig. 2.14) [97];

4. techniques based on *phase shift*: a well-known fringe projection method (see Fig. 2.15) that projects a set of phase shifted sinusoidal patterns onto the object surface [45];

5. techniques based on the *combination of Gray code encoding with the phase shift method*: the same Gray code pattern is projected several times, shifted in a certain direction in order to increase resolution [97].

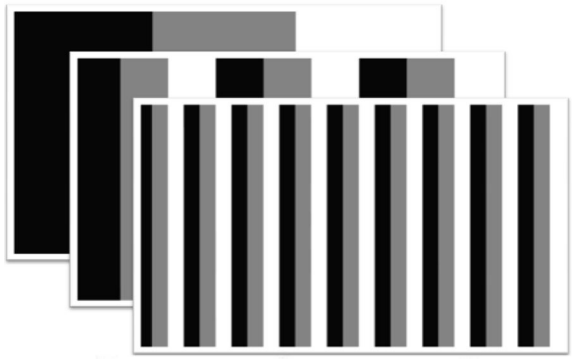


Fig. 2.14: Example of n-ary codes: three levels of intensity ($n = 3$) are used to encode 3^3 stripes with three patterns ($N = 3$) [45].

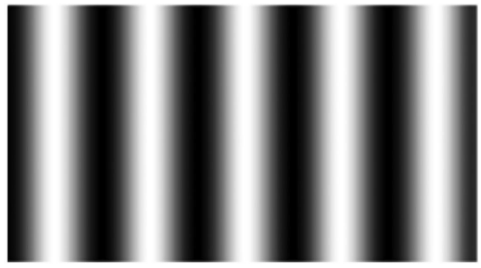
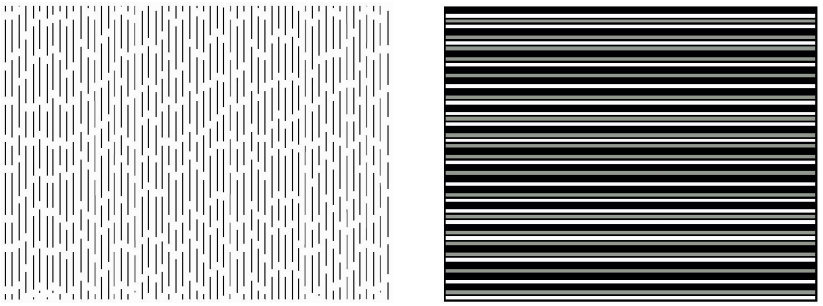


Fig. 2.15: Example of a fringe image [45].

Finally the spatial-multiplexing methods can be classified as follows:

1. strategies based on *non formal codification*: techniques (such as stripes indexing using colors, segment pattern or repeated gray-scale pattern) in which the pattern is divided into a certain number of regions in order to generate a different code-word intuitively, without using any mathematical coding theory [97];

2. strategies based on *De Bruijn sequences*: the spatial neighborhood window is defined using pseudorandom sequences. A De

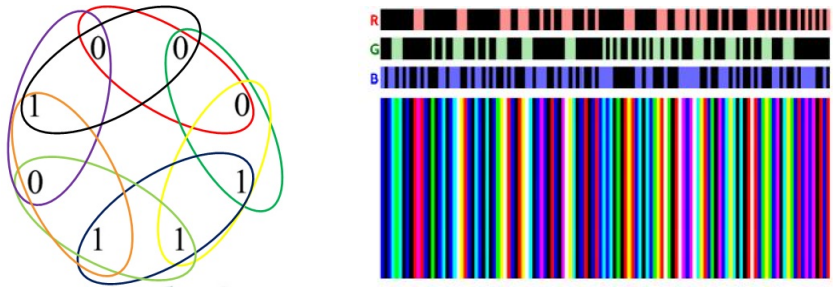


(a) (b)

Fig. 2.16: Example of patterns based on non-formal codification [97]: a) slits randomly cut [72]; b) periodic pattern [37].

Bruijn sequence of order m over an alphabet of n symbols is a circular string of length n^m that contains each substring of length m exactly once (Fig. 2.17a). Similarly, a *pseudorandom sequence* or a *m-sequence* has a length of $n^m - 1$ because it does not contain the substring formed by all zeros [38]. Pseudorandom sequences have been used to encode patterns based on column (Fig. 2.17b) or row lines and grid patterns (see next point) [97];

3. strategies based on *M-arrays* (also called *pseudorandom arrays*), the extension of the pseudorandom theory to the 2D case. A M-array is an $r \times c$ array, with $r \times c = 2^{nm} - 1$, where each nonzero $n \times m$ sub-matrix appears exactly once as a window in the array and each element is taken from an alphabet of k symbols [38]. M-arrays can be constructed through a pseudo-random binary sequence and are used to generate a 2D grid pattern that unambiguously labels every subwindow, such that any sub matrix is

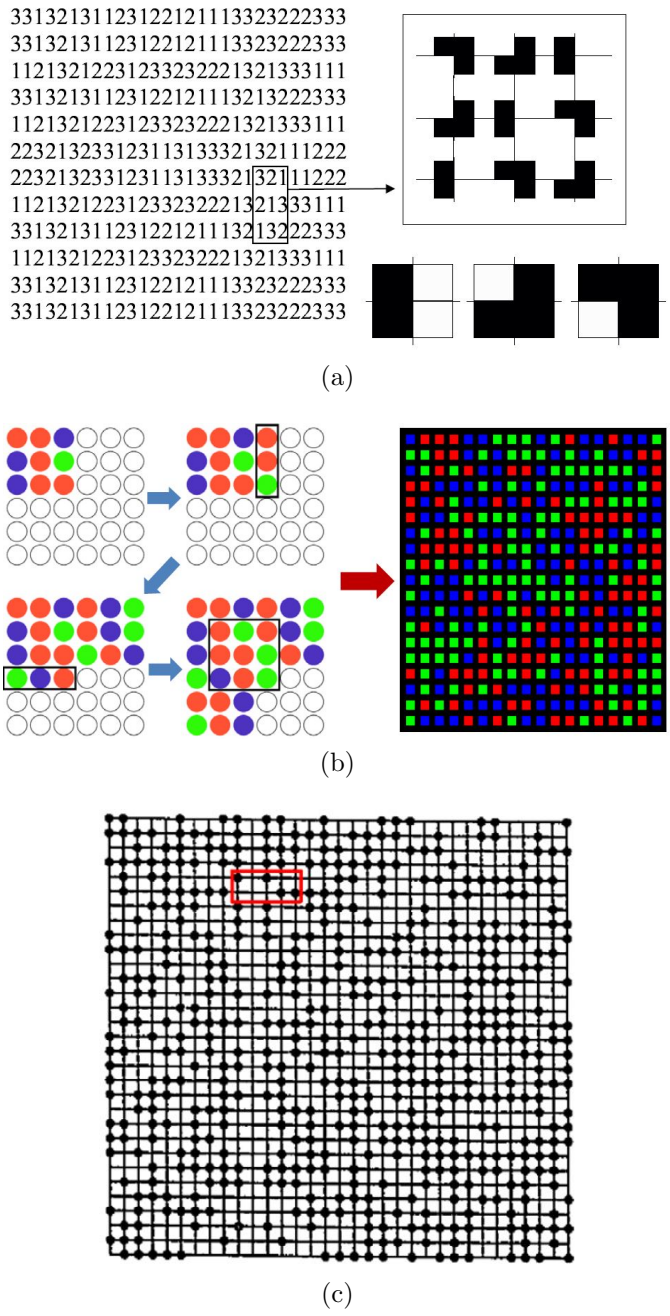


(a) (b)

Fig. 2.17: De Bruijn sequences: (a) sequence with $n = 2$ (the alphabet is $\{0, 1\}$) and $m = 3$ that generates 2^3 not repeating three-digit code-words (000, 001, 011, 111, 110, 101, 010, 100); (b) binary R, G, and B patterns generated using a De Bruijn sequence ($n = 5$ and $m = 3$): each three consecutive color transitions are unique [108].

unique and fully identifiable with respect to its 2D position in the pattern array [45]. The main differences between the techniques included in this group is the way in which the elements of the M-array, namely the k symbols, are represented in the pattern. Some authors prefer to define the pattern as an array of *colored spots* (Fig. 2.18a) or *colored dots*, other authors prefer to define different shapes for each symbol (*mini-patterns*, Fig. 2.18b) in order to obtain a multivalued pseudo-random array or to encode a *grid pattern* (Fig. 2.18c) where each cross-point represents an element of the M-array [97]

It is worth noting that each one of the four coding schemes described above has advantages and drawbacks. First of all, it is evident how direct coding methods are the easiest to implement. Furthermore, both direct coding and spatial–multiplexing coding strategies are suit-



(a)

(b)

(c)

Fig. 2.18: M-array based patterns examples: (a) M-array based on three shape primitives (mini-patterns) representing the symbols of the alphabet {1,2,3} [48]; (b) M-array with colored spots [73]; (c) A 31×33 M-array with a subwindow of 5×2 [66].

able to capture dynamic scenes, since they require the projection of a single pattern. Anyway, although time–multiplexing methods are not appropriate for dynamic situations, they can create arbitrarily different code-words for each pixel adopting a very small set of pattern primitives, like a binary set [34]. Finally, what makes the difference between direct coding schemes and spatial–multiplexing ones is that the latter are more robust with respect to projector and camera non-idealities and less sensitive to color or gray-level distortion due to scene color distribution, reflectivity properties and external illumination [34] as well. However, difficulties with occlusions and perspective distortion remain because of the finite size of the window. In fact, the choice of the window size (n_W) is crucial and should derive from a trade-off between robustness against perspective distortion (smaller windows) and robustness against non-idealities, noise and color distortions (greater windows). In conclusion, it is important to point out that all the methods just described work by using a visible light pattern/s, but nothing prevents to adopt the same methods with other wavelengths of the electromagnetic spectrum, as the next paragraph illustrates.

2.2.2.3 Infrared structured light: PrimeSense reference design

SL range cameras differ from traditional structured light systems since, instead of projecting a visible stream of changing shapes or bands of light, they illuminate the scene to be modeled with a NIR pattern.

The majority of SL cameras, for example the older ASUS devices (Xtion, Xtion PRO, Xtion PRO LIVE), the Microsoft Kinect v1 and

the newer Occipital Structure Sensor, Intel RealSense F200 and SR300[1], are based on the technology developed by PrimeSense Ltd., a former Israelian company acquired by Apple Inc. on November 2013, or a very similar system [36], [54], [59]. Their architecture is thus comparable to the PrimeSense reference design (see Fig. 2.19), with different choices for optional internal components and form factors, and thus it can be considered representative for describing their operation principle.

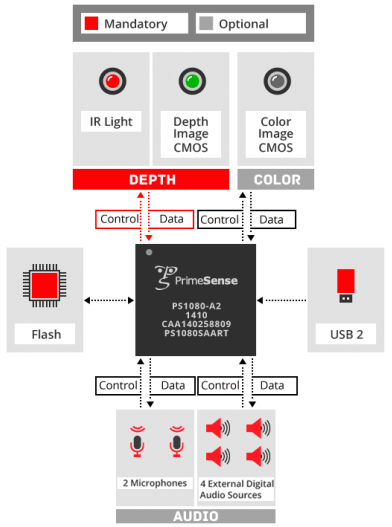
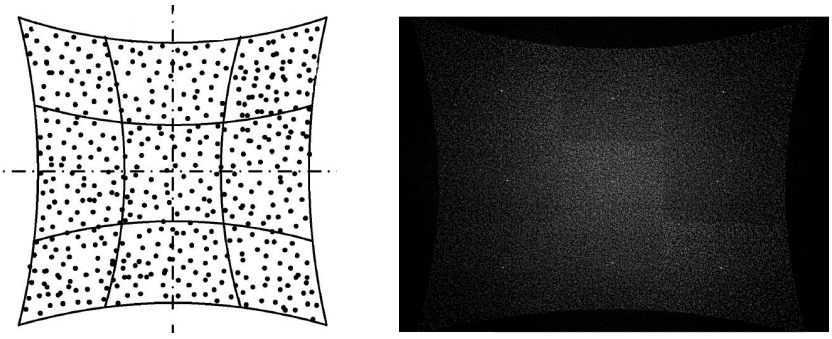


Fig. 2.19: PrimeSense depth sensor architecture [84].

In this design, the System on Chip (SoC) plays a fundamental role: it controls the IR light source that illuminates the scene with the IR light coded pattern and executes in parallel the algorithms that decode the image captured by the standard CMOS IR imaging sensor, computing a VGA depth map (640×480) of the scene at 30 fps. In

[1]The Intel RealSense R200, designed for a wider operative range, is an active stereo camera based on a different technology.

order to acquire the depth of dynamic scenes with such a high frame rate, the reference design exploits a PrimeSense proprietary technology called *Light Coding*™, which implements an active triangulation process based on the spatial-multiplexing approach. The implementation details are unique. The laser source emits a single beam which is split into multiple rays by a diffraction grating[1] to create a constant pattern of speckles [60] in the NIR light spectrum (830 nm) analogous to that shown in Fig. 2.20.



(a) (b)

Fig. 2.20: (a) PrimeSense speckle pattern as designed in the PrimeSense patent [101] and (b) the specific implementation of Kinect v1 [34].

The resulting pattern is characterized by an uncorrelated distribution across each row, feature used by several spatial–multiplexing methods to solve effectively the correspondence problem. Once the pattern is projected, for every frame the light response of the illuminated scene is captured by the infrared camera and it is correlated

[1]The Intel RealSense F200 projector consists of an IR laser diode, a line lens and a MEMS resonant micro mirror device. The last one moves thousands of times per second to scan the infrared light beam, painting an invisible grid on objects in front of it [36].

against the image of a reference pattern stored in the sensor memory. The reference image is obtained when the device is assembled through a *calibration procedure* in which the infrared camera acquires a planar surface oriented orthogonally to its optical axis, subject to the pattern illumination and most importantly placed at a known distance (depth) Z_{REF} from the sensor. Then this image, characterized by a constant and known disparity, is used in place of the emitted pattern for the tie points research that is carried out by means of correspondence algorithms. They consider a measure of the similarity (covariance) between a window centered around the specific pixel p_C in the acquired image of the scene and all the possible conjugates pixels in the same row of the pattern reference image. The algorithms simply select the pair $\{p_C(u_C, v), p_{REF}(u_{REF}, v)\}$ that maximizes the covariance [34].

It is important to point out that the efficiency of PrimeSense Light Coding technique is based on the reference pattern image: it allows to register significant covariance values, by overcoming the issues caused by the distortions of the projection process and avoiding the non-idealities of both projector and camera models. In this way the device chip can robustly compute the 3D coordinates of scene points starting from the horizontal shifts of the speckles in the image of the pattern projected onto the scene with respect to the corresponding ones in the reference image [43]. These shifts are measured for each speckle by the simple image correlation procedure described above that generates a 640×480 raw disparity image, using a 9×9 or 9×7 pixels window [62].

However, at least for the Kinect v1 sensor, this method creates a band of 8 pixels on the right of the image in which disparity cannot be computed, as shown in Fig. 2.21. The infrared camera has indeed

Fig. 2.21: Null band on the right area of the raw disparity image of the Kinect v1 sensor.

a wider vertical resolution (height) than the raw disparity image, but the same width. So there is no space on the imager to calculate more disparity information in the horizontal direction. On the contrary the vertical direction does not show the null band.

Anyway the computed disparities are essential since they carry out the depth information. Indeed the depth of a generic point K is inversely proportional to the observed disparity and it can be computed by exploiting the similarities of triangles (see Fig. 2.22) as follows [60]:

$$Z_K = \frac{Z_{REF}}{1 + \frac{Z_{REF}}{fb}\delta} \tag{2.19}$$

where Z_K denotes the depth of the point K from the sensor image plane in the object space, Z_{REF} is the known distance at which the reference pattern was captured, b is the length of the baseline between the infrared projector and the infrared camera, f is the focal length of the infrared camera and δ is precisely the observed disparity in the

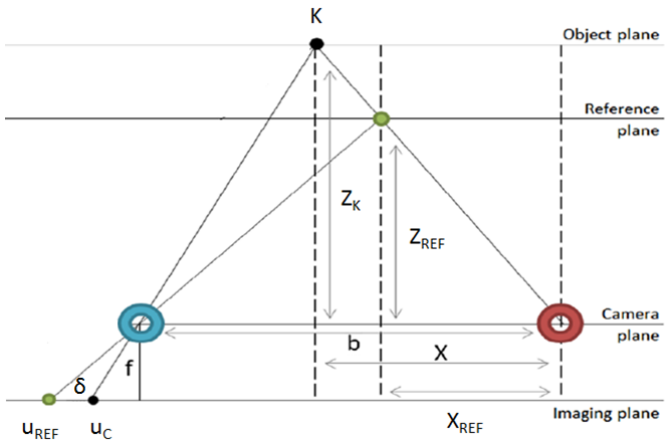


Fig. 2.22: Relation between relative depth and measured disparity [58]. The blue circle shows the location of the IR camera at a distance b from the IR projector, the red circle. The target point (black dot) is projected at depth Z on a plane farther from the reference pattern plane (green dot).

image plane, namely the displacement in pixels between the position of the point K in the scene image (u_C) and in the pattern reference image (u_{REF}):

$$\delta = u_C - u_{REF} \tag{2.20}$$

In conclusion it is worth noting that the adoption of the infrared spectrum has given a new impulse to the structured light technique. In fact, while this method is not conceptually new, the use of infrared patterns has made such sensors accessible to all, by allowing to build compact and low-cost devices, characterized by high frame rate and a relatively good accuracy. Furthermore, the projection of the IR light does not alter the texture of the captured object.

2.2.2.3.1 Error sources

As already described in the previous paragraph, there are several error sources that can potentially affect the depth estimation process of the SL range cameras. Some of these errors are due to the camera and/or projector, some to the adopted correspondence estimation algorithm and some to the geometry of the acquired scene [34].

For example, in the case of low reflectivity objects and/or excessive background illumination, the camera is unable to acquire any information about the reflected pattern and the correspondence algorithm cannot estimate the depth. The same effect occurs with very slanted surfaces in which the perspective distortion is too strong [34].

Furthermore, *occlusion* may happen at object boundaries where parts of the scene are not illuminated by the infrared pattern which results in a lack of depth information in those regions (invalid pixels) [98].

Finally, also the SL range cameras are active sensors working with the infrared spectrum and thus they show the same interference issues of ToF range cameras (see Section 2.2.1.2.1) in presence of other infrared light sources. Thus they cannot be used outdoor in a sunny day and the simultaneous use of several active IR devices may lead to *multi–device interference*. For the same reason these sensors can show difficulties to model black (they reduce the power of the pattern signal, absorbing the infrared light), shiny (they show a too high amplification of the reflected pattern signal) and transparent surfaces.

Moreover also SL sensors are affected by an instability of depth measurements during the warm-up time [32], [98] even if, generally, they do not produce as much heat as ToF cameras. Indeed they usually

require less illumination power to cover the scene with a relatively sparse point-based pattern than the ToF cameras to get a sufficient SNR for the emitted IR signal [98].

2.2.3 Software libraries for 3D modeling with range cameras

Specific drivers are needed to concretely turn the raw signals coming out of range cameras into usable 3D data, such as depth maps and/or point clouds.

Furthermore, in order to perform 3D modeling applications, it is essential to have available efficient *Simultaneous Localization And Mapping* (SLAM) algorithms that, by continuously tracking the position of range cameras, allow to use them as veritable 3D scanners. In fact, multiple scans from different points of view are usually required to collect complete information about the whole surface of the target object [20]. By exploiting the depth data and the high frame rate that range cameras offer, algorithms like KinectFusion [55, 78] are indeed able to continually reconstruct the six *Degrees of Freedom* (DoF) pose of the moving sensor and fuse the object depth maps captured from new view points as soon as they are acquired, merging them into an overall 3D model in real time. Such tracking process[1] is the fundamental stage to the 3D model generation and it can be carried out in different ways using the various implementations available in several software libraries. It is worth noting that the availability of this kind of algo-

[1] In this context the term tracking identifies the process by which a 3D scanner is able to lock onto and reliably track its own motion in relation to the object being scanned.

rithms has been precisely the key for the success of range cameras in the 3D modeling field.

The interested reader can find more details about the drivers, the algorithms and the software actually available for 3D modelling with range cameras in appendix A.

Chapter 3

Analysis of the geometric potential of low-cost range cameras

In this chapter the 3D modeling capabilities of the three most popular and/or promising low-cost range cameras, namely the Kinect v1, the Kinect v2 and the Occipital Structure Sensor, are investigated and compared. The specific aim is to evaluate their potentialities for geomatic applications and to provide useful indications for their practical use. Thus it is necessary to assess the metric quality (precision and accuracy) of the depth data retrieved by these sensors.

The chapter is organized as follows. In Sec. 3.1 the main features of the investigated sensors are described. In Sec. 3.2 the used reference systems are presented. In Sec. 3.3 the geomatic characterization of the three considered devices is illustrated and the models to represent their random and systematic errors are introduced and proven adequate for

calibrating the sensors. Finally, in Sec. 3.4 some investigations about the registration process of depth and color images are described.

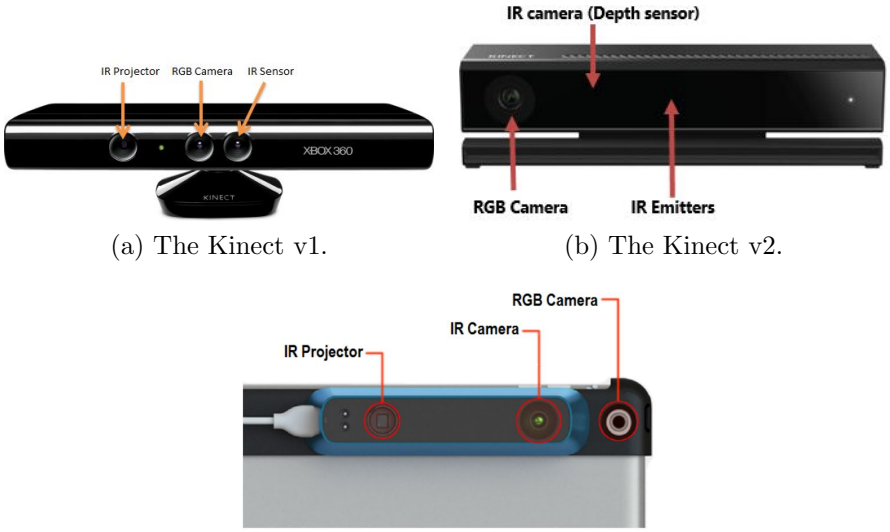
3.1 Sensor features

As is well known, the first Microsoft Kinect has opened the way to a new generation of range cameras, although it was primarily designed as motion sensing input peripheral for Xbox 360 game console. In fact, both the Kinect v1 and the Kinect v2 are webcamstyle, add-on devices which enable users to interact with the game console without using a traditional hand-held joystick. Players can indeed control play, action and movement of their on-screen characters only using body gestures, through a NUI. Gesture recognition is made possible by the devices depth sensor which provides a real time depth map. Therefore the Kinect depth sensor is, to all intents and purposes, a range camera.

On the other side, the Occipital Structure Sensor has been conceived from the beginning to be the first range camera for mobile devices: characterized by compact dimensions and an internal battery, it can be quickly and securely connected to an iOS device through a specific bracket accessory. It was launched on Kickstarter in September 2013, raising almost 1.3 millions of dollars in 45 days of campaign.

Therefore, the technical specifications of the three investigated sensors are very different, although their principal hardware components (see Fig. 3.1), namely the IR camera, the IR projector (for SL sensors: the Kinect v1 and the Structure Sensor) or the IR emitters (for the ToF sensor, that is the Kinect v2) and the color camera, if present, fulfill the same function: the first two make up the depth sensor, that

acquires the depth data, whereas the latter captures the texture information. Thus also the characteristics of the data captured by them are diverse.



(a) The Kinect v1. (b) The Kinect v2.

(c) The Structure Sensor.

Fig. 3.1: The internal components of the three investigated sensors.

As regards the depth stream, the Kinect v1 supports three different resolutions: 640×480 pixels, 320×240 pixels, and 80×60 pixels, all at 30 frames per second (fps). The Kinect v2 sensor provides instead a 512×424 16-bit depth stream, again at the same 30 fps acquisition rate. The bit-depth layout is exactly the same as Kinect v1 sensor, with 13 bits representing the pixel depth, and the remaining 3 bits used as segmentation mask [102].

Furthermore, the Kinect v2 sensor uses an active infrared stream, lighting independent, characterized by a 512×424 resolution, at 30

frames per second. Thanks to USB3 connection, the infrared stream can be used simultaneously with the colour stream, whereas for the Kinect v1 this is not possible because of USB2 low bandwidth and infrared data (640×480 resolution - 30 fps) can be retrieved only disabling the colour stream. Moreover, the Kinect v2 sensor presents a wider field of view (70°×60°), without needing a tilt motor, as it happens for Kinect v1 (57°×43°), where the motorized base can move the sensor bar 27° up and down.

In addition to the depth sensor, the bar of both the two Kinect sensors houses also the colour camera, an accelerometer and an array of four microphones (see Fig. 3.1a and Fig. 3.1b). The colour camera does not participate in the depth sensing process, but it has the purpose to collect the texture of the scene for applications like face tracking. The Kinect v1 colour camera is a 8-bit resolution VGA camera which supports different resolutions at different frame rates [28]: 640×480 pixels at 30 fps using RGB[1] format; 1280×960 pixels at 12 fps using RGB format; 640×480 pixels at 15 fps using YUV[2] (or raw YUV) format. Instead, the Kinect v2 colour camera is a full HD camera, with a resolution of 1920×1080 pixels returned at 30 fps in the YUY2[3] raw image format (the acquisition rate drops to 15 fps in case of low light). The microphones are needed for speech recognition: they can record audio as well as find the location of the sound source and the

[1]The RGB format is a 32-bit format that uses a linear X8R8G8B8 formatted colour in a standard Red Green Blue colour space: each component can vary from 0 to 255, inclusively.

[2]The YUV format is a 16-bit, gamma-corrected linear UYUY-formatted colour bitmap.

[3]YUY2 format packs two pixels as four 8-bit components: Y1, U, Y2, V where Y1 and Y2 are individual pixel luminance values, and U and V are shared chrominance values for the two pixels.

direction of the audio wave. Finally the tri-axis accelerometer (a Kionix KXSD9 for v1, a Kionix KXUD9 [31] for v2) is used to determine the orientation of the sensor (for Kinect v2 accelerometer APIs are no longer exposed in the official Microsoft SDK).

Concerning the Structure Sensor, it can stream depth data at two different depth resolutions: 320×240 pixels at both 30 fps and 60 fps and 640×480 pixels at 30 fps. As regards the infrared stream, the device provides two streams: one at 320×248 at 30 fps and one at 640×488 at 30 fps. However Structure Sensor does not have an own color camera, but it can retrieve the RGB data from the iOS device at which is connected. The depth streams at 30 fps can be aligned to the iOS device colour camera through the Structure SDK.

The technical specifications of the investigated sensors are summarized in Table 3.1.

	KINECT v1	**KINECT v2**	**STRUCTURE SENSOR**
Technology	Structured Light	Time of Flight	Structured Light
Official Depth Range	0.4−4 m	0.5−4.5 m	0.4−3.5 m
Depth Field of View (H×V)	57°×43°	70°×60°	58°×45°
Colour Stream	640×480 @ 30 fps 1280×960 @ 12 fps	1920×1080 @ 30 fps	n.a.
Depth Stream	80×60 @ 30 fps 320×240 @ 30 fps 640×480 @ 30 fps	512×424 @ 30 fps	320×240 @ 30 fps 320×240 @ 60 fps 640×480 @ 30 fps
Infrared Stream	640×480 @ 30 fps	512×424 @ 30 fps	320×248 @ 30 fps 640×488 @ 30 fps
Audio stream	4-mic array 16kHz	4-mic array 48kHz	n.a.
SDK Officially Supported OSs	Windows 7 Windows 8 Windows 8.1	Windows 8 Windows 8.1	iOs
Cost	150 $	150 $	379 $

Tab. 3.1: Technical specifications of the investigated sensors.

3.2 Reference System and coordinate calculation

With respect to reference systems, a main distinction has to be done between the *image reference systems* and *object reference system*.

The former systems are associated to the images captured respectively by the IR camera, the depth sensor and the color camera of the considered device (for the Sructure Sensor, the color camera of the iOS device at which it is connected).

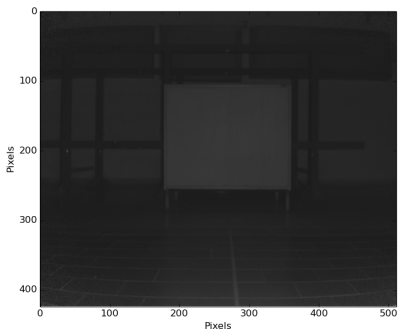
The latter system is the one related to the sensor itself and in which the depths are measured.

3.2.1 Image reference systems

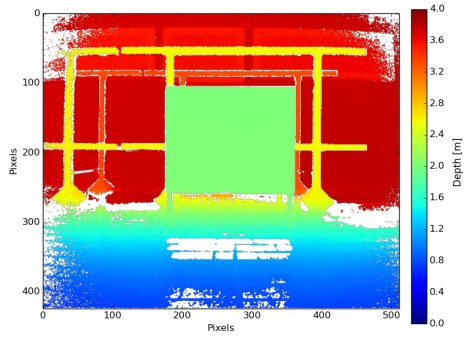
The image reference systems are three as the number of the streams retrieved by the range cameras: the infrared image system I_{IR}, the depth image system I_{DEPTH} and the color image system I_{RGB} can be distinguished. Anyway, since the depth image derives from calculations executed over the IR image, I_{IR} and I_{DEPTH} coincide. Instead, considering that the object geometry and the texture are captured by the depth sensor and the color camera from two different points of view, the depth image and the color image are placed in two diverse image reference systems. Therefore, in order to overlap them, it is theoretically necessary to estimate the roto-translation transform (and the intrinsic parameters) that brings one system to the other (see Sec. 2.1.1). Luckily the SDKs of the three used sensors provide specific mapping methods that allow to easily map a color pixel into the cor-

responding position of the depth map or viceversa (for the Structure Sensor the calibration must be previously performed).

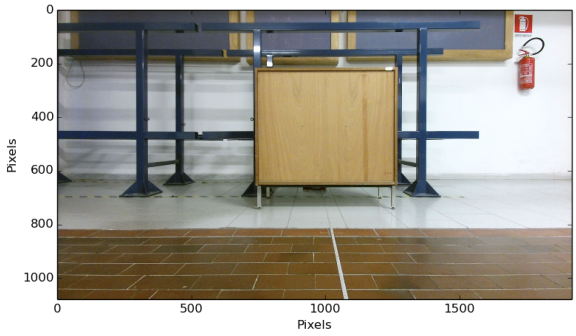
In particular, the image reference systems are planar and their coordinates are always positive. The origin is located in the top-left corner of the respective image, the x axis is directed rightwards and the y axis is downward.



(a) IR image.



(b) Depth image.



(c) Color image.

Fig. 3.2: The same scene in the three images captured by the Kinect v2.

For example, in the case of the Kinect v2 (see Fig. 3.2), the coordinates of the pixel on the top-left corner shows a value of zero in all

of the three image reference systems:

$$x_{IR} = x_{DEPTH} = x_{RGB} = 0 \tag{3.1a}$$

$$y_{IR} = y_{DEPTH} = y_{RGB} = 0 \tag{3.1b}$$

while for the pixel on the bottom-right corner, the x and y coordinates in the IR and depth image are respectively equal to the values of the IR width and IR height both decreased by one, and to the values of the RGB width and RGB height again decreased by one in the color image:

$$x_{IR} = x_{DEPTH} = 511 \quad x_{RGB} = 1919 \tag{3.2a}$$

$$y_{IR} = y_{DEPTH} = 423 \quad y_{RGB} = 1079 \tag{3.2b}$$

3.2.2 Object reference system

To express the 3D coordinates of points in the object space, a local reference system, sensor–centred, is used for all of the three devices. Indeed, although the value of each depth image pixel corresponds immediately to a physical distance, the x and y positions of the pixels in the depth image do not map directly to a physical location in the object space [34]. In the following paragraphs, the methods adopted to compute the point cloud starting from the depth image are described for each sensor. It is important to notice that in this way the obtained point cloud counts a number of points equal to the resolution of the depth map.

3.2.2.1 Kinect v1

For the Kinect v1, a 3D reference system with the origin in the central pixel of the depth/infrared image ($x_{IR} = 320$ and $y_{IR} = 240$) is considered. The Z axis is orthogonal to the image plane and oriented towards the object, while the X axis is oriented leftwards and the Y axis is oriented upwards (see Figure 3.3).

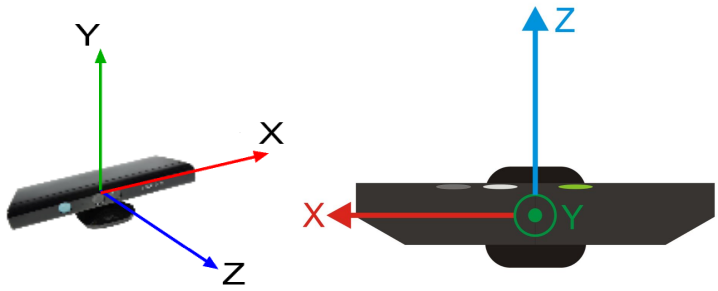


Fig. 3.3: The local Kinect reference system.

Obviously the Z coordinate for an acquired point is already known since it corresponds to the depth value, expressed in millimeters, stored in the depth map. To obtain the desired point cloud, it is necessary to compute the dimension of the pixels in the object space, respectively along the X axis and Y axis, through the following equations:

$$dp_X = \frac{2\,Z \tan\left(\frac{\alpha}{2}\right)}{HR} \tag{3.3a}$$

$$dp_Y = \frac{2\,Z \tan\left(\frac{\beta}{2}\right)}{VR} \tag{3.3b}$$

where dp_X and dp_Y are the metric pixel dimensions respectively along the X axis and Y axis, expressed in mm; Z is the Z coordinate of the pixel, expressed in mm and corresponding to its depth value; HR and

VR are the horizontal and vertical resolution of the depth image and finally α and β denote the Kinect v1 horizontal and vertical field of view. In this way, multiplying the x and y coordinates of the pixels by the metric pixel dimensions, the X and Y coordinates can be easily calculated:

$$X = \left(x - \frac{HR}{2}\right) dp_X \tag{3.4a}$$

$$Y = \left(-y + \frac{VR}{2}\right) dp_Y \tag{3.4b}$$

where X is the X coordinate, expressed in mm; Y is the Y coordinate, expressed in mm; x and y are the horizontal and vertical positions of the pixel in the depth image.

3.2.2.2 Kinect v2

The Kinect for Windows SDK 2.0 provides a method (MapDepthFrameToCameraSpace) that directly maps the depth image reference system to the object space.

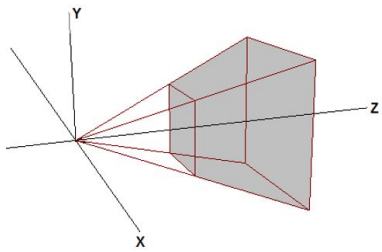


Fig. 3.4: The Kinect v2 reference system.

In particular, the 3D coordinate system used by the Kinect v2 (see Fig. 3.4) has its origin in the center of the IR/depth sensor, the X axis

oriented leftwards, the Y axis oriented upwards, whereas this direction is based on the sensor tilt, and the Z axis increases along the direction in which the sensor is pointed. In this case the 3D coordinates are retrieved in meters.

3.2.2.3 Structure Sensor

For the Structure Sensor, a 3D reference system with its origin in the principal point (c_x, c_y) of the IR/depth image is taken into account. The Z axis is orthogonal to the image plane and oriented towards the object, whereas the X axis is rightward and the Y axis is upward. In this case the X and Y coordinates are computed through the perspective projection relationship [34]:

$$X = Z\left(\frac{x - c_x}{f_x}\right) \tag{3.5a}$$

$$Y = Z\left(\frac{c_y - y}{f_y}\right) \tag{3.5b}$$

where X and Y are the coordinates of the pixel in the object space, expressed in mm; Z denotes the coordinate of the pixel, expressed in mm and corresponding to its depth value; c_x and c_y are the coordinate of the principal point in the depth image; f_x and f_y are the focal lengths along the x and y axes.

3.3 Models for depth precision and depth accuracy

In this work one Kinect v1, one Kinect v2 and one Structure Sensor were considered and characterized from a geomatic point of view. A specific test field was thus implemented in order to analyze the behaviors of the random and systematic errors of the depth measurements for the three investigated sensors.

In particular, the planar opaque surface of a cabinet (see Fig. 3.5b) was acquired by each sensor at several known distances, taking care to place each sensor as parallel as possible to the target plane. For each distance, the Kinect v1, the Kinect v2 and the Structure Sensor were set together on top of the same tripod (see Fig. 3.5a).

(a) (b)

Fig. 3.5: The experimental setup. (a) From top to bottom: the Structure Sensor, the Kinect v1, a spirit level and the Kinect v2 mounted on the tripod. (b) The tripod with the three sensors in front of the reference planar surface.

Moreover, in order to carefully measure the reference geometry with a total station, three targets were fixed onto the back side of the sensors. The parallelism between each sensor and the planar surface was achieved assuring that the horizontal distances of the three targets measured with the total station were equal (with an accuracy better than 1 mm) at each distance. In this way the tripod was moved for a total of nine steps from 90 to 400 cm from the planar surface. Lower distances were not investigated since in that case the Kinect v2 depth maps were affected by a too strong backscattering signal that compromised the measures.

The procedure was identically repeated twice for each device, always considering the corresponding warm-up time [32], [98], [63]. For each position/step, 150 consecutive depth maps were acquired by each sensor, but, in order to exclusively investigate the area well within the reference planar surface, only the central 50×50 pixel window was considered.

3.3.1 Depth resolution analysis

To evaluate the depth resolution of the three sensors, the depth histograms were computed considering a sample of 375000 observations (the depth of 50×50 pixels in the window for the 150 frames) for each distance from the reference surface (in our test case equivalent to the depth). Fig. 3.9 and Fig. 3.10 show the results obtained and clearly highlight two different behaviors, depending on the operational principle (SL or ToF) of the single sensor.

In particular, the resolution of the SL devices (Fig. 3.6 and Fig. 3.8) worsens following a parabolic trend (R^2 of 1.00 for both the Kinect

v1 and the Structure Sensor) when the distance/depth from the target increases (d_R); on the other side, the Kinect v2 resolution shows a constant value of 1 mm for every distance (Fig. 3.7).

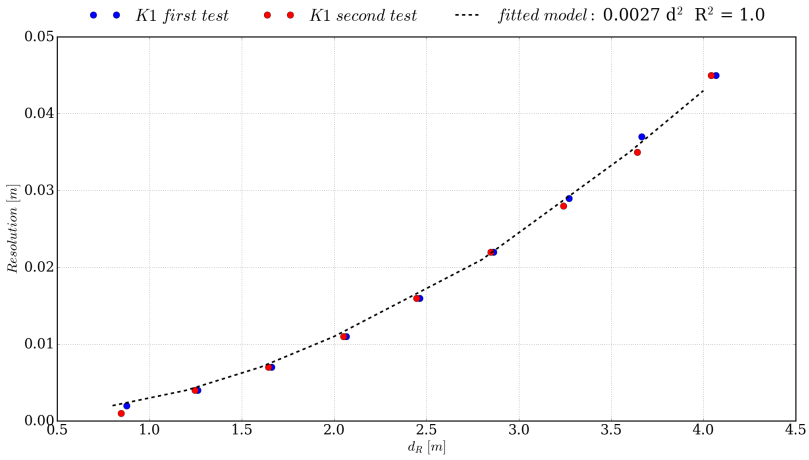


Fig. 3.6: Kinect v1 resolution.

Thus, the depth data acquired by the Kinect v2 are characterized by a substantially continuous distribution with a resolution of a millimiter level, independently from the distance from the reference plane; on the contrary, for Kinect v1 and Structure Sensor the depth data present a discrete distribution, with a resolution dependent from the distance itself. This behaviour can be explained considering that, for the SL range cameras, the resolution depends on the minimum measurable disparity $\Delta\delta$, so that, starting from Eq. 2.19, by finite differentiation, the resolution ΔZ_k at distance (or depth) Z_k results:

$$\Delta Z_k = \frac{{Z_k}^2}{fb}\,\Delta\delta \tag{3.6}$$

displaying a parabolic dependence over distance, as it is expected by a triangulation system.

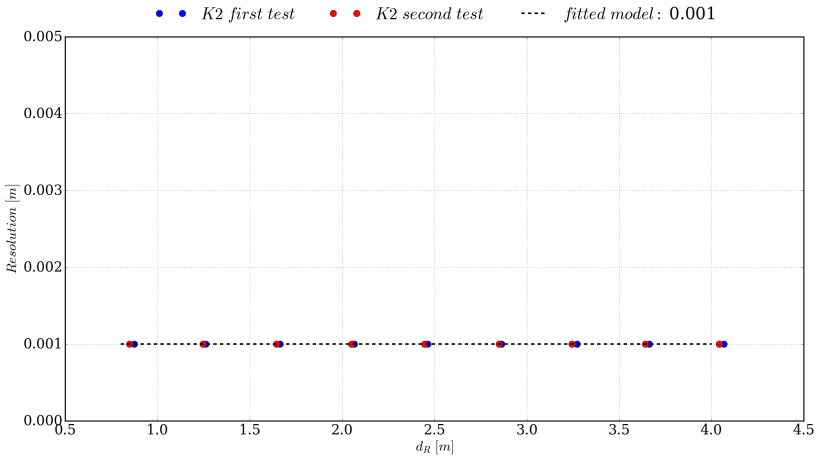


Fig. 3.7: Kinect v2 resolution.

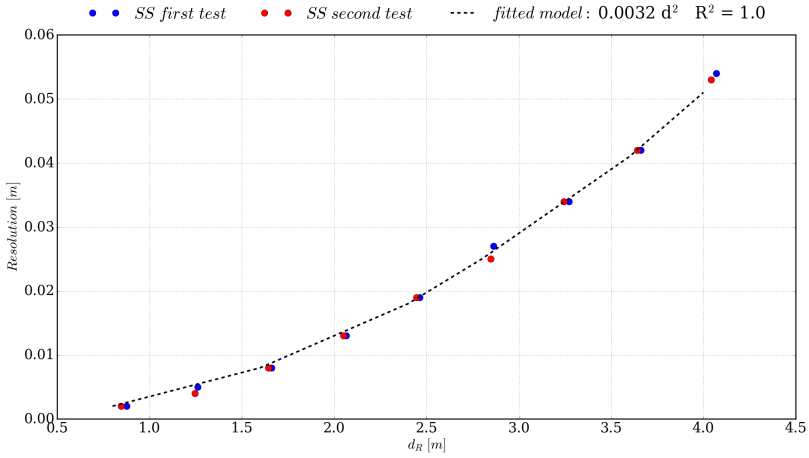


Fig. 3.8: Structure Sensor resolution.

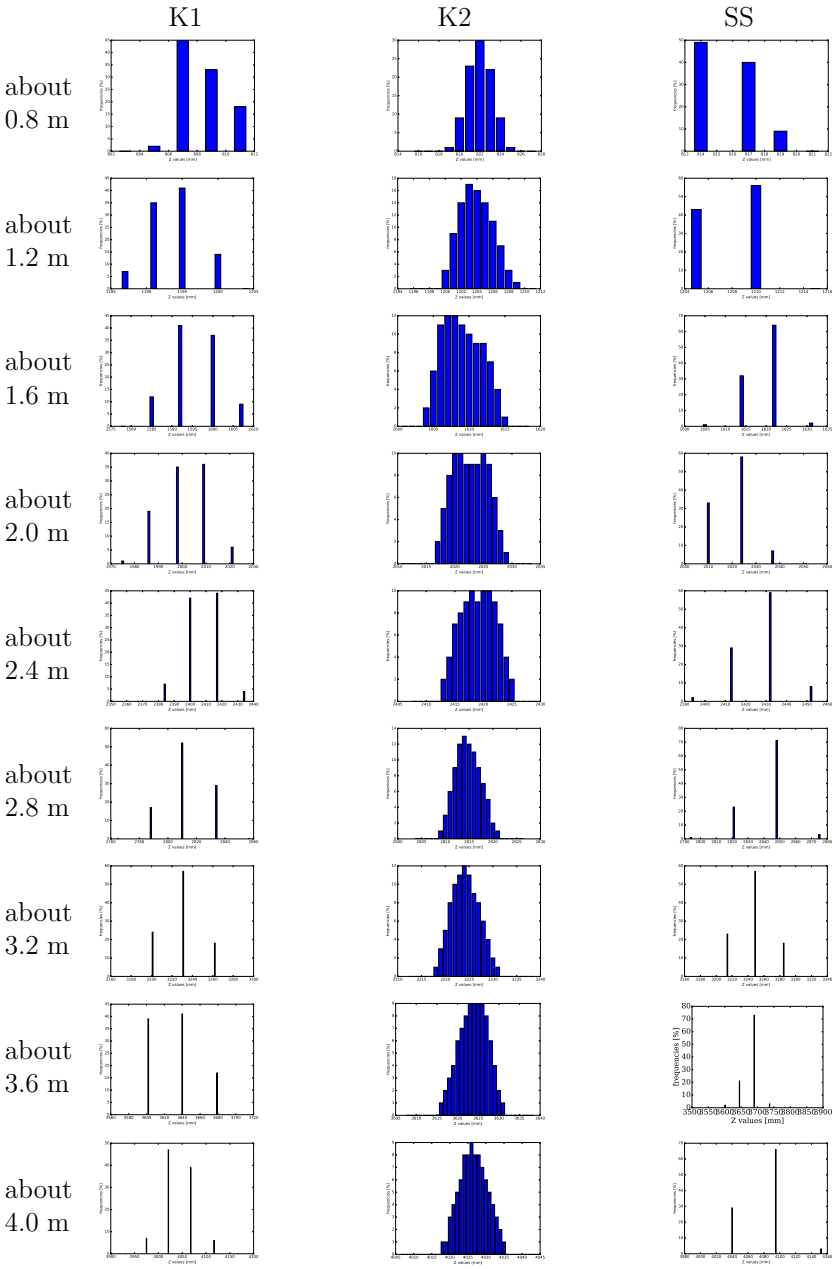


Fig. 3.9: Distributions at different depths/distances of the 50×50 depth values over the 150 frames for the first test.

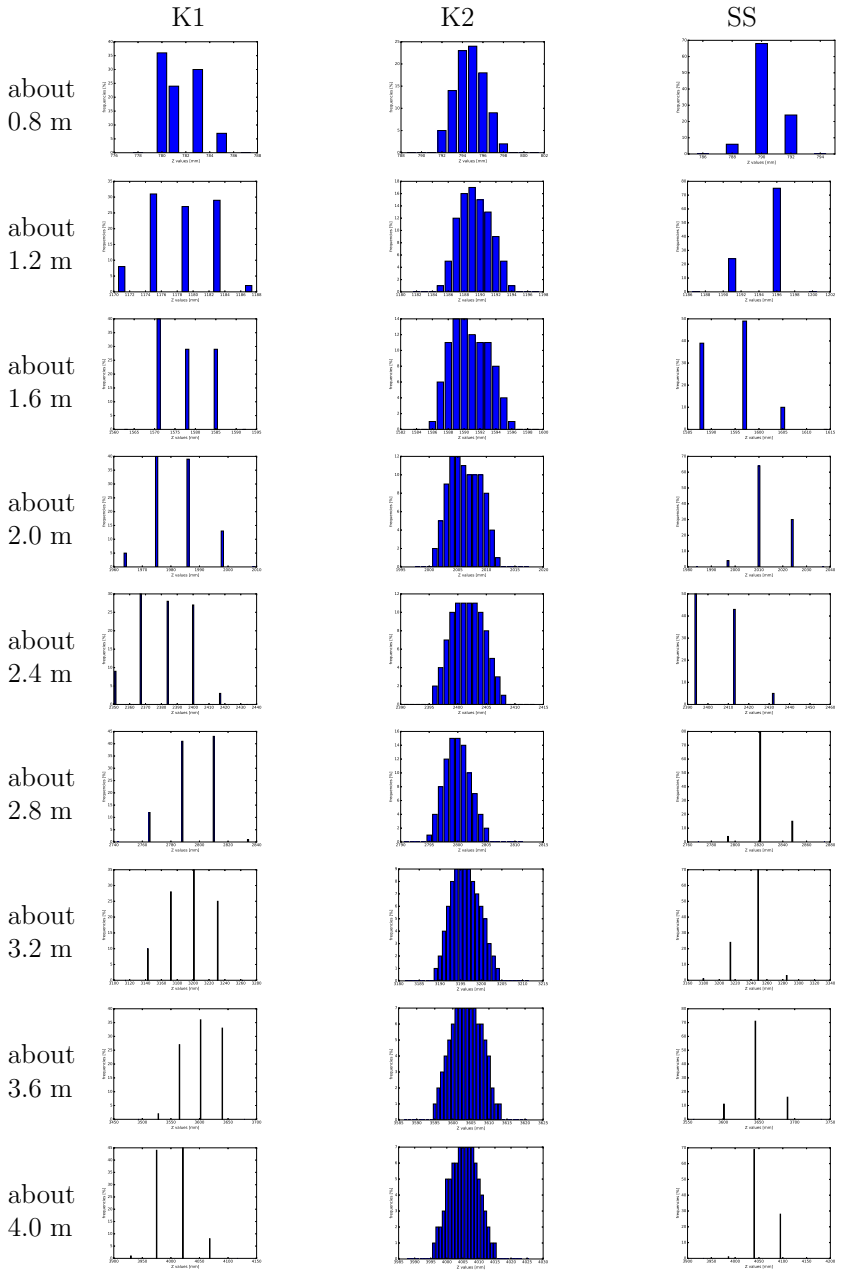


Fig. 3.10: Distributions at different depths/distances of the 50×50 depth values over the 150 frames for the second test.

3.3.2 Depth precision analysis

Considering the previous resolution analysis, the precision was evaluated through two different methods.

For the Kinect v1 and the Structure Sensor, the half of the resolution was considered representative of their precision (Fig. 3.11 and Fig. 3.12).

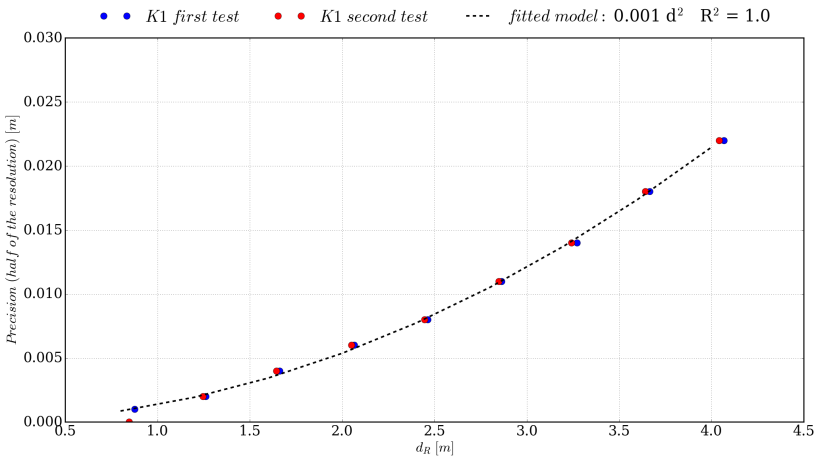


Fig. 3.11: Kinect v1 precision vs. depth.

Differently for the Kinect v2, whose resolution is independent from the distance and equal to 1 mm, the global standard deviation was calculated over the sample of 375000 depth observations for each distance from the reference surface. In details, the equation of the cabinet planar surface was least squares estimated for each distance and the dispersion around this plane in terms of standard deviation was considered as the precision (Fig. 3.13); the obtained results clearly show that also for the Kinect v2 the depth precision is getting coarser as the

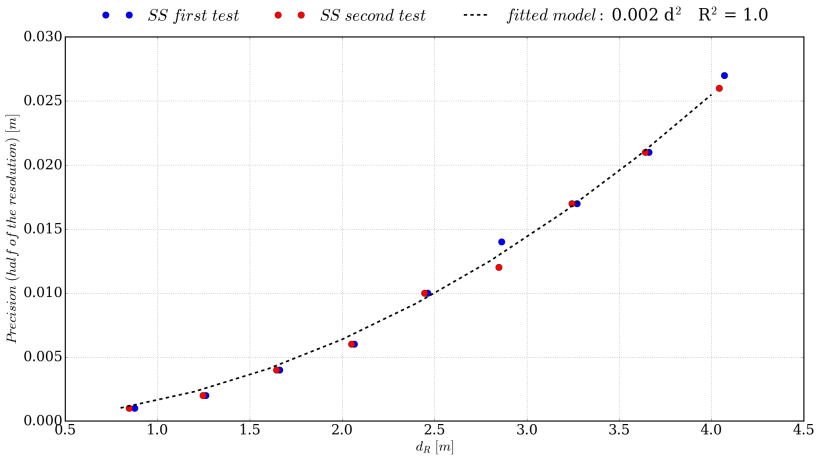


Fig. 3.12: Structure Sensor precision vs. depth.

distance from the target (d_R) increases. In particular, the precision varies linearly with the measured distance (R^2 of 0.96), as already reported by [82]. It is also evident (note that the scale along the vertical axis in Fig. 3.13 is magnified 6 times with respect to the scales of the corresponding axes in Fig. 3.11 and Fig. 3.12) that Kinect v2 is the most precise sensor: the dispersion reaches the value of only 5 mm at 4 meters, clearly better than the 22 mm of Kinect v1 and 27 mm of Structure Sensor.

In addition, to globally visualize the depth measurement noise, the depth standard deviation over the 150 frames of each pixel inside the analysed 50×50 window is shown in Fig. 3.14 and Fig. 3.15 for the two tests respectively.

In general the results show that SL depths are noisier than ToF ones; however the latter are much more sensitive to the scanned ob-

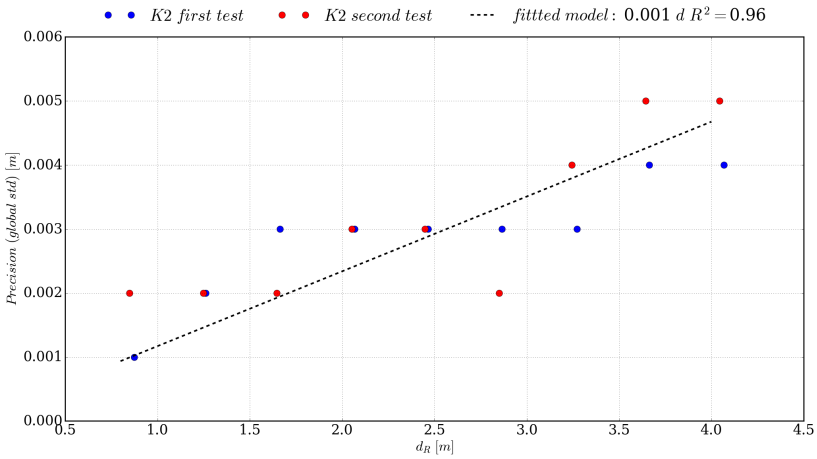


Fig. 3.13: Kinect v2 precision vs. depth.

ject material: highly glossy surfaces and color differences may produce different depth estimates, as reported in [51]. For each sensor, the statistical features (standard deviation for rows and columns and global standard deviation) are stable in the two tests, and also the cross correlation between the two tests at the same distance are very low (0.21 at the most), indicating that there are not stable patterns highlighting biases in the sensor behaviours, as also the figures show. Only in the case of the Kinect v1 [81] and the Structure Sensor, some evident permanent features (vertical bands) remain.

In the end, concerning the dependence of the depth random errors, that is precision, on the depth itself, the simple parabolic (for the Kinect v1 and the Structure Sensor) and linear (for the Kinect v2) models (Fig. 3.11, Fig. 3.13 and Fig. 3.12; Tab. 3.2) appear effective, at least under good reflective surface conditions.

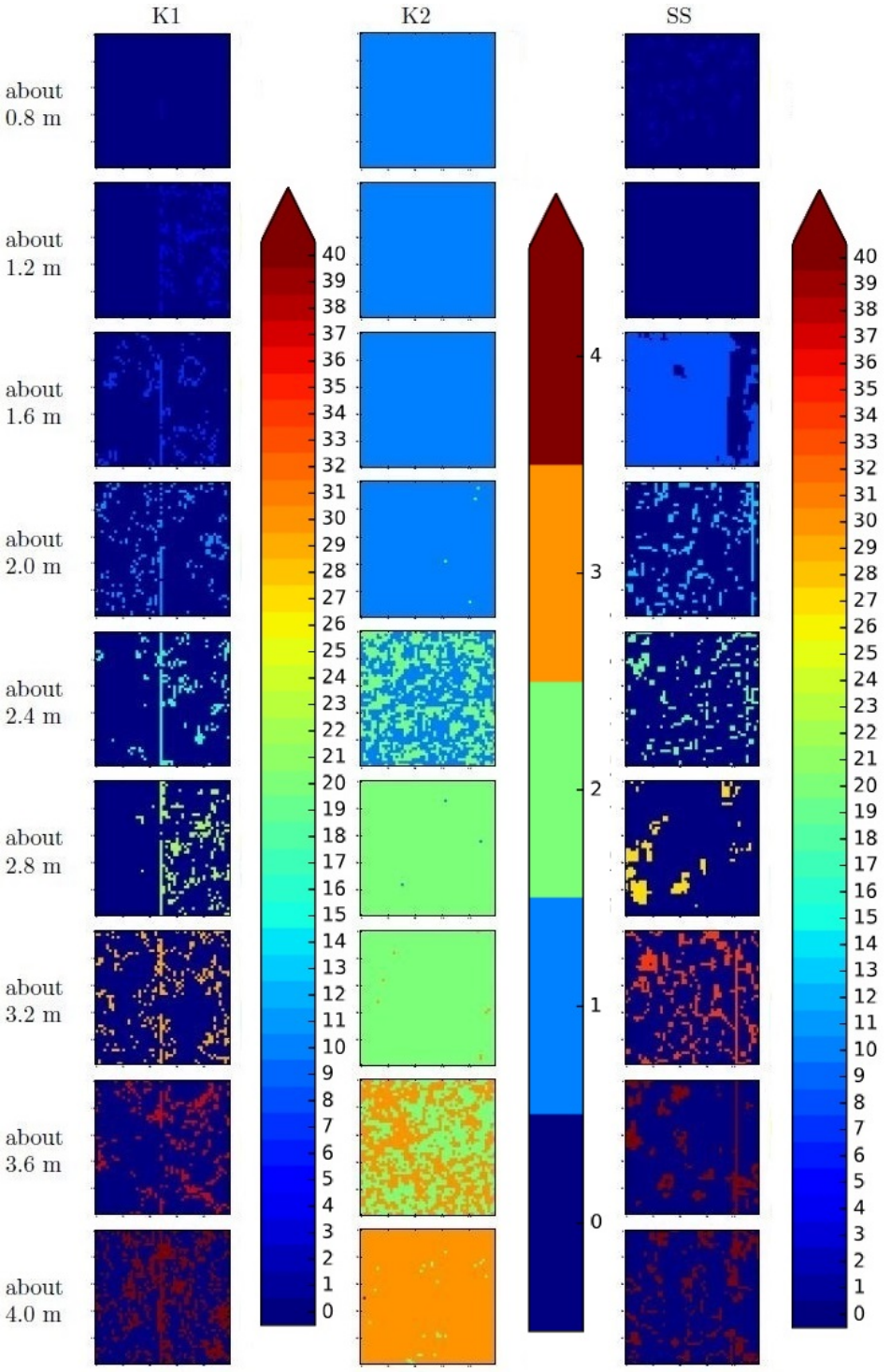


Fig. 3.14: Depth standard deviation (expressed in mm) for each pixel of the 50×50 window over the 150 frames for the first test.

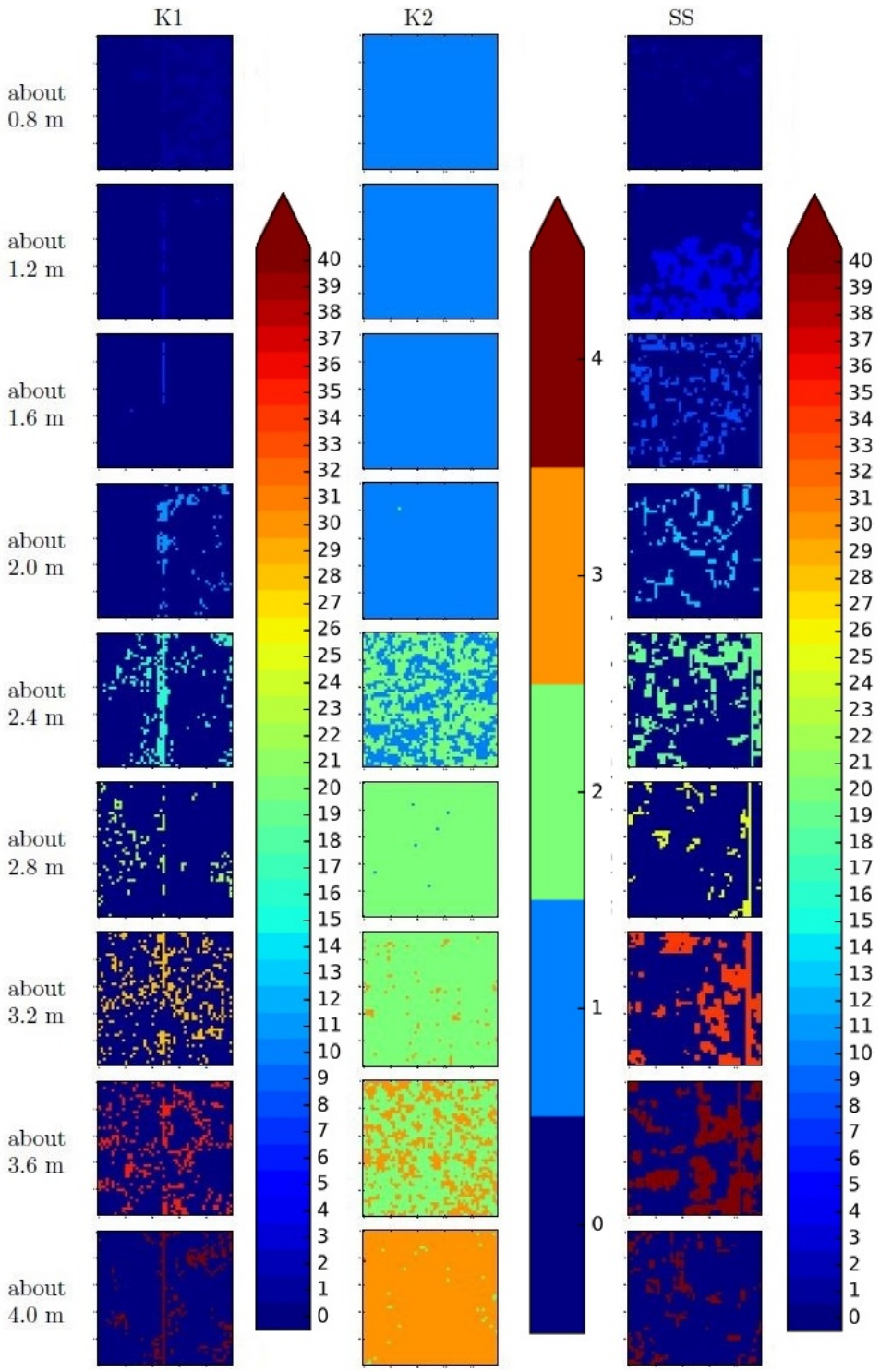


Fig. 3.15: Depth standard deviation (expressed in mm) for each pixel of the 50×50 window over the 150 frames for the second test.

3.3.3 Depth accuracy assessment

The aim of the accuracy assessment is to compare the reference depth values with the same depth measured with the three sensors. This test is meant to show the geomatic potential of the investigated sensors in modeling a known surface (here represented by the cabinet planar surface) through a close-range survey.

So, for each depth/distance from the reference surface, the global mean was computed over the sample of 375000 depth observations. In this way it was possible to compute the differences between the observed values (d_O) and the reference depths (d_R) measured with the total station.

Due to its high precision of few millimeters, for the Kinect v2 the reference depths d_R were corrected in order to consider the possible residual inclination between the sensor and the reference plane surface. Only the inclination angle along the vertical direction was considered, since for the installation features it was the most critical to control, and their parameters ($Y = aZ + c$) were least squares estimated; then the effect of this vertical inclination was removed in the following comparison between the observed and reference depths. Moreover, the depth mean over the 150 frames of the depth map central pixel was used as observation for each distance from the reference plane surface.

The results are reported in Fig. 3.16, Fig. 3.17 and Fig. 3.18. A visible trend of the accuracy vs. the depth/distance is evident for all the sensors, pointing out the presence of systematic errors, which increase with the distance from the reference surface. In particular the Kinect v2 is once again the best sensor, showing the lowest systematic error range: 0.019 m between the shortest and longest distance

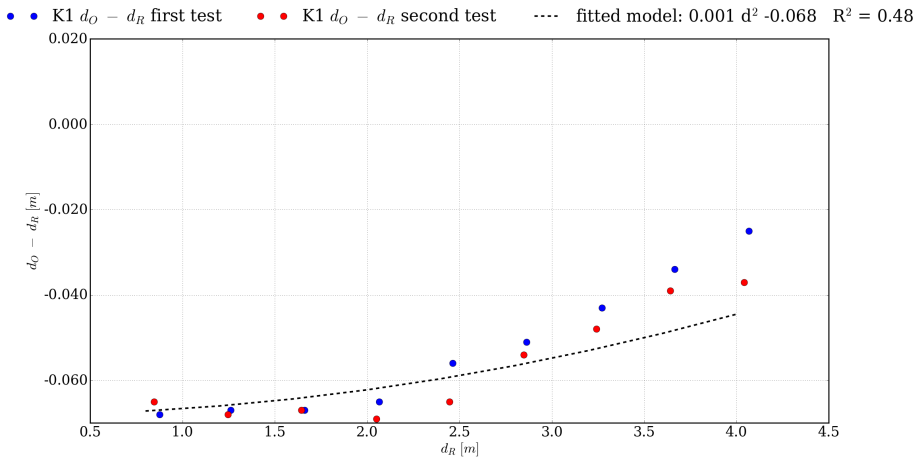


Fig. 3.16: Kinect v1 accuracy vs. depth.

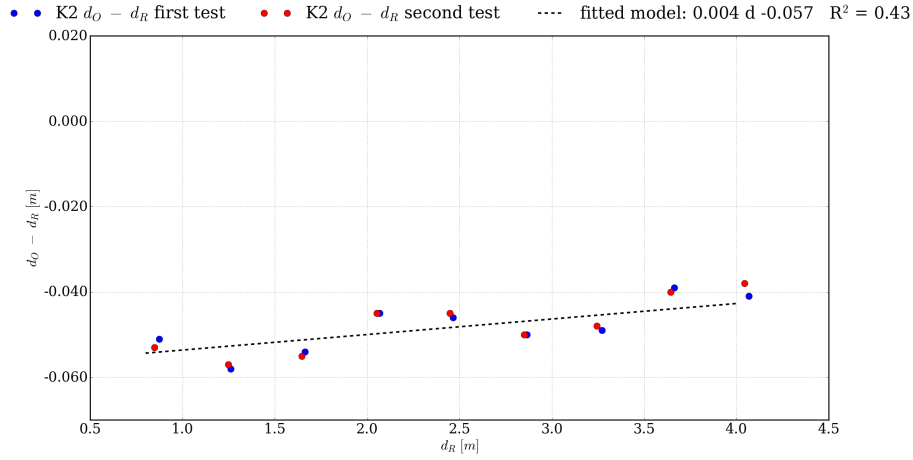


Fig. 3.17: Kinect v2 accuracy vs. depth.

versus 0.044 m of the Kinect v1 and 0.078 m of the Structure Sensor. As previously done for the precision, these accuracy trends were mod-

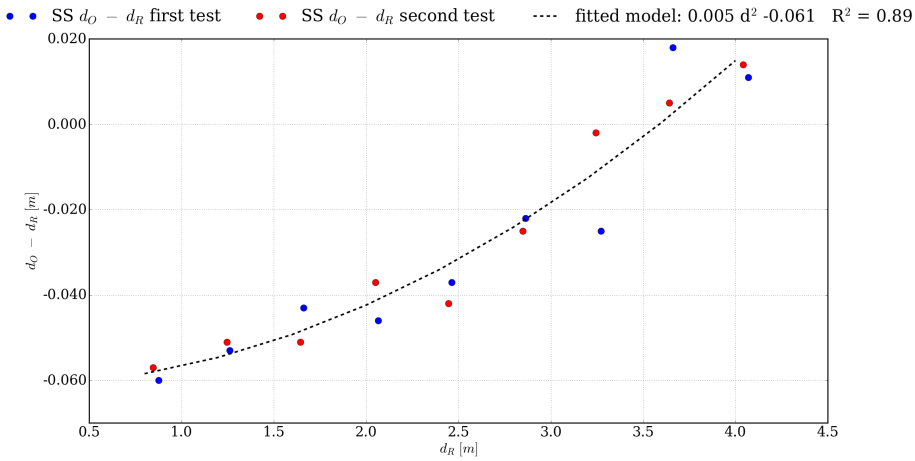
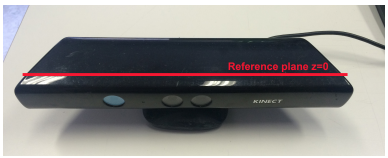


Fig. 3.18: Structure Sensor accuracy vs. depth.

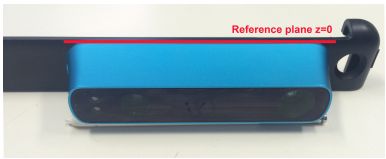
eled differently depending on the sensor operational principle (see Sec. 2.2): a linear model for the Kinect v2 and a quadratic model (with the minimum constrained at $d_R = 0$) for the Kinect v1 and the Structure Sensor were chosen; the zero order parameter (error at $d_R = 0$) represents the contribution of more offsets, precisely the offset of the total station distancemeter and the internal offset relative to the nominal reference of the depth measurement with respect to each sensor body (Fig. 3.19a, Fig. 3.19b, Fig. 3.19c). The weighted least squares were used to estimate the model parameters, with the weights based on the previously computed precisions.

Even if the regression coefficient R^2 shows lower values than before, respectively of 0.48, 0.43 and 0.89 for the Kinect v1, the Kinect v2 and the Structure Sensor, the effectiveness of such accuracy models for calibration was tested by correcting the measured depths. In particular



(a) The Kinect v1.

(b) The Kinect v2.



(c) The Structure Sensor.

Fig. 3.19: The internal offsets of the investigatd sensors.

it was checked if the residual errors after calibration were compliant and represented by the precision vs. depth models previously determined (Fig. 3.11, Fig. 3.13 and Fig. 3.12; Tab. 3.2).

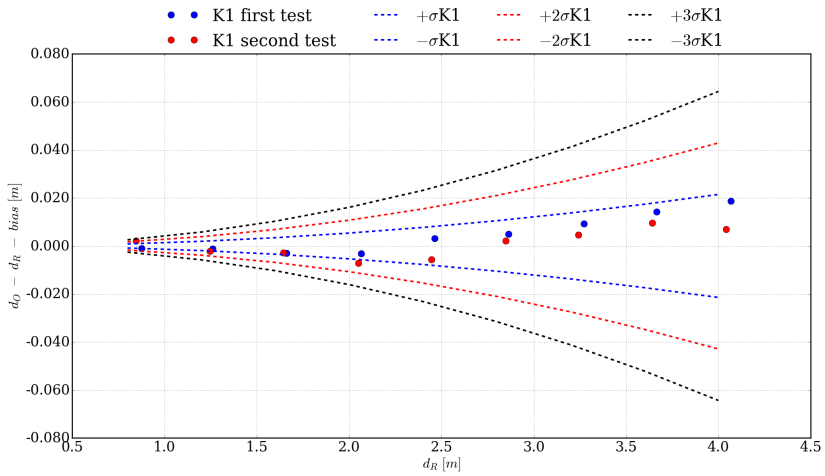


Fig. 3.20: Kinect v1 residual errors.

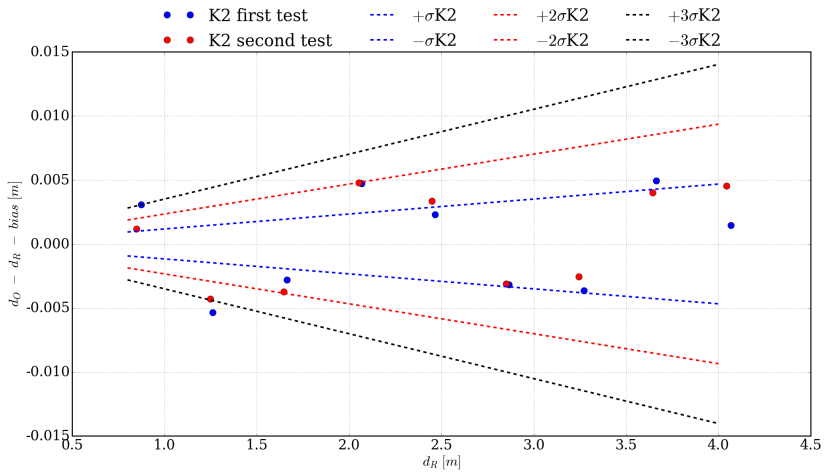


Fig. 3.21: Kinect v2 residual errors.

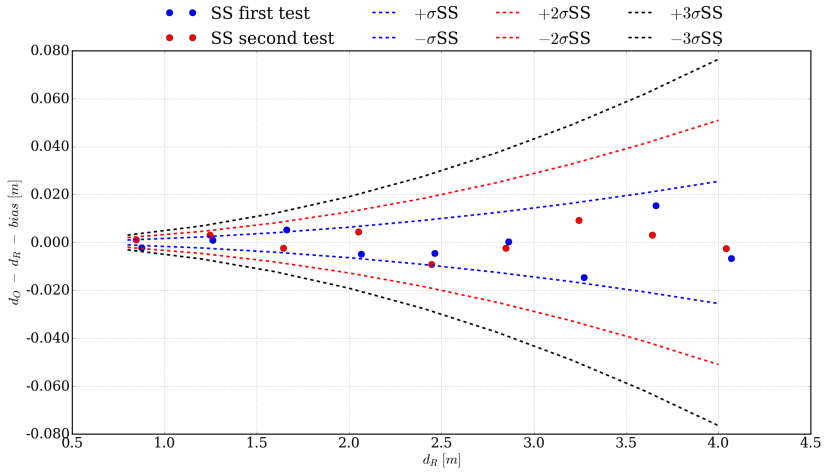


Fig. 3.22: Structure Sensor residual errors.

All the residual errors are within the $(-3\sigma, +3\sigma)$ range for each sensor and each depth (with the exception of just one residual for Kinect v2), being sigma the precision derived from these models (Fig.

3.20, Fig. 3.21 and Fig. 3.22), so that it is reasonable to conclude that the calibration has been effective by far.

Therefore the estimated models of the accuracy vs. depth can be used to correct the measured depths in such a way as to calibrate the sensors, thus enabling their use in geomatic applications. Anyway, before effectively adopting them, a more accurate estimate of the constant representing the internal offset of the sensor is necessary, as described in the next paragraph.

	Resolution [m]	Precision [m]	Accuracy [m]
Kinect v1	0.0027 d^2 (R^2=1.0)	0.0013 d^2 (R^2=1.0)	0.0015 d^2 - 0.0681 (R^2=0.48)
Kinect v2	0.001	0.0012 d (R^2=0.96)	0.0036 d - 0.0572 (R^2=0.43)
Structure Sensor	0.0032 d^2 (R^2=1.0)	0.0016 d^2 (R^2=1.0)	0.0048 d^2 - 0.0615 (R^2=0.89)

Tab. 3.2: Models for depth resolution, precision and accuracy (depth d in meters).

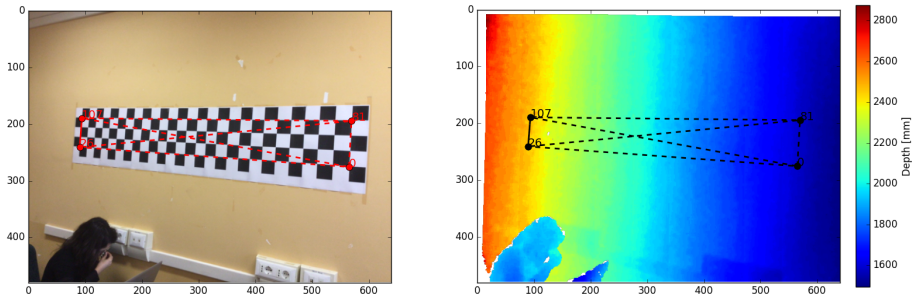
3.3.4 Validation of the proposed calibration models

To validate the found calibration models, further tests were performed with the Structure Sensor. The aim was to compare the six known distances among the four external vertexes of a rectangular checkerboard grid, with the same distances measured with the sensor, before and after having applied the depth calibration model.

Specifically, nine validation tests were performed and the grid was captured by the Structure Sensor at various distances and with different orientations, both perpendicular and tilted. In particular, in test 3 (Fig. 3.25) and test 4 (Fig. 3.26) the sensor was approximately perpendicular to the grid surface, which was acquired respectively at

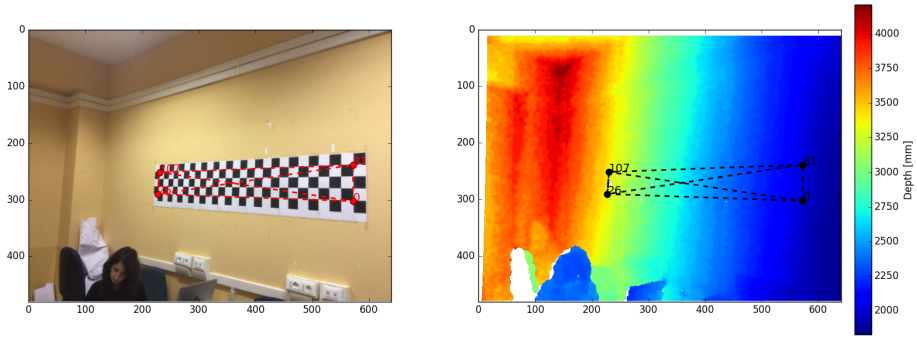
about 2 m and 4 m. In the first case the checkerboard occupied almost the whole width of the depth image, whereas in the latter it covered a smaller area. In the remaining tests the checkerboard was slightly or strongly inclined in relation to the sensor image plane, covering different parts of the depth map.

The 28×4 grid corners were automatically detected on the color image acquired by the iPad air 2 at which the range camera was connected, developing a specific iOS application (see Sec. B.3.2). The depth image and the color image were thus co-registered using the aligned depth stream provided by the Structure SDK, in such a way that the 2D corners locations were the same on both images. Obviously the Structure Sensor was previously calibrated using the Occipital Calibrator App. In this way, it was possible to retrieve the 3D coordinates of the grid points with the Eq. 3.5 and to compute the euclidean distances among them.



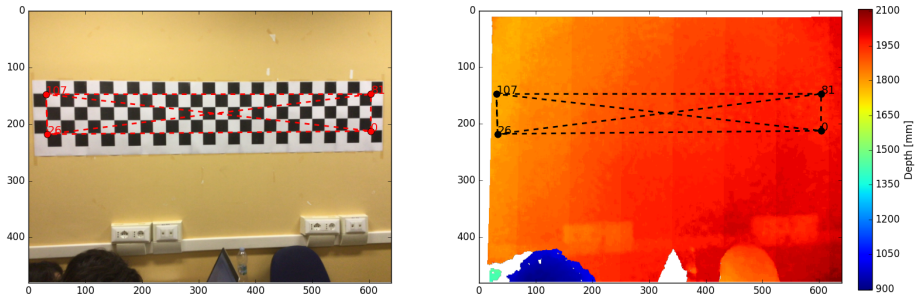
(a) Color image. (b) Depth image.

Fig. 3.23: Test 1.



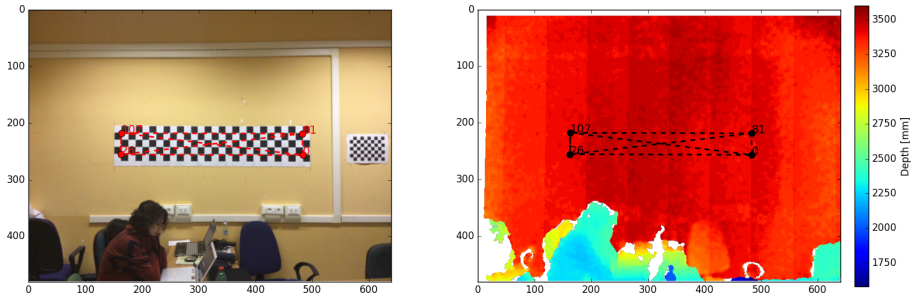
(a) Color image. (b) Depth image.

Fig. 3.24: Test 2.



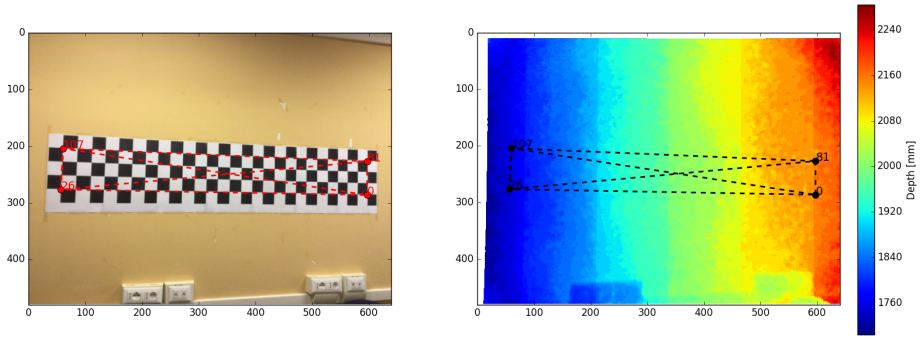
(a) Color image. (b) Depth image.

Fig. 3.25: Test 3.



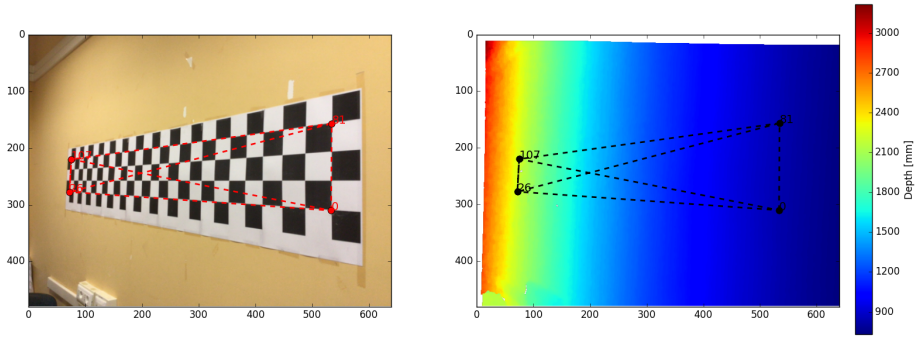
(a) Color image. (b) Depth image.

Fig. 3.26: Test 4.



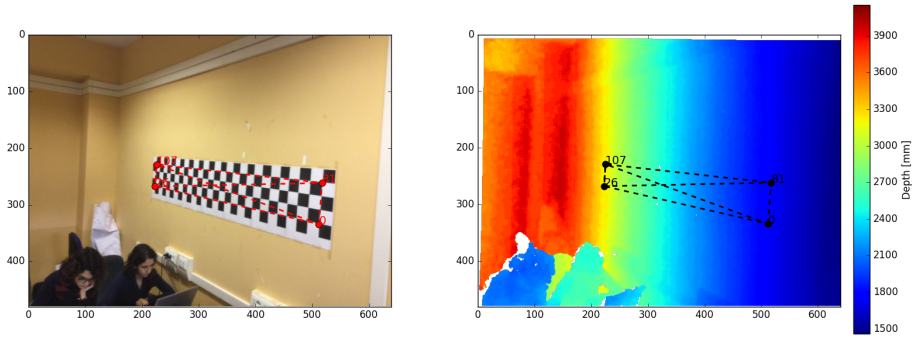
(a) Color image. (b) Depth image.

Fig. 3.27: Test 5.



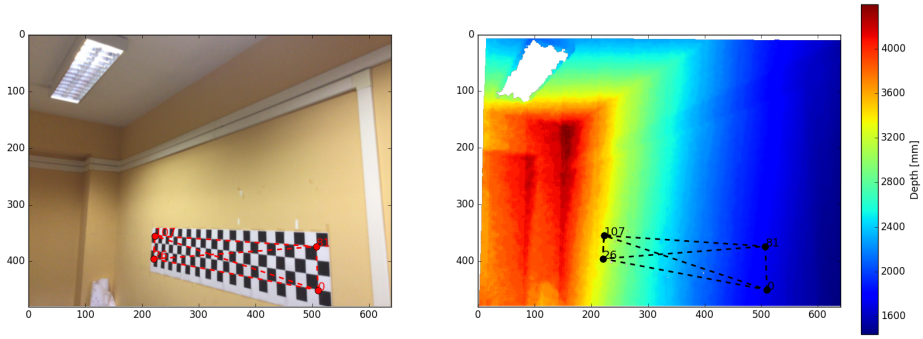
(a) Color image. (b) Depth image.

Fig. 3.28: Test 6.



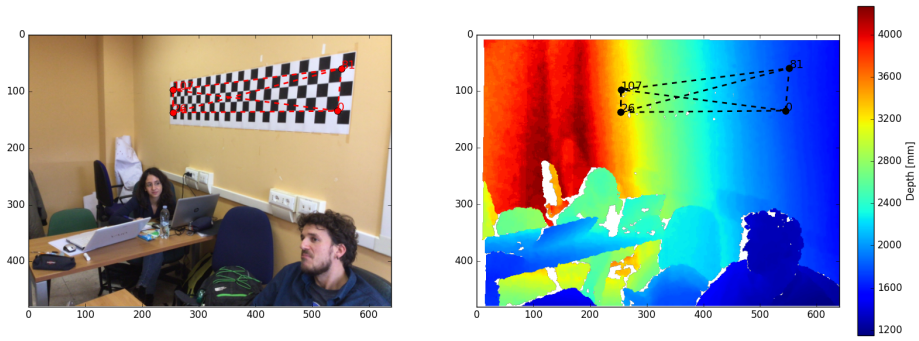
(a) Color image. (b) Depth image.

Fig. 3.29: Test 7.



(a) Color image. (b) Depth image.

Fig. 3.30: Test 8.



(a) Color image. (b) Depth image.

Fig. 3.31: Test 9.

However, before applying the found accuracy vs. depth calibration model, the zero order parameter c was re-estimated since its role is essential to describe the systematic error err that affects the depth measurement:

$$err = 0.0048\ Z_O^2 + c \tag{3.7}$$

In fact this kind error has to be removed from the depths observed Z_O by the sensor in order to properly calibrate the device:

$$Z_{CAL} = Z_O - err \tag{3.8}$$

More precisely, the c parameter was estimated in a single test, by calculating the value that minimizes the Root Mean Square Error (RMSE) of the differences between the six calibrated d_{CAL} and reference d_{REF} distances acquired:

$$RMSE = \sqrt{\frac{1}{6}\sum_{i=1}^{6}(d_{CAL,i} - d_{REF,i})^2} \tag{3.9}$$

The found value for c is 31 mm, and it was then used in the nine validation tests to calibrate the Structure Sensor.

Results are reported in Tab. 3.3: with the calibration, the overall RMSE, computed over the 9×6 distances, decreases from 27 to 16 mm. The proposed calibration model thus seems to improve effectively the accuracy of the Structure Sensor, at least for this limited number of tests.

	ID point [-]	ID point [-]	d_{REF} [mm]	d_O [mm]	d_{CAL} [mm]	$d_O - d_{REF}$ [mm]	$d_{CAL} - d_{REF}$ [mm]
Test 1	107	81	1880	1878	1877	-2	-3
	0	26	1880	1881	1880	1	0
	81	0	221	222	225	1	4
	107	26	221	220	220	-1	-1
	81	26	1892	1895	1894	3	2
	0	107	1893	1890	1889	-3	-4
Test 2	107	81	1880	1938	1916	58	36
	0	26	1880	1961	1938	81	58
	81	0	221	222	224	1	3
	107	26	221	226	225	5	4
	81	26	1892	1971	1949	79	57
	0	107	1893	1954	1931	61	38
Test 3	107	81	1880	1851	1865	-29	-15
	0	26	1880	1852	1866	-28	-14
	81	0	221	219	221	-2	0
	107	26	221	217	219	-4	-2
	81	26	1892	1869	1882	-23	-10
	0	107	1893	1860	1874	-33	-19
Test 4	107	81	1880	1889	1876	9	-4
	0	26	1880	1897	1883	17	3
	81	0	221	225	224	4	3
	107	26	221	222	220	1	-1
	81	26	1892	1906	1892	14	0
	0	107	1893	1907	1893	14	0
Test 5	107	81	1880	1966	1876	-14	-4
	0	26	1880	1883	1893	3	13
	81	0	221	223	224	2	3
	107	26	221	220	222	-1	1
	81	26	1892	1878	1888	-14	-4
	0	107	1893	1897	1906	4	13
Test 6	107	81	1880	1893	1886	13	6
	0	26	1880	1887	1882	7	2
	81	0	221	224	231	3	10
	107	26	221	223	224	2	3
	81	26	1892	1902	1897	10	5
	0	107	1893	1904	1898	11	5
Test 7	107	81	1880	1932	1903	52	23
	0	26	1880	1898	1870	18	-10
	81	0	221	223	225	2	4
	107	26	221	224	222	3	1
	81	26	1892	1906	1878	14	-14
	0	107	1893	1950	1921	57	28
Test 8	107	81	1880	1914	1884	34	4
	0	26	1880	1925	1896	45	16
	81	0	221	221	223	0	2
	107	26	221	226	225	5	4
	81	26	1892	1931	1902	39	10
	0	107	1893	1935	1905	42	12
Test 9	107	81	1880	1895	1866	15	-14
	0	26	1880	1892	1862	12	-18
	81	0	221	222	224	1	3
	107	26	221	223	222	2	1
	81	26	1892	1910	1881	18	-11
	0	107	1893	1902	1874	9	-19
					MEAN	**11**	**4**
					STD	**24**	**16**
					RMSE	**27**	**16**

Tab. 3.3: Validation results.

3.4 Investigations in color–depth alignment

To provide a complete description of an object, both the 3D geometry and the texture information are necessary. For range cameras these data are though captured from two different point of views. The alignment of color and depth images is therefore an important process in order to obtain a final 3D model characterized by high geometric accuracy and good quality texture.

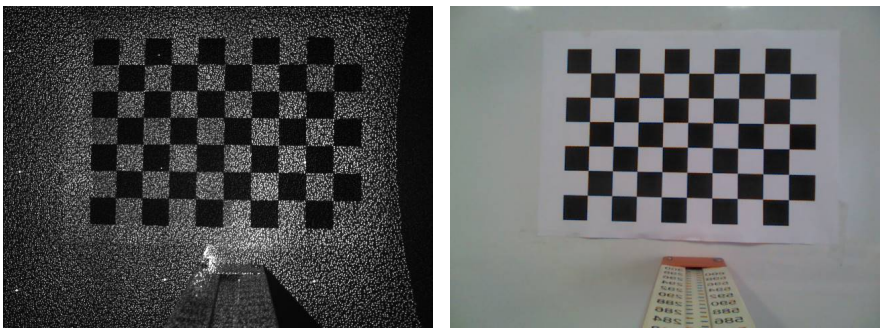
In the following sections two different investigations about this process are shortly described, respectively for the Kinect v1 and the Structure Sensor devices.

3.4.1 Kinect v1 RGB–IR shift

For the Kinect v1, the Microsoft SDK provides a specific object, the *CoordinateMapper*, able to map a color pixel into the corresponding position of the depth map. The point is that this mapper could be affected by errors and thus it is necessary to investigate its inner behaviour [35]. In particular, the *MapColorFrameToDepthFrame* method, by knowing the format of both color and depth image, maps the information contained in *depthPixels* into *depthPoints*. The former is an array of 640×480 elements where the depth information is stored, whereas the latter is an array long as the number of color image pixels, depending on the resolution (usually 1280×960).

In order to study the coordinate mapper, it was necessary to bypass it somehow, developing a specific application (see Sec. B.2) The solution was found in the infrared image. In fact, one of the formats of the color stream is the infrared with 640×480 resolution (see Fig.

3.32).



(a) (b)

Fig. 3.32: In (a) there is the IR image, while in (b) there is the same view but in the RGB format.

Since depth image and infrared image coincide (see Sec. 3.2.1), they are already aligned and, in order to retrieve the depth of a point on the IR image, it is sufficient to calculate its pixel index, i.e. $x + (y \cdot width)$, and to use it to access the correspondent element in depthPixels array, where the depth information is contained.

Notice that, to allow the automatic corner detection on the IR image, the emitter has to be disabled and afterwards activated again to allow to get depth information.

It was decided to refer directly to the image coordinates, in pixel units. For the IR image it was straightforward thanks to the unique correspondence of the images. On the other hand, to obtain comparable image coordinates for the color image, it was necessary to pass through the depthPoints array, which means through the mapper. In fact, depthPoints elements have the .x and .y properties which give back the image coordinates of the corresponding element (relatively to

Fig. 3.33: The installation used to perform the shift tests varying the distance.

the pixel index) on the depth image.

At this point, several tests were performed. First of all, it was necessary to check if the mapper applied on the infrared image gives the same result obtained by-passing it. Secondly, it was decided to verify that the mapper works well on both color image resolution. Also with a 640×480 color image the results where acceptable. Then, the possibility to oversample the infrared image was also evaluated in order to make the automatic collimation easier. Magnification factor of ×2, ×4, ×6, and ×8 were investigated and they all show the same results, unless of numeric approximation. Notice that it was necessary to divide the corners location by the resize factor to obtain the correct image coordinates.

All these tests showed an evident shift between image coordinates obtained with or without going through the mapper. According to the previous assumptions, it was therefore decided to study the shift when

	Test A		Test B	
Distance	Δx	Δy	Δx	Δy
[cm]	*pixel*	*pixel*	*pixel*	*pixel*
50	3.872	-6.318	3.842	-6.251
100	3.725	-6.458	3.628	-6.206
150	4.031	-6.600	4.150	-6.370
200	3.782	-6.609	3.689	-6.418
250	3.205	-6.656	3.393	-6.423
300	2.575	-6.770	missing	missing

Tab. 3.4: Test A, artificial light on and Test B, only sunlight.

the distance from the target increases. Initially the sensor was located at the nominal distance of 50 cm from the target, and moved in 50 cm steps up to 300 cm. For each position, the collimation was performed first on the infrared image and afterwards on the color image, without moving the Kinect v1 device.

Test A was performed with artificial light, while test B with only sunlight. All the other conditions where unchanged. A 6×9 checker board with 35.04 mm step was used. For every distance the Δx and Δy (RGB – IR) was calculated for every point, and then the average values were recorded. The results are shown in Tab. 3.4. A negative shift means that the IR–coordinate is bigger than the RGB–coordinate, and as a result there is a collimation error towards the left (x-direction) and towards up (y-direction) with respect to the color image. On the other hand, if the shift is positive, the collimation error on the color image is towards right and down respectively. From data in the Tab. 3.4 it is possible to see an almost constant negative shift of around 6 pixels in the y-direction and a positive shift of around 4 pixels in the x-direction.

3.4.2 Stereo calibration of the Structure Sensor

Since the Structure Sensor does not have its own colour camera, the 3D scanning apps leverage the iPad camera in order to retrieve the color information of the objects being scanned. Therefore, considering that the 3D geometry and the texture are captured from two different points of view, it is necessary to calibrate the precise alignment (reconstruct the geometric relationship) between the Structure Sensor and the iOS device camera in order to accurately overlap the 3D and colour data. The Occipital Calibrator app, the unique calibration app actually available on the Apple Store, can achieve this goal. It supplies already good calibration results, but its code is not open and it is designed specifically to work with the bracket accessory, which imposes a constraint for the baseline length and orientation. Furthermore it does not share the computed calibration parameters and requires the user to refine manually the calibration quality (only for the horizontal component), by touching and dragging the depth map over the colour image until they are perfectly superimposed.

For all of these reasons, it was decided to develop a specific calibration application (see Sec. B.3.1), exploiting the capabilities of the OpenCV library [79]. It implements a stereo vision calibration approach, allowing the user to automatically acquire, from different positions, several pairs of images of a chessboard grid, both with the Structure Sensor infrared camera and the iPad color camera.

In particular, the calibration grid was captured from different angles of view and in a range of distances that would have been used later for the depth map acquisition. Particular attention was paid in measuring the size of the grid squares providing the geometry scale.

The grid corners, whose 2D positions on both the IR and color images are automatically detected by the OpenCV algorithms, act as ground control points: their positions are known both in object space and image space.

In this way the application is able to compute the interior and the distortion parameters of two cameras, together with the rotation matrix and the translation vector that relate the color image to the IR/depth image (for an explanation about these parameters see Sec. 2.1.1).

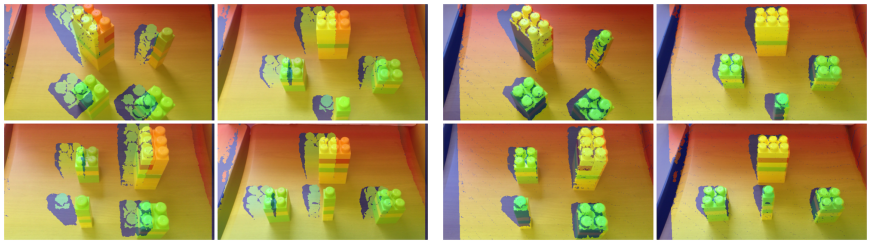
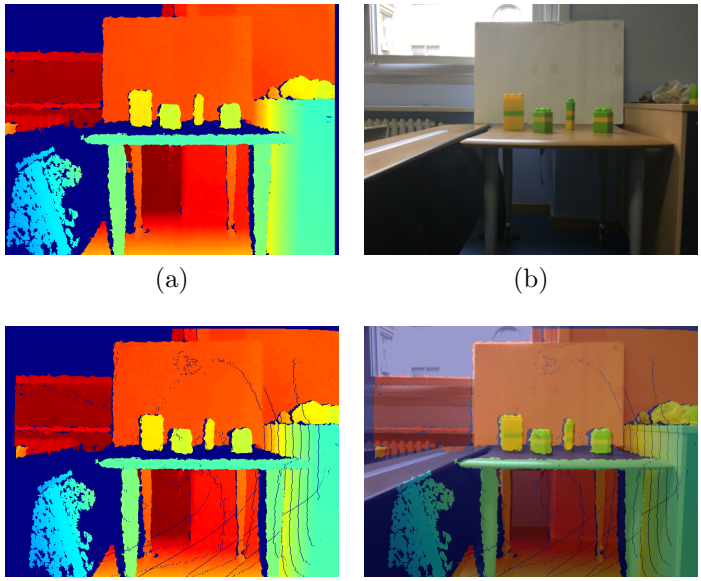


Fig. 3.34: Calibration results on simple geometry objects.

The accuracy of the calibration was measured through the reprojection error, that is the sum of squared 2D distances between the observed feature points (the 2D location of the corners) detected in the calibration images and the corresponding world points projected (using the found values for camera parameters and the poses) into the same images. The reprojection error was always lower than one pixel (it is respectively 0.32, 0.32 and 0.34 for the IR interior parameters, the color interior parameters and the stereo parameters).

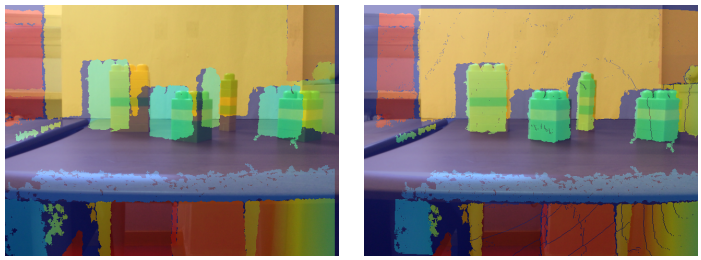
Finally the computed calibration parameters were further tested to check if they were effectively able to register the depth and colour images captured by the Structure Sensor and the iPad camera. Re-



(a) (b)

(c) (d)

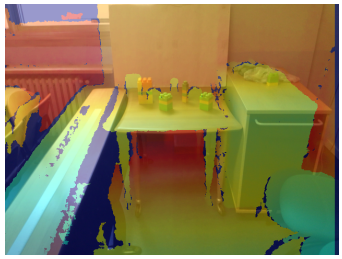
Fig. 3.35: Calibration results: (a) the captured depth image; (b) the color image; (c) the aligned depth image; (d) the color and depth images superimposed.



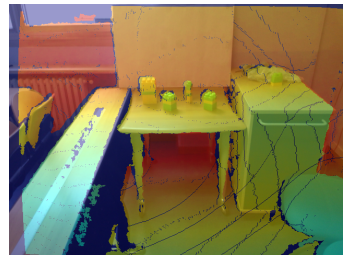
(a) Depth/color alignment before calibration

(b) Depth/color alignment after calibration

Fig. 3.36: Calibration results on a wide scene.



(a) Depth/color alignment before calibration



(b) Depth/color alignment after calibration

Fig. 3.37: Calibration results on a wide scene.

sults are reported in Fig. 3.34, Fig. 3.35, Fig. 3.36, Fig. 3.37 and show a good alignment between the two images, at least from a visual inspection.

Chapter 4

Case studies

Nowadays range camera technology is ripe for playing an important role in close-range 3D modeling. Their characteristics make these devices a suitable tool for measurement and modeling in several fields. Anyway, only the specific knowledge of the geometric quality of these sensors, investigated in the previous chapter, allows to consciously and efficiently use them in geomatic applications such as architectural surveys, documentation of cultural heritage, monitoring applications, anthropometric survey (static and dynamic), security issues related to movement recognition or target extraction, crime scene reconstruction and many more.

In this chapter four examples of the practical use regarding two of the three range cameras previously characterized are presented in the form of case studies, carried out in order to evaluate:

- the performances of the Kinect v2 sensor for monitoring oscillatory motions;
- the integration feasibility of Kinect v2 with a classical stereo

system;

- the potentialities of the Structure Sensor for the 3D surveying of indoor environments;
- the potentialities of the Structure Sensor to document archaeological small finds.

4.1 Performance of Kinect v2 for small amplitude oscillatory motion monitoring

The case study [19] described in this paragraph analyzes the performances of the Kinect v2 sensor for monitoring oscillatory motions characterized by small challenging amplitudes (0.02 m and 0.03 m) and different oscillation frequencies (in the range of 1.5–3 Hz). Amplitude and frequency accuracies for the detected positions, velocities and accelerations were evaluated with respect to the reference data provided by a Mikrotron EoSens high-resolution camera.

Although the Kinect v1 and the Kinect v2 are specifically designed for body motion tracking, the object tracking is not yet a deeply studied topic. At present, the Kinect v1 sensor has been already used for real time tracking of moving objects reaching the accuracy of few millimeters in 3D position detection ([40], [77]), whereas no particular attention was paid to both velocity and acceleration measurements. On the other hand, [85] investigated the use of Kinect v1 for monitoring the deflection of reinforced concrete beams subjected to cyclic

loads, measuring vertical displacements. Moreover, to our knowledge, the Kinect v2 sensor was never tested for object tracking up to now.

This case study is organized as follows. In Sec. 4.1.1, the main features of the tracking software are shortly presented, together with details about the vibrating table and experimental design. In Sec. 4.1.2 the data processing approach is illustrated and the obtained results are discussed. Finally, in Sec. 4.1.3 some conclusions and future prospects are outlined.

4.1.1 Experiments: devices and tools

The equipment involved in the experimental investigations consists of one Microsoft Kinect v2 range camera and one Mikrotron EoSens high-resolution camera as reference. Several tests were performed using a one-direction vibrating table at which a chessboard target was connected in order to allow both cameras to track the moving object (see Fig. 4.1).

In particular, a dedicated software tool (see Sec. B.2) was developed with the Microsoft Kinect for Windows SDK v2.0 to retrieve data from the sensor. It is based on both the depth and color video streams, and it makes possible to capture in *real-time* the 3D position of the edges of a moving chessboard grid target (see Fig. 4.1) for each frame, while preserving the native acquisition rate (30 Hz). It is worth noting that a good visibility is needed to capture satisfying texture, but at the same time direct sunlight must be avoided since depth data cannot be acquired in these conditions.

Reference data were provided by an acquisition system consisting of a high-speed, high-resolution camera (Mikrotron EoSens) equipped

with a Nikon 50-mm focal length lens capturing gray-scale images at up to 500 fps with a resolution of 1280×1024 pixels (for the present set of measurements, images were acquired at 250 fps and 100 fps) and a high-speed Camera Link digital video recorder operating in Full configuration (IO Industries DVR Express® Core) to manage data acquisition and storage.

The native kinematic parameters retrieved by the sensors are the same: displacements for both the Mikrotron EoSens camera and the Kinect v2. However the acquisition rates are remarkably different: up to 250 Hz for Mikrotron EoSens camera and 30 Hz for Kinect v2. In Table 4.1 the acquisition rate used during the tests, the kinematic parameter supplied by each sensor and its paid cost are summarized.

	Kinect v2	**Mik. EoS. camera**
Acquisition rate (Hz)	30	100-250
Native kinematic parameter	Displacements	Displacements
Cost (€)	200	10000

Tab. 4.1: Acquisition rate and kinematic parameter captured by each sensor.

Both the Kinect v2 range camera and the Mikrotron EoSens camera were placed at a distance of about one meter from the table, with the optical axis of both cameras orthogonal to the target. The orthogonality was checked with a laser pointer. Both sensors were connected to a lap-top for storing the acquired observations.

Two oscillation amplitudes (0.02 m and 0.03 m) were tested. For each amplitude, four oscillations frequencies ($f_1 \simeq 1.7$ Hz, $f_2 \simeq 2.0$ Hz, $f_3 \simeq 2.2$ Hz, $f_4 \simeq 2.7$ Hz) were set, each kept constant for approximately 15 seconds. Oscillation frequencies were roughly set through the vibrating table controller (potentiometer). The values of those fre-

quencies were determined by analysing the high temporal resolution data acquired with the Mikrotron camera.

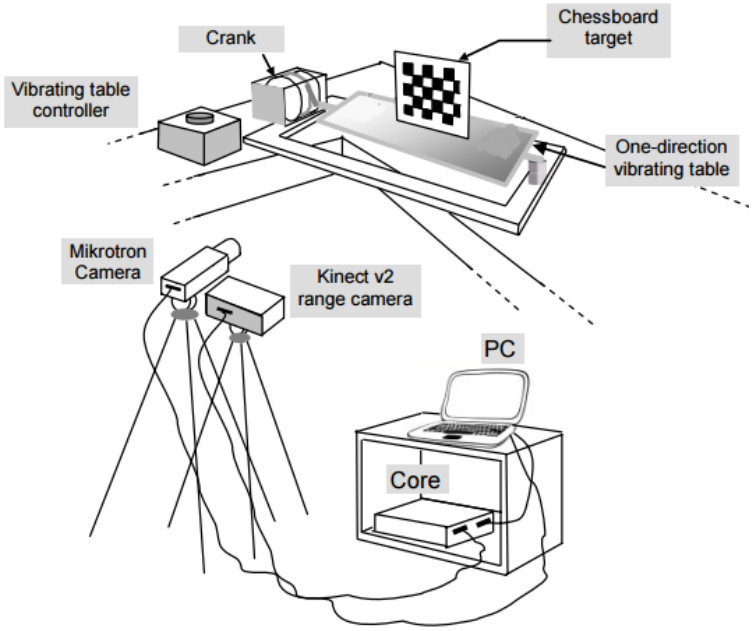


Fig. 4.1: Vibrating table equipped with target and sensors.

4.1.2 Analysis of results: methodology and discussion

Displacements, velocities and accelerations of the vibrating table monitored by the Microsoft Kinect v2 range camera were compared to those recorded by the Mikrotron EoSens high-resolution camera.

The images acquired by the Mikrotron EoSens camera were post-processed using a Lagrangian Particle Tracking technique named Hy-

brid Lagrangian Particle Tracking (HLPT) [100], which selects image features and tracks them from frame to frame.

Once the trajectories are reconstructed, displacements, velocities, and accelerations are computed via central differences. Displacement, velocity and acceleration components belonging to the same frame are arithmetically averaged to compute their time history. To characterize the reference signal, the standard deviations of its amplitude were computed by averaging the detected amplitudes for the entire signal. The mean amplitude for 0.02 cm amplitude test turned out to be 0.0199 m with a standard deviation of 0.0001 m; for 0.03 cm amplitude the mean is equal to 0.0299 m with a standard deviation of 0.0002.

The Fast Fourier Transform (FFT) was then employed to identify the four different oscillation frequencies of the vibrating table on displacement data. It is evident that the vibrating table is only roughly a harmonic oscillator, so the frequency peaks are identifiable but they are not perfectly separated from each other.

The same procedure was also applied to the raw data acquired by the Kinect v2. Results are presented in Fig. 4.2 and Fig. 4.3 and show that the range camera failed in the test at the fourth frequency with amplitude 0.03 m.

In particular, to study the four main peaks, the spectra acquired by the Kinect v2 was divided into four intervals (hereinafter subtests) and for each interval a passband filter was applied in order to better analyse the kinematic parameters of each subset; the band width was selected analysing the peaks of the Mikrotron EoSens high-resolution camera power spectra. Successively, the filtered results were resampled at 100 Hz through cubic splines to facilitate the comparison and the synchronization with reference data. It is worth noting that the results

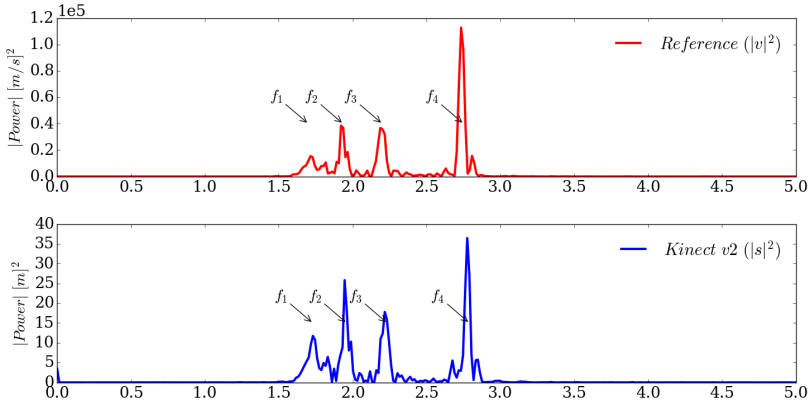


Fig. 4.2: Power spectrum of the results related to the test with 0.02 m oscillation amplitude: velocities for the Mikrotron EoSens camera and displacements for the Kinect v2 range camera.

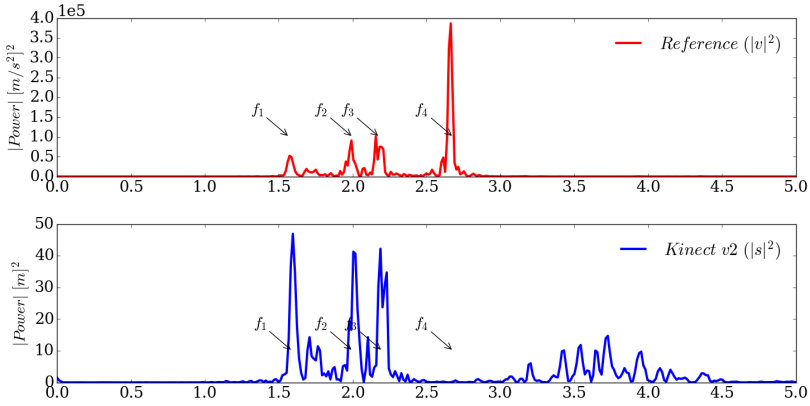


Fig. 4.3: Power spectrum of the results related to the test with 0.03 m oscillation amplitude: velocities for the Mikrotron EoSens camera and displacements for the Kinect v2 range camera.

obtained by processing the Mikrotron EoSens camera data at 100 Hz and 250 Hz were comparable. For these reasons only the results at 100 Hz are presented. Fig. 4.4 shows the results obtained for the lowest frequency (f_1) and 0.02 m oscillation amplitude in the displacement domain.

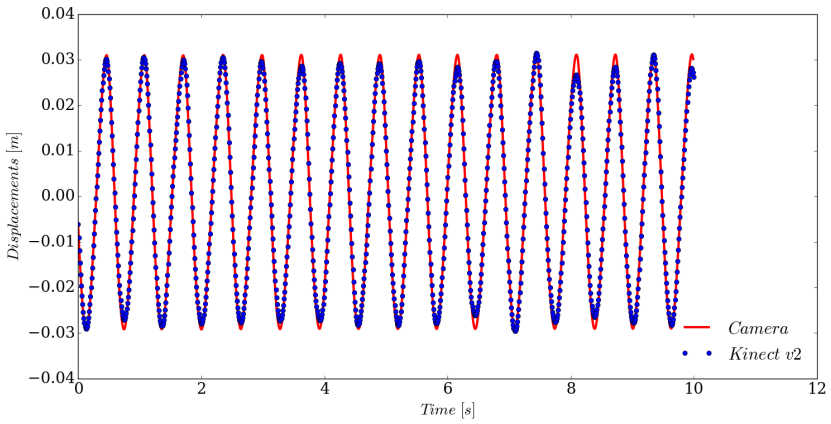


Fig. 4.4: Displacements retrieved with the Kinect v2 sensor in comparison with the Mikrotron EoSens camera for the lowest frequency (f_1) and 0.03 m oscillation amplitude.

The quantitative measure of the similarity among the kinematic parameters of the Kinect v2 sensor and reference data is the RMSE defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(kp_{K2,i} - kp_{ref,i})^2} \tag{4.1}$$

where N is the amount of data available within each subtest, kp_{K2} is the kinematic parameter monitored by the Kinect v2 sensor, kp_{ref} is the kinematic parameter detected with the Mikrotron camera.

To compute the RMSE, it was necessary to synchronize the time scales, that were approximately aligned through cross-correlation. Then the synchronization was improved using a linear interpolation, whose slope coefficient was calculated by comparing the zero-crossing times of the Mikrotron EoSens high-resolution camera with the corresponding zero-crossing times of Kinect v2 sensor. The RMSE was not calculated for all the differences, but only on the LE95 population.

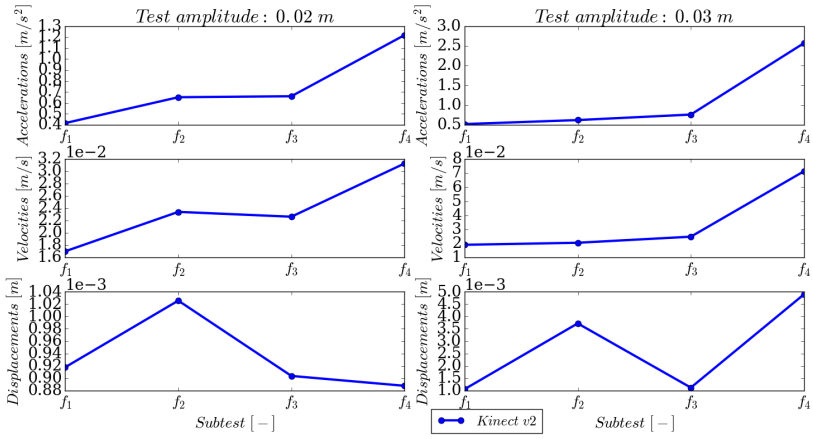


Fig. 4.5: RMSE trend of the kinematic parameters retrieved by Kinect v2 in the performed tests.

Results are summarized in Tab. 4.2 and Tab. 4.3 where mean and standard deviations of the residuals were reported as well. Fig. 4.5 shows the RMSE trend of the kinematic parameters retrieved by the Kinect v2 range camera as a function of the vibrating table oscillation frequency and amplitude. The RMSE shows a generally increasing trend where, as expected, the maximum value is reached in the test with oscillation amplitude of 0.03 m and frequency f_4, which was not properly identified (see Fig. 4.3); in addition, both Table 4.3 and Fig.

4.3 show that Kinect v2 failed during the second frequency test with 0.03 m amplitude, probably due to tracking algorithm errors.

Tab. 4.2 and Tab. 4.3 show also the results of the correlation analysis aimed at obtaining the R^2 parameter, computed with the least squares regression method. To do so, for each amplitude and frequency, kinematic parameters detected or derived from Kinect v2 sensor acquisitions were drawn in a 2D plot vs. reference data. In particular, Fig. 4.6 shows the results for the 0.03 m amplitude test at the lowest frequency and the high R^2 values are representative of the effectiveness of the adopted synchronization strategy.

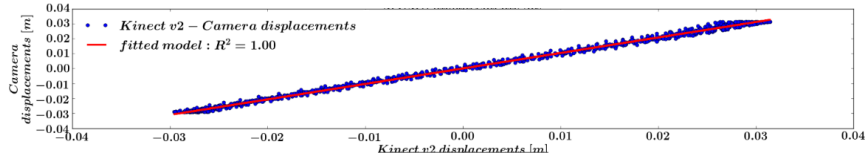


Fig. 4.6: Cross correlation between displacements retrieved with the Kinect v2 and the Mikrotron EoSens camera for the lowest frequency (f_1) and 0.03 m oscillation amplitude.

The Kinect v2 displays a rather stable noise across all the tests, but it is characterized by a high stability (lower bias) on displacements. The accuracy (RMSE) of displacements is indeed within 4 - 5% of the reference solution, except for the already mentioned two failures.

4.1.3 Conclusions and future prospects

The results obtained are promising in the prospective of employing the Kinect v2 in the field of oscillatory motions monitoring. The application fields are manifold (structural monitoring, industrial control system development, ground monitoring and so on).

	a [m/s^2]				v [m/s]				s [m]			
	RMSE	**MEAN**	**STD**	**R^2**	**RMSE**	**MEAN**	**STD**	**R^2**	**RMSE**	**MEAN**	**STD**	**R^2**
f$_1$	0.415	-0.014	0.415	0.93	0.017	0.000	0.017	0.98	0.0009	-0.0005	0.0008	0.99
f$_2$	0.651	-0.005	0.651	0.89	0.023	0.001	0.023	0.98	0.0010	0.0000	0.0010	0.99
f$_3$	0.660	0.016	0.660	0.92	0.023	0.000	0.023	0.97	0.0009	0.0001	0.0009	0.99
f$_4$	1.218	0.002	1.218	0.89	0.031	0.001	0.031	0.98	0.0009	0.0000	0.0009	0.99

Tab. 4.2: Accuracy (RMSE), Bias (Mean) and Noise (Standard deviation) in test with 0.02 m amplitude.

	a [m/s^2]				v [m/s]				s [m]			
	RMSE	**MEAN**	**STD**	**R^2**	**RMSE**	**MEAN**	**STD**	**R^2**	**RMSE**	**MEAN**	**STD**	**R^2**
f$_1$	0.517	0.000	0.517	0.94	0.019	0.003	0.019	0.99	0.0012	0.0000	0.0012	1.00
f$_2$	0.619	0.018	0.619	0.96	0.021	0.002	0.020	0.99	0.0037	-0.0003	0.0037	0.93
f$_3$	0.758	-0.002	0.758	0.96	0.025	-0.001	0.025	0.99	0.0011	0.0001	0.0011	0.99
f$_4$	2.569	0.980	2.375	0.82	0.071	0.032	0.064	0.94	0.0049	-0.0015	0.0047	0.24

Tab. 4.3: Accuracy (RMSE), Noise (Standard deviation) and Bias (Mean) in test with 0.03 m amplitude.

As regards future prospects and possible improvements, some items can be addressed:

- the upgrading of the Kinect v2 tracking tool, improving the target automatic collimation by optimizing real-time data management in order to avoid failures (as happened for the f_2 and f_4 frequencies of the test at 0.03 m amplitude);
- the possibility of tracking different targets simultaneously with the Kinect v2 must be considered, together with the possibility to use the Kinect v2 reference frame with axes directed independently from the object to be monitored (in our tests the optical axis was aligned orthogonally to the object motion direction);
- the repetition of the tests over longer periods, in order to investigate the effectiveness of the synchronization procedure and possibly to refine it.

4.2 Kinect v2 and RGB stereo cameras integration for point cloud enhancement: a first test

This second case study [90] was performed to evaluate the integration feasibility of range camera technology with a classical stereo system. In fact, the integration between these two methods can offer many advantages since they are characterized by complementary features.

On the one side, range cameras are low-cost and easy to use imaging sensors, able to measure distances of the scanned scene at high frame

rate and to easily collect dense point clouds practically in real time. At the same time, though, these devices show issues with transparent and very reflective surfaces. Furthermore, the depth maps obtained are generally noisy and the finer details of the resulting 3D models are often smoothed.

On the other side, stereo vision is an established technique, but the resulting 3D models are mostly incomplete in low texture regions. In addition, the processing needs an external scale that the user must provide and it is often computationally onerous and time consuming.

Therefore the leading idea, which will be developed in future works, is that a preliminary depth map of the investigated object can be obtained in real-time through a low-cost range camera. This depth map will be employed as a coarse 3D model for classical stereo processing, which will add the details coming from the stereo images acquired through standard cameras.

In details, the coarse depth map acquired by the range camera will be the geometrical constraint for the subsequent Semi Global Matching (SGM) algorithm that will compute the stereo disparity map. In this way the efficiency of the dense matching algorithm will be increased.

This case study is organized as follows. In Sec. 4.2.1 a short review of the state-of-the art is illustrated; in Sec. 4.2.2 this first test is described and its results are discussed in Sec. 4.2.3. Finally some conclusions are outlined in 4.2.4.

4.2.1 Related works

The topic of integration between products from range and RGB cameras has been investigated for several years and a substantial literature

is available [76], mainly considering middle to high cost professional sensors. The goal of this case study is to reconsider the already obtained methodological results under the light of new available low-cost sensors that can be integrated in a flexible solution. [109] combines a professional ToF sensor with two CCD cameras, introducing a method for improving the range camera manufacturer calibration. [107] proposes a system formed by a professional three stereo camera and a low-cost SL range camera and suggests an accuracy improvement of the resulting 3D model through a stereo visual odometry integration. [39] presents a high-resolution stereo matching algorithm guided by low-resolution depth data, that helps the algorithm to compensate for its difficulty in estimating disparities over weakly textured areas.

4.2.2 Discussion

For the purposes of this study, a 3D model of a DUPLO^{TM} bricks construction was reconstructed both with the Kinect v2 range camera and by processing one stereo pair acquired with a Canon EoS 1200D DSLR camera. The two 3D models were then fused, obtaining the integrated model.

A specific software tool (see Sec. B.2) was developed to download the 3D data with the Microsoft SDK and the model point cloud was reconstructed from the depth map (see Fig. 4.7) acquired in a single frame, since the final aim is the near real time integration.

Regarding the stereo model, one stereo pair was acquired with a proper stereo baseline (base-to-height ratio of about 0.3) and it was processed with the Agisoft PhotoScan^{TM} photogrammetric software [15]. The approximated model scale was estimated by imposing the

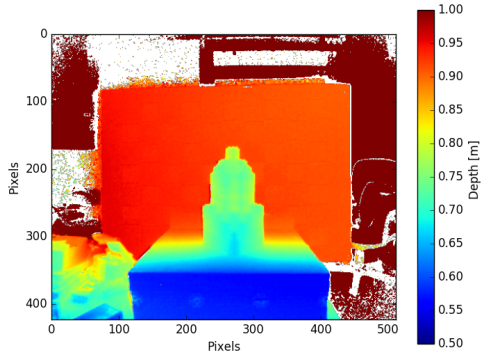
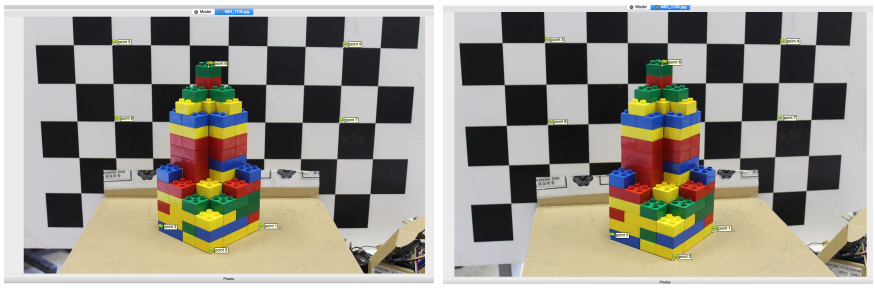


Fig. 4.7: Depth map acquired by the Kinect v2 range camera.

values of the coordinates measured by the Kinect v2 to eight points collimated on both images (see Fig. 4.8).



(a) First image (b) Second image

Fig. 4.8: Acquired stereo pair.

The fusion of the Kinect v2 and the photogrammetric models was performed trough the CloudCompare [47] 3D point cloud and mesh processing software. Since the two points clouds were already in the same reference system, the co-registration was only refined using the Iterative Closest Point (ICP) algorithm [21], which estimated the parameters of the residual roto-translation (with scale) transformation.

4.2.3 Results

To assess the metric quality of the results obtained, both the integrated model and the models reconstructed with the single 3D modeling technique were compared with the reference model of the DUPLO bricks construction.

The dimensions of bricks were measured with a vernier caliper and the reference mesh model (see Fig. 4.9) was reconstructed with a standard CAD software.

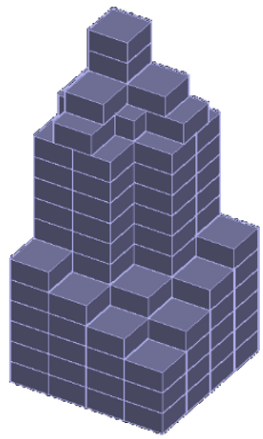


Fig. 4.9: CAD reference model.

In particular, precision and accuracy were evaluated in terms of signed distances (positive inside and negative outside the reference mesh surface) of the 3D model points from the reference mesh.

Concerning the completeness assessment, it was based on the criterion of 2D grid occupancy: the point cloud was projected onto a regular grid mesh (with a posting of 0.004 m $\times$ 0.004 m) and the completeness

index was computed as the ratio of the non empty cell number to the grid total size.

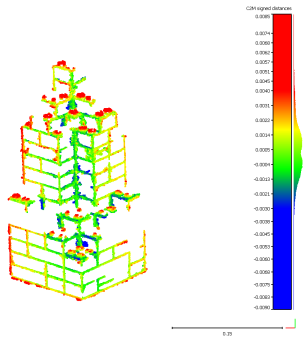
Model	Distance mean [m]	Distance std [m]	Completeness [-]
Photogrammetric	0.000	0.002	38%
Kinect v2	0.004	0.015	67%
Integrated	0.001	0.003	69%

Tab. 4.4: Distance statistics.

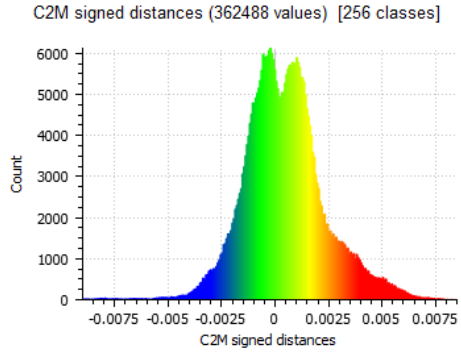
The photogrammetric model is the most accurate and precise, as the statistics of distances show (see Tab. 4.4), and it is reported in Fig. 4.10a. Anyway the borders between the bricks are reconstructed very well, thanks to the high texture variation, but the model is not complete in correspondence of the areas with uniform texture (single bricks).

As regards the model reconstructed by the Kinect v2, it is less accurate and less precise: the mean and the standard deviation of distances from the reference mesh model reach the values of 0.004 m and 0.015 m respectively. The details of the bricks are generally less recognizable (see Fig. 4.10c) and the model shows some inaccuracies (flying pixels) on the edge of the DUPLO construction, where there are high depth variations.

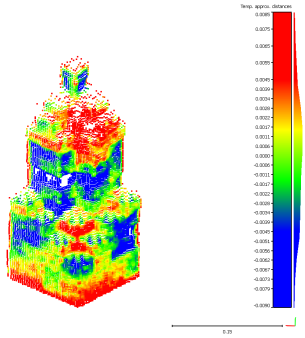
Finally the integrated model (see Fig. 4.10e) preserves the accuracy and the precision of the photogrammetric model, but it also presents the greatest level of completeness, provided by the contribution of the Kinect v2 sensor.



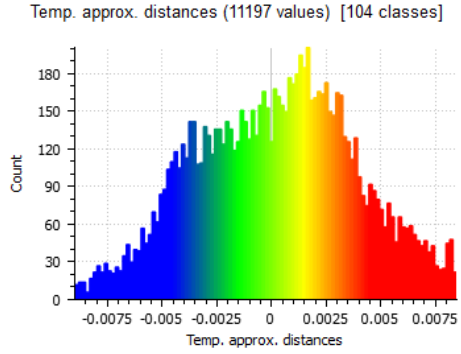
(a) Photogrammetric model results



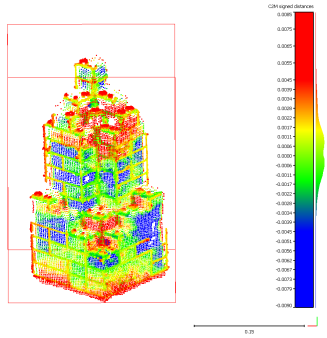
(b) Photogrammetric model distance histogram



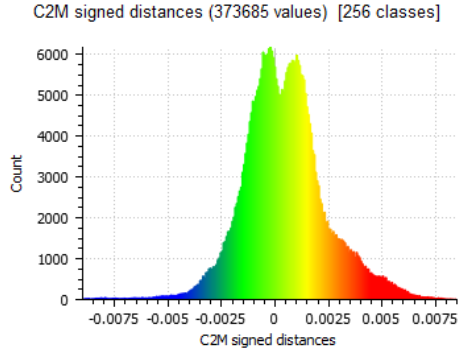
(c) Kinect v2 model results



(d) Kinect v2 model distance histogram



(e) Integrated model results



(f) Integrated model distance histogram

Fig. 4.10: Results.

4.2.4 Conclusions and further developments

The obtained results are encouraging and show that this integrated approach leads to higher metric accuracy of the final 3D model with respect to that obtained by only using a range camera and to an higher level of completeness respect to that obtained by only processing a stereo image pair.

Future works will deeply investigate the effects of distance from the object to be scanned and automate the processing procedure by implementing the method previously described.

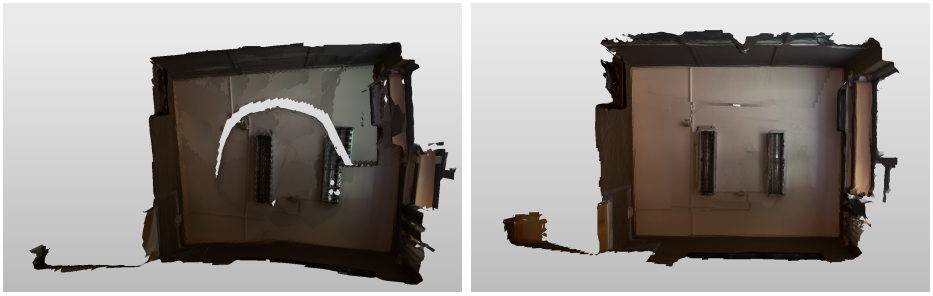
4.3 Near real time indoor mapping with the Structure Sensor

The case study illustrated in this paragraph analyzes the potentialities of the Structure Sensor for the 3D surveying of indoor environments. The specific aim is to evaluate its accuracy in reconstructing near real-time planimetric maps of interiors of buildings [86]. The Room Capture application (see Sec. A.3.5.1) was therefore used to acquire the 3D models of the ceilings of three rooms (*Aula piccola*, *Aula tesisti* and *Aula grande*) with different shapes and sizes.

4.3.1 Data collection

First of all it was necessary to identify a general procedure to be followed in order to obtain good quality 3D models of the rooms with the Structure Sensor.

In fact the tracking process can fail, causing an incorrect estimation of the sensor motion during the scanning. The incremental alignment of successive frames can indeed generate a progressive accumulation of distortions along the scan path which results in a wrong estimation of the sensor trajectory. When, at the end of the scanning, previously visited areas are re-captured, they are placed in the wrong global location, and the estimated trajectory does not close. This may lead to a severe corruption of the final 3D model, whose reconstructed 3D geometry does not correspond in this case to the actual shape of the scanned scene (see Fig. 4.11).



(a) Uncorrect tracking (b) Correct tracking

Fig. 4.11: 3D model.

It is therefore indispensable to limit all the unnecessary sensor movements that can bother the tracking process. The 3D models were thus acquired with the operator seated on a swivel chair placed in the middle of the room, with the sensor maintained as close as possible to the rotation axis of the chair. In this way the translation movements were minimized, and the path traveled by the device was mostly circular, as Occipital itself recommends. Finally the dimensions of the

scanning volume were set in such a way as to allow the maximum coverage of the upper part of the walls with a minimum upward tilt of the sensor in order to collect ceiling information.

4.3.2 Processing of the 3D models

All of the three rooms were captured with the circular scanning approach described in the previous paragraph. The resulting 3D models were then processed to obtain a planimetric map allowing to uniquely measure the principal quantities (sides, perimeter and area) of the scanned rooms. The aim was to compare them with the ones obtained through a traditional survey performed with a measuring tape. Several 3D models were acquired for the same room. In particular each 3D model was cut through the Slic3r open source software along a section, parallel to the ceiling plane, and very close to the ceiling itself (see Fig. 4.12).

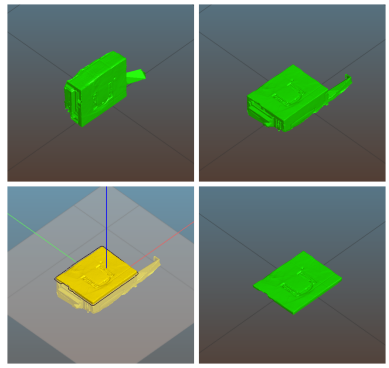


Fig. 4.12: Cutting of the model.

Fig. 4.14, Fig. 4.15 and Fig. 4.16 show an example of the obtained planimetric maps for the three rooms.

To evaluate the accuracy, for each room model, the differences between the lengths of the sides measured with the Structure Sensor and those obtained through the traditional survey were firstly computed. Successively the mean and the standard deviation of the residuals were computed over all the sides. In this way it was possible to calculate the RMSE for each model, representing its accuracy. The obtained results are reported in Fig. 4.13 and in Tab. 4.5. The RMSE varies in the range of 3 - 10 cm, very low values if compared to the dimensions of the sides of the room. The achievable degree of accuracy is therefore better than 10 cm. Furthermore, results highlight a slight trend as a function of the dimensions of the rooms, generating better 3D models for smaller rooms.

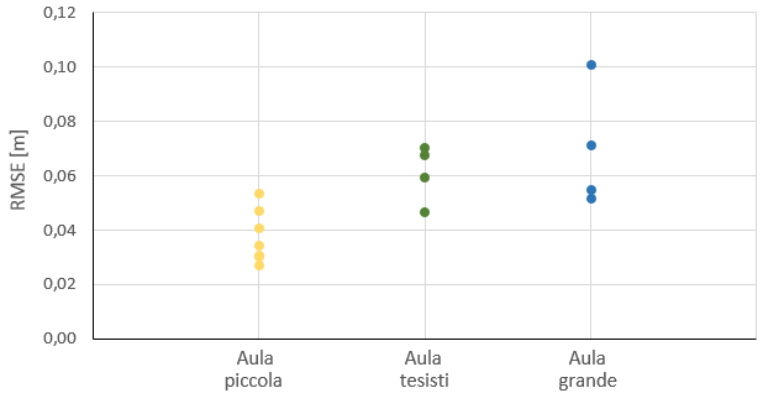


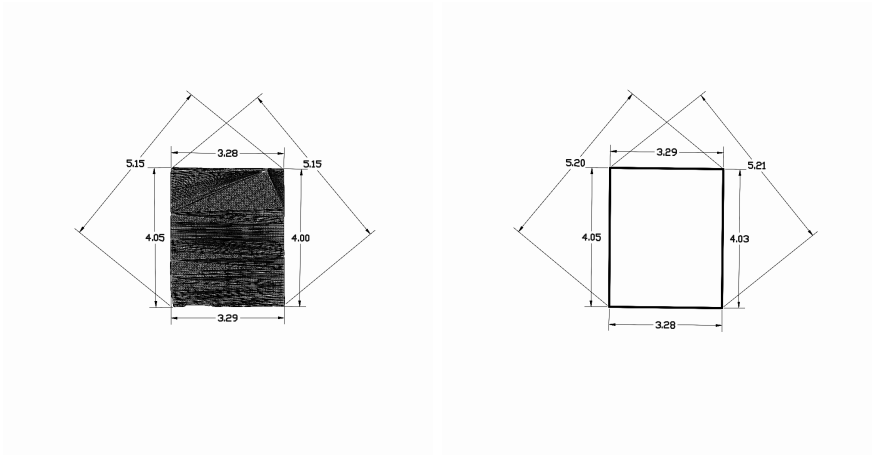
Fig. 4.13: RMSE over the length of the sides for the obtained planimetric maps.

The obtained results thus show that it is effectively possible to use the Structure Sensor in order to compute planimetric maps of rooms in scale 1:200. In fact the reconstructed planimetric layouts show a mean accuracy of 5 cm, 10 cm in the worst case, and thus the error is

	RMSE [m]		
	Aula Piccola	*Aula Tesisti*	*Aula Grande*
Model 1	0,03	0,06	0,05
Model 2	0,05	0,05	0,07
Model 3	0,03	0,07	0,05
Model 4	0,05	0,07	0,10
Model 5	0,04	–	–
Model 6	0,03	–	–
Model 7	0,03	–	–
Mean	0,04	0,06	0,06

Tab. 4.5: RMSE over the length of the sides for the obtained planimetric maps.

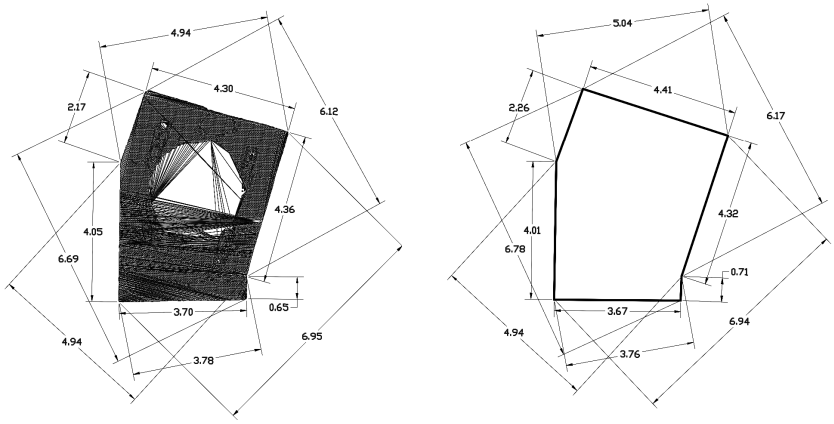
still acceptable for many applications not demanding for centimetric accuracy.



(a) Structure Sensor: Perimeter = 14.61 m; Area = 13.20 m^2

(b) Reference: Perimeter = 14.65 m; Area = 13.26 m^2

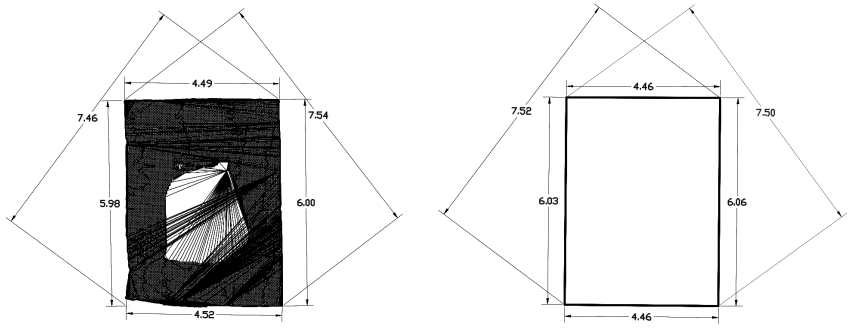
Fig. 4.14: Planimetric maps of *Aula piccola* room.



(a) Structure Sensor: Perimeter = 19.29 m; Area = 23.19 m^2

(b) Reference: Perimeter = 19.37 m; Area = 23.36 m^2

Fig. 4.15: Planimetric maps of *Aula tesisti* room.



(a) Structure Sensor: Perimeter = 20.94 m; Area = 26.84 m^2

(b) Reference: Perimeter = 21.01 m; Area = 26.96 m^2

Fig. 4.16: Planimetric maps of *Aula grande* room.

4.4 Archaeological applications: catching small finds in 3D

It is well known that small finds[1] provide a variegated myriad of data of crucial importance to the study of their finding contexts. Therefore, the production of reliable documentation of small finds is a crucial process during archaeological excavations. Nowadays archaeologists usually document them in 2D by proper representations, but the full comprehension of their multiple functions is strictly dependent on the possibility of a close all-around examination. It is for this reason that the small finds documentation during excavation can be still considered an open problem.

Range cameras seem one of the most promising tools to solve this issue: they can be used to reconstruct accurate and reliable 3D models of small finds, with a reasonable little effort, making possible their systematic application on the field. Their capability to immediately capture the model in metric units, without the need of providing an external scale during the scanning, is an essential feature to document the archaeological small finds quickly, effectively and in a comprehensive way. In this section, the first applications of the Structure Sensor for scanning archaeological small finds are shortly described.

[1]Small finds is an archaeological term for artifacts discovered on excavations which are somewhat special compared with the common finds for that type site. The special nature of the find is given by the information the artifact can provide to interpret that particular site.

4.4.1 Data collection and elaboration

The Structure Sensor was used to scan several small finds coming from the excavations of the University of Rome *La Sapienza* directed by Prof. Lorenzo Nigro in the island of Motya, a Phoenician city in Western Sicily. The Scanner app was selected among all the 3D scanning apps actually available for the Structure Sensor since it is free, very easy to use and furthermore its code is open, thus customizable according to the specific requirements of the application.

For concave objects such as vases, if the size of the vase opening is big enough, it is possible to scan also interior part in order to model also the inner volume. In particular the target objects were mounted on a sort of pedestal placed on a smooth and flat surface in order to facilitate the tracking. A suitable pedestal should be thin in order to cover a minimum surface of the target object but at the same time it should be sufficiently stable to assure the object safety. Once the object was scanned, following the procedure described in Sec. A.3.1, the pedestal was removed from the final mesh for all the models and the related hole filled with a standard 3D mesh/point cloud processing software, such as CloudCompare [47]. Indeed with the Interactive Segmentation Tool it is possible to remove the points (or triangles) falling inside (or outside) the border of the 2D polygon defined interactively by the user. Then, to close the base of the model, the user can cut a thin section in correspondence of the pedestal hole with the Cross Section Tool. This operation generates a remaining cloud on which a planar mesh can be fitted with the Delaunay 2.5D Mesh Tool. Sometimes it is necessary to refine the results by translating downwards the mesh obtained with the Translate/Rotation Tool. Finally the closing mesh can be merged

to the object model.

Some of the obtained models are reported in Fig. 4.17 and Fig. 4.18.

(a) Wireframe visualization. (b) Color visualization.

Fig. 4.17: The obtained 3D model for vase T177/2.

In order to evaluate the quality of the 3D geometry reconstruction, the 3D model of a Phoenician vase captured by the Structure Sensor was compared to the 3D model of the same object obtained by processing 91 images with the Agisoft photogrammetric software [15] [65]. Since it was not possible to provide an external scale while acquiring the images, a non-scaled photogrammetric model was firstly produced. Then the scale factor was estimated by computing the ratio between the length of the same well identifiable detail measured on both the photogrammetric model and the one captured by the Structure Sensor. In particular, the length of a crack was measured in both models, and the scaling factor resulted equal to 0.068.

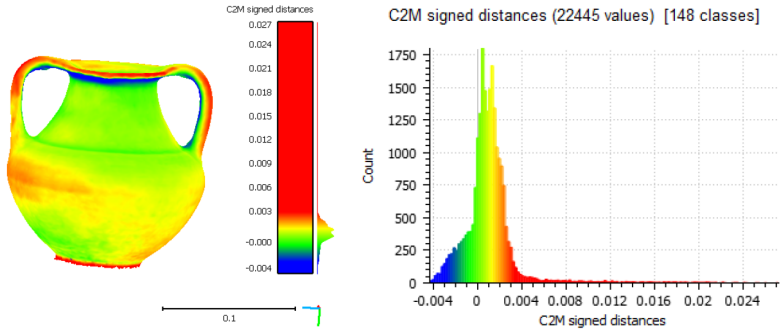
Once the photogrametric model was scaled, the meshes were reg-

(a) (b)

Fig. 4.18: The wireframe visualization of the obtained 3D models for vase (a) T180/4 and and vase (b) T181/2.

istered through the Iterative Closest Point (ICP) algorithm [21] implemented in the CloudCompare [47] software. In this way it was possible to evaluate the precision and the accuracy in terms of signed distances (positive inside and negative outside) of the points of the photogrammetric model from the mesh of the Structure Sensor model. The computed distances and the related histogram are reported in Fig. 4.19a and Fig. 4.19b.

The distance mean and standard deviation are equal to 1.0 mm and 2.4 mm, respectively. From what is visible in Fig. 4.19a, the most different areas are located in correspondence of the handles and of the base, while the remaining surfaces are generally quite similar. For what regards the base, the higher distances values are probably caused by errors in the post processing phase necessary to close the model.



(a) (b)

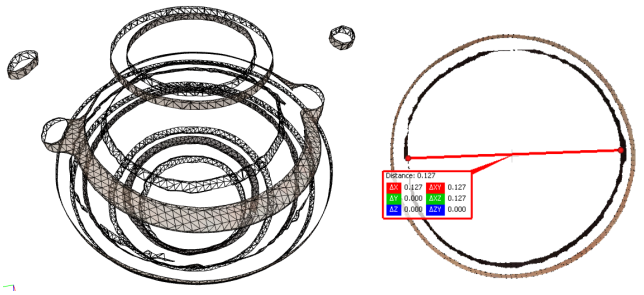
Fig. 4.19: (a) Distances between the points of the photogrammetric vase model and the mesh of the Structure Sensor model; (b) related histogram.

Concerning the texture reconstruction, it is instead less accurate, since occasionally the colour is not perfectly aligned to the 3D geometry in some areas of the model, in particular for those captured at end of the scanning process (at the end of the 360° path). This behaviour can be explained with a not perfect outcome of the calibration and/or residual tracking errors. Finally the colouring approach used by the Scanner app (the only 3D scanning app tested so far) of Occipital tends to smooth the texture details.

4.4.2 Possible applications of the obtained 3D models

The 3D model provides all the necessary information to completely describe the archaeological small finds. Furthermore it allows to take in depth a posteriori measurements, such as the volume computation and section visualization. It is important to notice that all these mea-

surements are expressed in metric units because, as mentioned above, the Structure Sensor generates 3D models that belong intrinsically to a metric space. In particular, with the CloudCompare software it is possible to cut the model in one and/or several slices through the Cross Section Tool (see Fig. 4.20) and then to measure the principal quantities (see Fig. 4.21).



(a) Sections.

(b) Diameter measurement.

Fig. 4.20: Elaboration performed on the model T177/2.

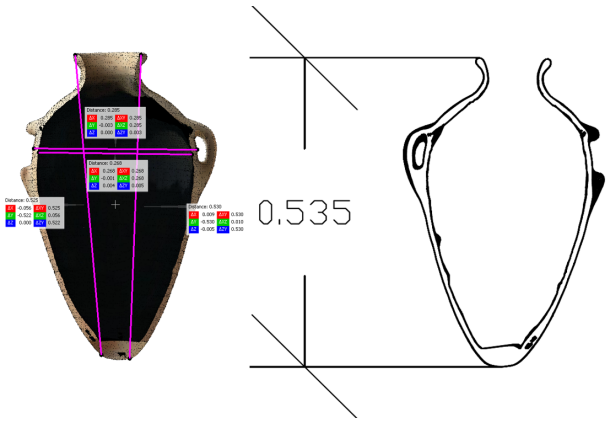


Fig. 4.21: Vertical sections on model BL1536.

Instead, to compute the volume, the Compute Geometric Measures (Quality, Measure and computation Filter) Tool of the Meshlab software can be used. Anyway it is essential to underline that only the volume of closed model (watertight mesh) can be computed. For example to compute the volume of a vase, the user must close also the higher opening, using the same procedure adopted for the pedestal hole.

Chapter 5

Conclusions

Range cameras are low-cost 3D scanners; thanks to their high frame rate they can easily collect dense point clouds in a short range (few meters) from the imaged objects. The aim of this thesis was to evaluate the potentialities of these sensors for geomatic applications and to provide useful indications for their practical use. The leading idea that guided this work is to supply a feasible and effective procedure for the calibration of range cameras, enabling their use for close-range 3D modeling of objects and environments.

Therefore the three most popular and/or promising low-cost range cameras, namely the Microsoft Kinect v1, the Micorsoft Kinect v2 and the Occipital Structure Sensor, were firstly characterized from a geomatic point of view in order to assess the metric quality of the depth data retrieved by them.

These investigations showed that such sensors present a depth precision and a depth accuracy depending both on the operational principle adopted by the single device (Structured Light or Time of Flight)

and on the depth itself.

On this basis, two different models were identified for precision and accuracy vs. depth: parabolic for the Structured Light (the Kinect v1 and the Structure Sensor) and linear for Time of Flight (the Kinect v2) sensors, respectively. Then the effectiveness of such accuracy models for calibration was tested by correcting the measured depths through the estimated parameters. The residual errors were globally compliant with the found precision models for all of the sensors. Overall, the best performances, at an accuracy level of very few millimeters, were supplied by the Kinect v2 Time of Flight sensor. In fact, for the Kinect v2 the residuals are always below 5 mm, independently from the depth/distance, while for the Kinect v1 and the Structure Sensor they are within 19 mm and 15 mm respectively, thus leading to an accuracy level around 1 cm.

Furthermore, in order to validate the found calibration models, nine additional tests were performed with the Structure Sensor. First of all, the value of the constant representing the internal offset in the accuracy model was re-estimated. Then the six known distances among the four external vertexes of a rectangular checkerboard grid were measured through the sensor, before and after having applied the depth calibration model. With calibration, the overall RMSE, computed over the 9×6 distances, decreased from 27 to 16 mm. The proposed calibration model thus seems to improve effectively the accuracy of the Structure Sensor, at least for this limited number of tests.

Successively, some investigations about the registration process of depth and color images were also performed. Specifically, for the Kinect v1 some tests were carried out in order to evaluate the effect and the performances of the aligning/mapping algorithm (provided by

the Microsoft SDK) between the two images. The goal was to investigate if the coordinates collimated on the RGB image were affected by the mapping algorithm, therefore leading to wrong depth values. As a result, an almost constant negative shift of around 6 pixels in the y direction and a positive shift of around 4 pixels in the x direction was observed at several distances from the captured object.

For the Structure Sensor, instead, a stereo calibration was carried out in order to reconstruct the geometrical relationship between the depth sensor and the color camera of the device at which it was connected. The estimated calibration parameters were effectively able to register the depth and color images captured by the Structure Sensor and the color camera.

Finally four case studies were analyzed.

In the first one, the performances of the Kinect v2 sensor for monitoring oscillatory motions characterized by small challenging amplitudes (0.02 m and 0.03 m) and different oscillation frequencies (in the range of 1.5–3 Hz) were evaluated. The Kinect v2 displayed a rather stable noise across all the tests, but it was characterized by a high stability (lower bias) on displacements. The accuracy (RMSE) of displacements was generally within 4 - 5% of the reference solution. The results obtained are thus promising in the prospective of employing the Kinect v2 in the field of oscillatory motions monitoring, such as structural monitoring, industrial control system development, ground monitoring and so on.

The second case study was performed to evaluate the integration feasibility of range camera technology with a classical stereo system. For this purpose, a 3D model of a DUPLO bricks construction was reconstructed both with the Kinect v2 and by processing one stereo

pair acquired with a Canon Eos 1200D DSLR camera. The scale of the photogrammetric model was retrieved from the coordinates measured by the Kinect v2. The results are encouraging and show that this integrated approach leads to higher metric accuracy of the final 3D model with respect to that obtained by only using a range camera and to an higher level of completeness respect to that obtained by exclusively processing a stereo image pair.

The third case study analyzed the potentialities of the Structure Sensor for the 3D surveying of indoor environments. The specific aim was to evaluate its accuracy in reconstructing near real-time planimetric maps of building interiors. The obtained results show that the reconstructed planimetric layouts are characterized by a mean accuracy of 5 cm, 10 cm in the worst case, and thus they are suitable for collecting 2D maps at a scale 1:200 and several applications not demanding for centimetric accuracy.

Finally in the last case study the first applications of the Structure Sensor for scanning archaeological small finds were described. In fact cultural heritage documentation can be one of the natural field of application of range camera technology, where speed and ease of use are predominant with respect to accuracy. A general procedure was therefore identified in order to allow not expert users to reconstruct a 3D model of archaeological small finds with a range camera. In particular several small finds were acquired with the Structure Sensor, which was able to capture well the geometry of these objects.

In conclusion, although the experimental results demonstrated that range cameras have the capability to give good and encouraging results, the performances of traditional 3D modeling techniques in terms of accuracy and precision are still superior and must be preferred when the

accuracy requirements are restrictive. But for a very wide and continuously increasing range of applications, when the required accuracy can be at the level from few millimeters (very close-range) to few centimeters, then range cameras can be a valuable alternative, especially when non expert users are involved. Furthermore, the technology on which these sensors are based, driven also by the new generation of AR/VR reality kits (see for example [80]), is continually evolving and certainly also their geometric performances will soon improve.

Finally future work should be directed towards improving the calibrations results and studying the integration with other 3D modeling techniques. Summarizing, further developments could be devoted to:

- refine the precision and accuracy models for the Kinect v1 and the Kinect v2;
- develop similar functional and stochastic models for other low-cost range cameras as soon as they come on the market;
- analyze the effectiveness of the proposed calibration procedure in practical applications, where possible non-optimal reflection conditions may also arise
- apply the calibration models in the 3D modeling software.

Appendix A

Software libraries for 3D modeling with range cameras

In this appendix an overview of the drivers, algorithms and software available for 3D modeling with range cameras is given, with a particular focus on the most used and/or promising proprietary and open source software.

A.1 Drivers and libraries

There are many libraries which allow to gain access to the data streamed by the range cameras. In the following sections the most popular among them will be briefly described.

A.1.1 OpenNI

OpenNI (Open Natural Interaction) was established in November 2010 as an industry-led non-profit consortium and today it is an open source

software project focused on promoting, standardizing and improving the compatibility and interoperability of depth sensing devices [14]. The original aim was to help developers implementing device independent applications based on natural interfaces[1], but, starting from version 2.0, the NUI functionalities (gestures and skeleton detection and tracking, see Fig. A.1) have been eliminated from the SDK core and transferred to an additional and optional layer of middleware.

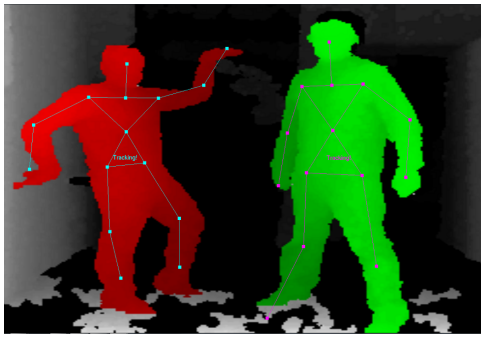


Fig. A.1: The OpenNI skeleton tracking functionality [14].

OpenNI has been the largest 3D sensing development framework and community until it was closed by Apple Inc. when it acquired PrimeSense Ltd., one of the founding member of the original OpenNI consortium[2], on November 24, 2013 [17]. Since then, Occipital and

[1]A natural interface, also called natural user interface (NUI) is a human-machine interface that feels natural to its users, based on an evolving model for human-computer interaction that is context-appropriate and adaptive. Because the NUI exploits the existing skills and expectations of its users, it is easy to learn: the user transition from novice to expert is quickly. A NUI might incorporate speech, gesture, touch, or location, depending upon the application and the user environment.

[2]The original members of the OpenNI consortium were: PrimeSense; Willow-Garage, experts in personal robotics applications; OpenPerception, the makers of the Point Cloud Library (PCL); ASUS; Side-Kick, a leading production house for

other former partners of PrimeSense are still keeping a forked version of OpenNI 2.0 operative as an open source software project, but it is no longer active as it used to be [8].

Anyway OpenNI framework still defines a device-independent C++ API (Fig. A.2 shows the OpenNI 2.0 SDK Architecture) that gives access to the raw data provided by OpenNI compatible depth sensors, providing a uniform interface that third party developers can use to interact with such devices. It supports PrimeSense reference design sensors, the Asus Xtion, the Microsoft Kinect v1 (the Microsoft SDK is also needed) and also other types of range cameras such as Microsoft Kinect v2 and Occipital Structure Sensor.

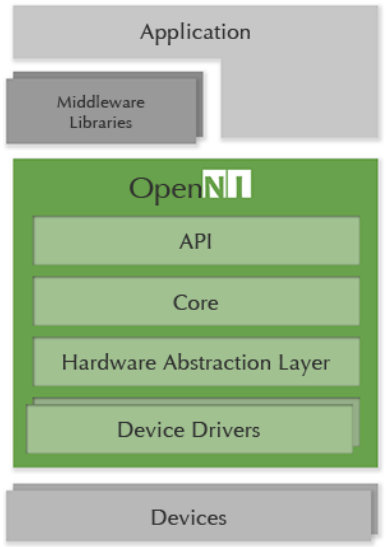


Fig. A.2: The OpenNI 2.0 SDK Architecture [14].

In particular, getting access to the depth streams requires the use of

motion control games.

four main classes that allow developers to initialize the selected sensor and receive depth, RGB, and IR video streams from it [8]:

1. openni::OpenNI class provides a single static entry point to the API. It also provides access to devices, device related events, version and error information;

2. openni::Device class provides an interface to a single device connected to the system, giving access to the streams captured by the sensor;

3. openni::VideoStream class abstracts a single video stream (depth, IR or RGB). Obtained from a specific device, it is required to gain access to the frame data;

4. openni::VideoFrameRef is the basic class used to read each new frame from a video stream. It provides access to the underlying array that contains the frame data, as well as any metadata that are required to work with the video stream.

In addition, various supporting classes and structures are provided for holding specific types of data. A Recorder class can store OpenNI video streams to files, whereas the Listener Classes handle the events that Stream classes generate [8]. Finally video streams can be read using one of two basic methods: loop based (polling) and event based.

A.1.2 OpenKinect: libfreenect1 and libfreenect2

OpenKinect is a community of over 2000 members interested in developing free, open source libraries enabling the two versions of Microsoft

Kinect to be used on platforms other than just the Xbox [7], in order to allow a wider and more general adoption of these devices in different fields of application. Indeed, as is well known, the Microsoft Kinect v1 and the Microsoft Kinect v2 sensors were primarily designed as motion sensing input peripheral, respectively for the Xbox 360 and the Xbox One game consoles. The libfreenect project, developed by the OpenKinect community, was the first open source and cross platform driver available for the Kinect v1, whose USB connection was decoded and reverse-engineered.

Anyway when the OpenNI framework was released, several projects swapped their libfreenect dependencies for the OpenNI ones because they offered more flexibility when replacing the Kinect with other hardware, as well as a more robust set of features to build applications on top of. That said, many developers still adopt the libfreenect drivers since they are easy to redistribute without requiring users to download dependent software [59].

In particular, the libfreenect library includes all code necessary to activate, initialize, and communicate data with the Kinect v1 (libfreenect1) and the Kinect v2 (libfreenect2) sensors [7]. They include drivers and a cross-platform API that runs on Windows, Linux, and OS X systems. The libfreenect1 API support bindings and extensions for C, C++, .NET (C#/VB.NET), Java, Python, and C Synchronous Interface languages/platforms, whereas for libfreenect2 only C++ API is actually available [23]. Both the APIs provide access to the three main sets of data from Kinect sensors in the form of video streams: the depth image, the RGB image and the IR image. In addition to the image based sensor data, Libfreenect also gives access to the audio stream. However, Libfreenect does not supply a skeleton tracking

feature [59].

A.1.3 Microsoft Kinect for Windows SDKs

The Kinect for Windows SDKs (version 1.8 for Kinect v1 and version 2.0 for Kinect v2) supply the tools and APIs, both native (C++) and managed (C#/VB.NET), needed to develop Kinect enabled applications for Microsoft Windows platform with the Microsoft Visual Studio IDE. The Kinect SDKs provide support for the features of the Kinect v1 and Kinect v2, including color, depth and infrared images, audio input, and skeletal data. Both the SDKs consist of a sophisticated software library that allows developers to exploit the rich form of Kinect based natural input, which senses and reacts to real-world events. In addition they also provide the implementation of the KinectFusion algorithm (see Sec. A.2.1 and Sec. A.3.2) that turns the Kinect range cameras into veritable 3D scanners.

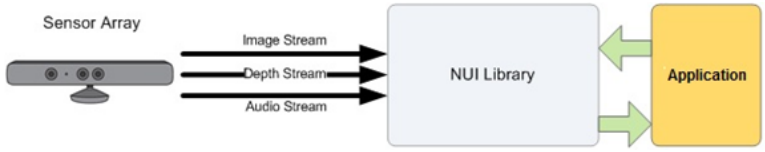


Fig. A.3: Hardware and software interaction with an application [4].

As shown by Fig. A.4, the Kinect for Windows SDK v1.8 includes the following components [4]:

1. Kinect hardware components, including the Kinect v1 device and the USB hub through which the sensor is connected to the computer;

2. Kinect for Windows drivers supporting: microphone array, audio and video streaming controls, device enumeration functions that enable an application to use more than one Kinect v1;

3. audio and video components, i.e. Kinect v1 NUI for skeleton tracking, audio, and color, infrared and depth imaging and accelerometer data;

4. DirectX Media Object (DMO) for microphone array beamforming and audio source localization;

5. Windows standard APIs.

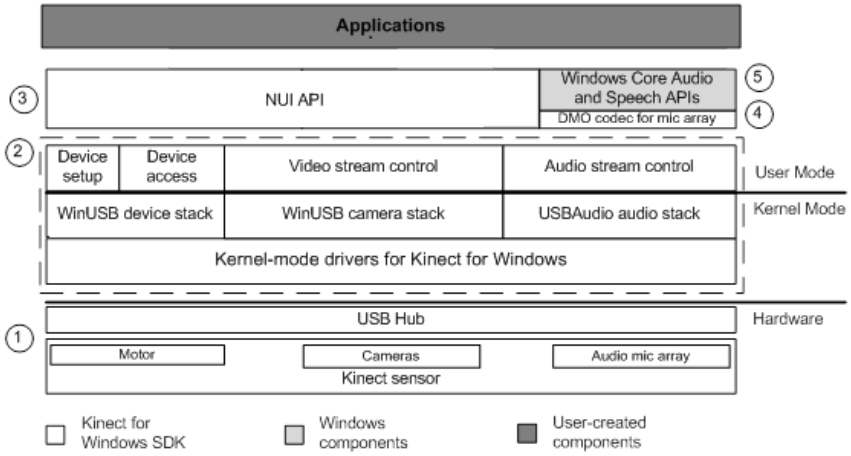


Fig. A.4: Microsoft Kinect for Windows SDK v1.8 architecture [4].

As regards the depth and player data, they can be retrieved in either of two formats: packed depth information and full depth information. The packed depth information is the older format, for which each pixel is represented by one 16-bit value: the 13 high-order bits contain the

depth value and the 3 low-order bits contain the player index. Any depth value outside the reliable range is replaced with a special value to indicate that it was too near, too far, or unknown. Full depth information is the newer format, introduced in version 1.6. In this case each pixel is represented by a structure with two fields: a 16-bit depth and a 16-bit player index. All detected depth values, including those outside the reliable range, are reported. Pixels whose depth could not be detected are reported with a depth value of 0 [4].

Concerning the Kinect for Windows SDK 2.0, it provides three different API sets that can be used to create Kinect v2 enabled applications. A set of Windows Runtime APIs support the development of Windows Store applications. A set of .NET APIs enable the development of WPF applications. And a set of native APIs allow to write applications that require the performance advantages of native code [5]. With these APIs it is possible to retrieve both low-level data, such as infrared and color, as well as processed data, like depth and body (commonly referred to as skeleton) from the Kinect v2 sensor. Each stream has its own reader, but they all share the same basic functionalities.

In particular, the KinectSensor Class is required to configure the Kinect v2 device and access sensor data. Only one device is supported, and the data delivered from the sensor are then stored temporarily in a frame in order to avoid memory allocation. An application should get the data out of each frame and close/dispose it as quickly as possible to free up the underlying handle and make sure that the system does not keep allocating new items to store incoming frame data [5].

Moreover the DepthFrame Class is the central class for 3D modeling applications, since it contains the depth data. They are stored as

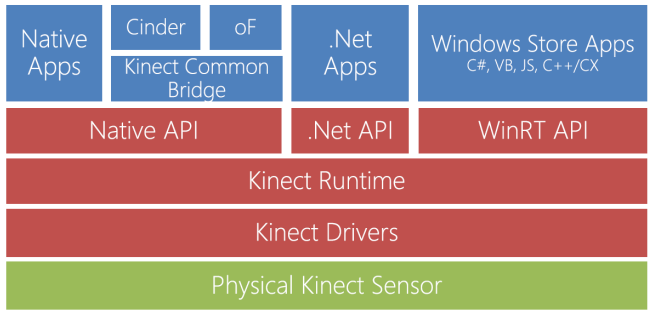


Fig. A.5: Microsoft Kinect for Windows SDK v2.0 architecture [5].

16-bit unsigned integers, where each value represents the distance in millimeters of the closest object observed by that pixel. The maximum depth distance is 8 meters, although reliability starts to degrade at around 4.5 meters [5].

Finally, although the Microsoft Kinect for Windows SDKs can be used exclusively with the Kinect v1 and the Kinect v2 hardware and only on Windows platforms, they are very popular, at least among inexperienced developers, because they are very easy to install and to use.

A.1.4 Occipital Structure SDK

The Structure SDK is the Occipital proprietary framework that defines a stable, easy to use, flexible and constantly improving Objective-C interface for developing applications leveraging the Structure Sensor on iOS devices. It is split in two parts [13]:

1. a low-level Sensor Controller layer, exposing raw depth and color stream access along with sensor information and status;

2. a high-level SLAM Engine, including 3D mapping, tracking and scanning features.

Developing applications for the Structure Sensor requires Xcode 6 or above. The APIs can be called also through Swift programming language and the resulting application can integrate Objective-C++ libraries (for example OpenCV), too. Furthermore the Structure SDK contains source code for useful sample apps such as 3D scanning and indoor mapping [13] (see Sec. A.3.5).

Anyway the Structure SDK is compatible only with iOS devices. Thus, to build applications able to run on not iOS platforms, Occipital maintains OpenNI 2.0 (see Sec. A.1.1), which allows developers to create applications for Structure Sensor on Windows, Linux, macOS, and Android.

A.1.5 Intel RealSense SDK For Windows

The Intel RealSense SDK is a Windows library that implements pattern detection and recognition algorithms, exposed through standardized interfaces. The library aims to help developers to build innovative applications for the next generation of human computer experience.

The 2016 version (R3) of the SDK supports the SR300 and F200 sensors and a few popular languages, frameworks and game engines like C++, C#, Unity and C# Universal Windows Platform (UWP). The SDK allows developers to easily integrate several functionalities in their applications, such as [3]:

- hand tracking, gesture recognition and cursor mode;

- face tracking and recognition, for which the presence of faces in the field of view of the sensor, or facial features on an individual face, can be easily identified. It supports 78 landmark points for 3D face detection as well as face orientation (roll, pitch, and yaw);
- 3D scanning of stationary objects, faces, bodies and heads. The resulting 3D models are stored through standard mesh formats and can be used for inspection, rendering, editing, or printing;
- background removal: it allows to segment the captured scene to remove the background and to create a digital green screen.

A.1.6 Intel RealSense Cross Platform API: librealsense

The librealsense project [1] is a cross-platform C++ library (Linux, Windows, Os X) for capturing data from the Intel RealSense F200, SR300, R200, LR200 and the ZR300 sensors. In particular, it provides support for retrieving native depth, color and infrared streams, synthetic streams for rectified images, calibration information and multi-camera capture. Anyway this library only encompasses camera capture functionality without additional computer vision algorithms, implemented instead in the official Intel RealSense SDK for Windows platforms. This effort was initiated to better support researchers, creative coders, and app developers in fields such as robotics, virtual reality and the internet of things [1].

A.1.7 Point Cloud Library (PCL)

Although it cannot be used for gaining access to the range cameras data, the Point Cloud Library (PCL) is however a powerful tool to process the 3D data collected by these devices.

PCL is indeed a collection of state of the art algorithms and tools for 2D/3D image and point cloud processing. It is a modern, fully templated and open source C++ library, licensed under Berkeley Software Distribution (BSD) terms and, therefore, free for commercial and research use [96]. Furthermore it is cross-platform and it has been successfully compiled and deployed on Linux, MacOS, Windows, and Android [9].

Fig. A.6: The Point Cloud Library logo [9].

The PCL core is structured in smaller libraries (see Fig. A.7), that can be compiled separately. They implement algorithms and tools for specific areas of 3D processing, which can be combined to efficiently solve common problems such as 3D object recognition and 6 Degrees of Freedom (DoF) pose estimation, registration and segmentation of point clouds, surface reconstruction [16]:

- libpcl filters: implements data filters such as downsampling, outlier removal, indices extraction, projections, etc;
- libpcl features: implements many 3D features such as surface normals and curvatures, boundary point estimation, moment invariants, principal curvatures, PFH and FPFH descriptors, spin

images, integral images, NARF descriptors, RIFT, RSD, VFH, SIFT on intensity data, etc;

- libpcl io: implements I/O operations such as writing to/reading from Point Cloud Data (PCD) files;
- libpcl segmentation: implements cluster extraction, model fitting via sample consensus methods for a variety of parametric models (planes, cylinders, spheres, lines, etc), polygonal prism extraction, etc;
- libpcl surface: implements surface reconstruction techniques, meshing, convex hulls, Moving Least Squares, etc;
- libpcl registration: implements point cloud registration methods such as Iterative Closest Point (ICP) [21] algorithm, etc;
- libpcl keypoints: implements different keypoint extraction methods, that can be used as a preprocessing step to decide where to extract feature descriptors;
- libpcl range image: implements support for range images created from point cloud datasets.

Each set of algorithms is defined via base classes that attempt to integrate all the common functionality used throughout the entire pipeline, thus keeping the implementations of the actual algorithms compact and clean. The basic interface for such a processing pipeline in PCL is the following [96]:

- create the processing object (e.g., filter, feature estimator, segmentation);

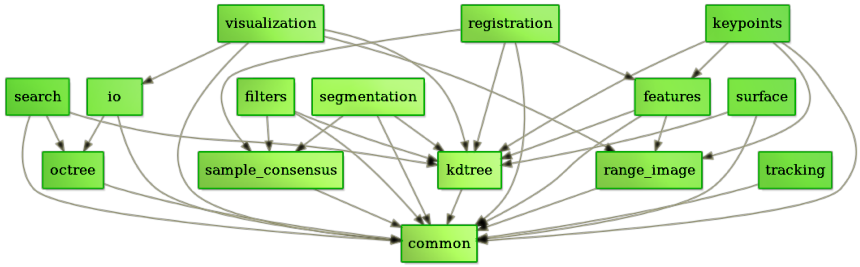


Fig. A.7: PCL architecture [9].

- use setInputCloud to pass the input point cloud dataset to the processing module;
- set some parameters;
- call compute (or filter, segment, etc) to get the output.

Most mathematical operations are implemented with and based on Eigen, an open source template library for linear algebra. In addition, PCL provides support for OpenMP , Intel Threading Building Blocks (TBB) library for multi-core parallelization, CUDA for GPU acceleration and VTK for rendering 3D point cloud and surface data. The backbone for fast k-nearest neighbor search operations is provided by FLANN (Fast Library for Approximate Nearest Neighbors). All the modules and algorithms in PCL pass data around using Boost shared pointers, thus avoiding the need to re-copy data that is already present in the system [96].

The PCL project brings together individuals from all around the world, universities and companies such as, among others, NVIDIA, Google, Toyota, Trimble, Honda Research Institute, Sandia, Dinast, Optronic, Ocular Robotics, Velodyne, Fotonic and Leica Geosystems,

and it has become a reference for anyone interested in 3D processing, computer vision, and robotic perception [16].

A.2 Algorithms for 3D modeling with range cameras

With the advent of the new generation of range cameras, the use of three dimensional data has become increasingly popular. As these sensors are commodity hardware and sold at low-cost, a rapidly growing group of people can acquire 3D data cheaply and in real time. Anyway multiple scans, even hundreds, captured from many different points of view, are usually required to collect information about all sides of the target object. Indeed the obtained individual depth maps have to be brought into a common reference system, a process that is usually called *alignment* or *registration*, so that they can be integrated into a complete and single 3D model, representing the whole surface of the object. This entire process, going from the single depth map acquisition to the overall 3D model generation, is usually known as the *3D scanning pipeline* [20]. Today there are several algorithms that allow to carry out this process in real time. The following sections describe the most popular of them.

A.2.1 KinectFusion

KinectFusion [55, 78] was firstly developed as a research project at the Microsoft Research laboratory in Cambridge, U.K. Today it is included in the official Kinect for Windows SDKs.

KinectFusion is an algorithm for accurate real time, dense volumetric mapping and reconstruction of complex and arbitrary indoor scenes, using only a moving low-cost range camera, originally a handheld Kinect v1 device. All of the depth data streamed from the sensor are fused into a single global surface representation of the observed scene [78]. Note that to generate a complete 3D model, different viewpoints of the physical scene must be captured [55]. Even small motions, caused for example by camera shake, result in new viewpoints of the scene and hence refinements to the model. This creates an effect similar to image super resolution [55].

KinectFusion is the result of the evolution of both algorithms for estimating camera pose and extracting geometry from images and camera technologies. In fact, newer range cameras based either on ToF or SL sensing offer dense measurements of depth in an integrated device. Moreover such technology has now reached consumer level accessibility [78].

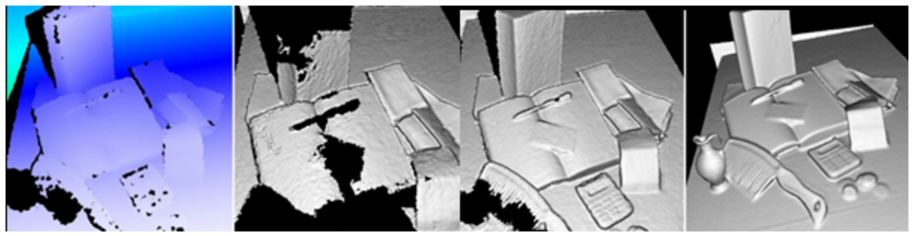


Fig. A.8: KinectFusion in action, taking the depth image from the range camera (here a Kinect v1) with lots of missing data and within a few seconds, producing a realistic smooth 3D reconstruction of a static scene by moving the sensor around [6].

In particular, users can simply pick up and move a range camera to generate a continuously updating, smooth, fully fused 3D surface

reconstruction. This can be accomplished either by moving the sensor around the object or environment or by moving the object being scanned in front of the sensor. Using only depth data, the KinectFusion algorithm continually tracks the six DoF pose of the camera and merges live data from the camera into a single global 3D model in real time. As the user explores the space, new views of the physical scene are revealed and these are fused into the same model. The reconstruction therefore grows in detail as new depth measurements are added. Holes are filled (see Fig. A.8), and the model becomes more complete and refined over time [55]. A strong point by using only depth data is that the proposed system can work in complete darkness, thus mitigating any issues concerning low light conditions, problematic for passive camera and RGB-D based systems [78]. Real-time camera tracking and surface reconstruction by Kinect Fusion is based on the following processing steps [78]:

- surface measurement: a pre-processing stage, where a dense vertex map and normal map (it supplies the surface normals, providing the orientation of the scanned surface) pyramid are generated from the raw depth measurements obtained from the range camera [78];

- surface reconstruction update: the global scene fusion process, where given the pose determined by tracking the depth data from a new sensor frame, the surface measurement is integrated into the scene model maintained with a volumetric, Truncated Signed Distance Function (TSDF) representation [78];

- surface prediction: unlike frame to frame pose estimation, the

loop between mapping and localisation is closed by tracking the live depth frame against the globally fused model. This is performed by raycasting the signed distance function into the estimated frame to provide a dense surface prediction against which the live depth map is aligned [78];

- sensor pose estimation: live sensor tracking is achieved using a multi-scale ICP alignment between the predicted surface and current sensor measurement. The GPU based implementation uses all the available data at frame-rate [78].

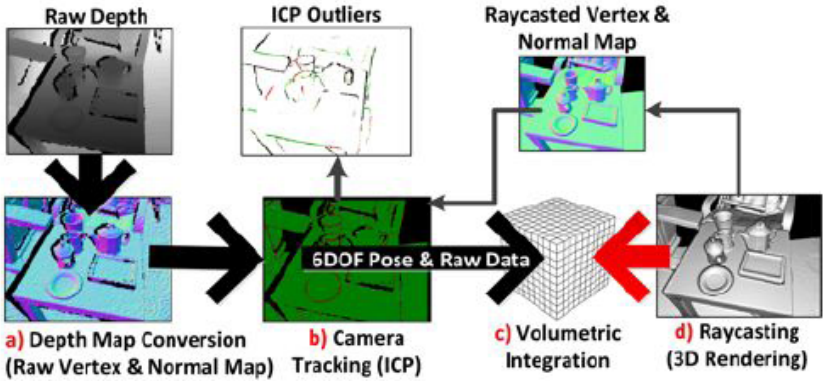


Fig. A.9: Overview of tracking and reconstruction pipeline from raw depth map to rendered view of 3D scene [55].

The main system 3D scanning pipeline consists therefore of four main stages (labelled appropriately in Fig. A.9) [55]:

- *Depth Map Conversion.* The live depth map is converted from image coordinates into 3D points – referred to as vertices of the point cloud – in the coordinate space of the sensor. Although the

quality of this depth map is generally remarkable given the cost of the device, it contains numerous holes and depth measurements often fluctuate and are inherently noisy [55].

- *Camera Tracking.* In this step, a rigid 6 DoF transform is computed to closely align the current oriented points with the previous frame, using a GPU implementation of the ICP [21] algorithm. Relative transforms are incrementally applied to a single transform that defines the global pose of the sensor [55]. In other words, this second stage calculates the global/world camera pose (its location and orientation) and tracks this pose as the sensor moves in each frame using an iterative alignment algorithm, so the system always knows the current sensor pose relative to the initial starting frame [6].

- *Volumetric Integration.* It is performed using a volumetric surface representation of the space around the sensor, instead of fusing point clouds or creating a mesh. Given the global pose of the camera, oriented points are converted into global coordinates, and a single 3D voxel grid is updated. Each voxel stores a running average (to reduce noise, yet handle some dynamic change in the scene) of its distance to the assumed position of the physical surface being scanned [55].

- *Raycasting.* Finally, the volume is raycasted to extract views of the implicit surface, for rendering to the user. When using the global pose of the camera, this raycasted view of the volume also equates to a synthetic depth map, which can be used as a less noisy and more globally consistent reference frame for the next

iteration of ICP. This allows tracking by aligning the current live depth map with the less noisy raycasted view of the model, as opposed to using only the live depth maps frame-to-frame [55].

The KinectFusion alghorithm has an open source implementation, PCL KinFu, realized by the developers of the Point Cloud Library. It will briefly be described later (see Sec. A.3.3).

Finally it is worth noting that, although the KinectFusion algorithm was originally developed for the Kinect v1 device, it is applicable to all range cameras, and, in general, to all sensors able to generate good quality real time depth maps.

A.2.2 Kintinuous: Spatially Extended KinectFusion

Kintinuous, or Spatially Extendend KinectFusion, was developed as a research project by the Department of Computer Science of National University of Ireland Maynooth in conjunction with the Computer Science, the Artificial Intelligence Laboratory (CSAIL), the Massachusetts Institute of Technology (MIT) and Cambridge. Kintinuous [105] is an extension to the original KinectFusion algorithm that allows *dense* mesh based mapping of *extended* scale environments in real time, by virtually translating the volumetric model as the sensor moves. This is achieved through:

i. altering the original version such that the region of space being mapped by the KinectFusion algorithm can vary dynamically [105];

ii. extracting a dense point cloud from the regions that leave the KinectFusion volume due to this variation [105];

iii. incrementally adding the resulting points to a triangular mesh representation of the environment [105].

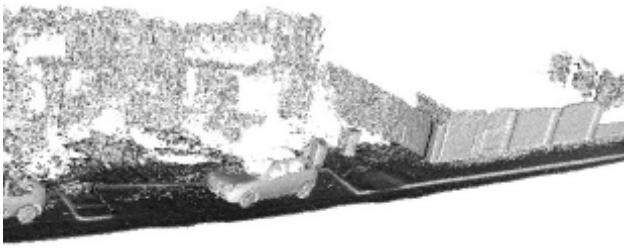


Fig. A.10: Kintinuous map reconstruction of an outdoor dataset captured from a moving car [105].

The system is implemented as a set of hierarchical multi-threaded components which are capable of operating in real-time. The architecture facilitates the creation and integration of new modules with minimal impact on the performance of the dense volume tracking and surface reconstruction modules. Experimental results demonstrate the ability of the system to map areas considerably beyond the scale of the original KinectFusion algorithm, including a two story apartment and an extended sequence taken from a car at night (see Fig. A.10) [105]. What is more, Kintinuous has an open source version implemented by the PCL programmers and its name is KinFu Large Scale (see A.3.3).

Finally, a second version of Kintinuous [106] algorithm was also developed. Being the first version an open loop process, it inevitably suffers from unbounded drift. The second version presents a method

for dealing with this problem which takes advantage of camera pose graph optimisation and non-rigid space deformation for map correction during loop closures. The result is a visual SLAM system which captures high fidelity dense maps in real time with the local reconstruction quality of KinectFusion, and also the advantages of global consistency given by camera pose graph optimisation [106].

A.2.3 OmniKinect

OmniKinect [57], developed by the Institute for Computer Graphics and Vision of the Graz University of Technology, is an extension to the KinectFusion algorithm as well. It allows to produce high quality volumetric reconstructions from multiple Kinect v1 devices and in real time, whilst overcoming systematic errors in the depth measurements.

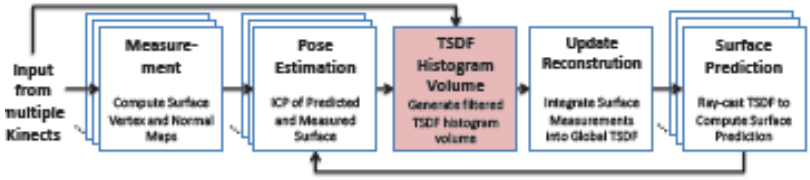


Fig. A.11: Overall system work flow for the modified KinectFusion algorithm to support multiple simultaneous Kinects with different inaccuracies. The additional step is marked as red center square [57].

To work properly with simultaneous uncorrected input streams from multiple Kinect v1 devices, an additional step is added to the original KinectFusion algorithm (see Fig. A.11). In particular OmniKinect uses a smoothed histogram volume of truncated signed distance functions to filter outlier measurements of the signed distance field before a temporal smoothing. This approach uses only an initial

extrinsic pose estimation of the cameras. In this way, persistent outliers due to variations in the registered pose or depth accuracies are removed, yielding a more robust estimate of the surface generating a complete and accurate reconstruction of the observed volume.

The OmniKinect system also combines element from the *Shake'n' Sense* technique. It is a simple method that mitigates the interference caused when multiple structured light depth cameras point at the same part of a scene. Simultaneous multiple depth cameras can extend the coverage of the single device, overcome occlusions and create complete 360° 3D representations of environments and objects contained. However, the depth signal severely degrades when multiple cameras are pointing the same scene. In fact, there is a *crosstalk* when dot patterns of devices interfere with one another [25]. Shake'n'Sense consists only in a mechanical augmentation, thus being non-destructive and does not impact depth values or geometry. The key behind this technique is to minimally vibrate a Kinect camera unit using an offset-weight vibration motor and thereby artificially introduce motion blur. Both the structured light diffractive optical element emitter and the IR camera of structured light sensor will move together, which means that depth sensor works as normal, albeit with a little induced blur [25]. The qualitative results shows that by vibrating each structured light device independently, interference is dramatically reduced and the number of holes improved.

A.3 Publicly available 3D modeling tools for range cameras

Today, many tools are available to digitize a scene/object in 3D with range cameras. Through these tools, this kind of sensors can be easily used as ordinary video cameras, simply moving them around the object to be captured, but, instead of recording a video, they reconstruct a complete 3D model of the scene in real time. In addition, it is worth noting that, thanks to the range cameras, the entire scanning process is performed in metric space. After a brief overview of the common features of the 3D modeling tools, the following sections describe the most important 3D scanning tools for range cameras, whose main characteristics are resumed in Tab. A.1.

A.3.1 General features of 3D modeling tools

Most of the 3D modeling tools for range cameras share generally several features. First of all, the object/scene must be captured from different points of view in order to reconstruct a complete 3D model. Therefore the user must slowly move around the target object, following a 360° path and not forgetting to scan all of its sides (including the top and the bottom). In most cases the needed scanning time is about a few minutes, depending on the object shape and complexity.

Furthermore, virtually all the 3D scanning tools for range cameras implement the Augmented Reality (AR) functionality for which the preview of the 3D model appears in real time on top of the object on the screen of the computing device to which they are connected.

Therefore it is possible to coordinate the movements of the sensor with the live view and check the model quality immediately, during the very same scan. In this way the operator can improve the results instantly, by scanning again the problematic areas and filling the model holes.

Moreover, most of the 3D scanning tools, especially the commercial ones, permit to move the scanning volume over the object to be modelled. Usually this function is carried out automatically, leveraging specific features of the scene surface. Anyway the operator should always check the dimensions of the scanning volume. Indeed, the scanning volume can be usually restricted or enlarged in relation to the target object size. In order to obtain a most accurate 3D model, it is strongly recommended not to waste resolution and thus the scanning volume should be just a little bit wider than the object within it.

Finally it is relevant to notice that the 3D scanning tools can lose the tracking of the object. The tracking loss can happen, for example, when the user moves too fast or when the object to be scanned is too small. This case is handled differently depending on the 3D scanning application used. For example, with some tools, the operator should try to overlap again the model to the object by re–scanning an already captured part or restart the scan. However it is also important to consider the possibility that the object could not be suitable for scanning with range cameras (for example if it has too tiny particulars, or a dark/shiny surface).

A.3.2 Microsoft for Windows SDKs KinectFusion Samples

The Microsoft Kinect for Windows SDKs provide the implementation of KinectFusion algorithm in several samples, both for the Kinect v1 and Kinect v2 devices. The basic samples demonstrate the fastest way to get started and minimum code required for KinectFusion operation, whereas the Explorer samples expose many of the API parameters as editable controls in the graphical user interface of the application, allowing more exploration of the KinectFusion capabilities [6].

The source code is an integral part of the SDK and it can be built with Visual Studio. Anyway some of the inner functions are not accessible to the user since they are placed in closed dlls. The Microsoft implementation can process data either on a DirectX 11 compatible GPU with C++ AMP, or on the CPU, by setting the reconstruction processor type during reconstruction volume creation. The CPU processor is best suited for offline processing as only modern DirectX 11 GPUs will enable real time and interactive frame rates during reconstruction. Typical volume sizes that can be scanned are up to around 8 m^3. Typical real world voxel resolutions can be up to around $1-2$ mm per voxel. However, it is not possible to obtain both of these simultaneously [6].

In particular the KinectFusionExplorer sample provides more configurability over the algorithm parameters. For instance, the user can specify the size of the desired scanning volume. The number of voxels that can be created depends on the amount of memory available to be allocated on the used computer, and typically up to around $640 \times 640 \times 640 = 262144000$ voxels can be created in total on devices

with 1.5GB of memory or more [6].

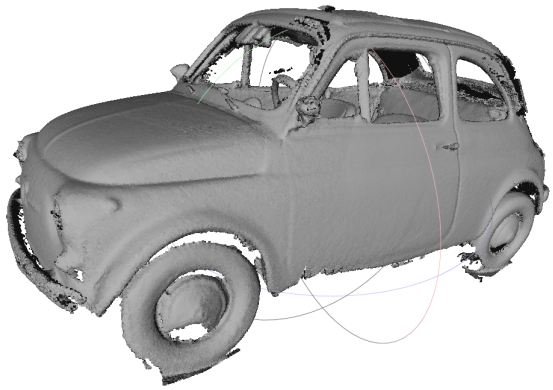


Fig. A.12: A FIAT 500 car model collected with the KinectFusion sample of the Microsoft for Windows SDK v1.8.

The aspect ratio of this volume can be arbitrary; however, the user should try to match the volume voxel dimensions to the shape of the area in the real world. The voxelsPerMeter member scales the size that 1 voxel represents in the real world, so a cubic $384 \times 384 \times 384$ volume can either represent a 3 m cube in the real world if the voxelsPerMeter parameter is set to 128 vpm (as $384/128 = 3$, where each voxel is $3m/384 = 7.8mm^3$), or a 1.5 m cube if it is set to 256 vpm ($384/256 = 1.5$, where each voxel is $1.5m/384 = 3.9\ mm^3$). The combination of voxels in the x, y, z axis and voxels per meter enables to specify a volume with different sizes and resolutions, but it is worth to notice that, with a fixed number of voxels, there is a tradeoff between the resolution and the size of the volume [6].

A.3.3 PCL KinFu and KinFu Large Scale, KinFu remake

As told before, KinFu and KinFu Large Scale are respectively the open source PCL implementation of the Microsoft KinectFusion and Kintinuous algorithms. Anyway they are difficult to install, at least for not expert users. Indeed they require Boost (multithread), eigen3 (Matrix operation), FLANN (classification), VTK (3D visualization), OpenNI (range cameras data I/O), CUDA (GPU acceleration) and obviously PCL.

Fig. A.13: A room model captured with the PCL KinFu Large Scale software [9].

KinFu, originally shared in PCL in 2011, was the first open implementation of the KinectFusion algorithm. Today KinFu code is old, deprecated and probably it will be soon removed from PCL library. However a KinFu remake has been already developed: it is a lightweight, reworked and optimized version of the original KinFu [18]. This new version still requires OpenNI, CUDA, OpenCV and VTK libraries, but it is now independent from OpenCV GPU module and

PCL library, making the building process easier, and the code size is reduced drastically, with a great readability improvement. Furthermore the performance has been enhanced by 1.6x factor (Fermi-tested) and the algorithm parameters are no longer hardcoded: all of them can be changed at runtime (volume, size, ...) [18].

With respect to the Microsoft implementation, KinFu has the advantage that it can be used with all the devices compatible with OpenNI (see Sec. A.1.1) and not only with Microsoft Kinect sensors. Furthermore every part of the code is open and can be customized by the developers. Anyway KinFu, depending on CUDA, runs only on computer with NVIDIA graphic cards, while the Microsoft implementation can run on all DirectX 11 compatible GPUs, and it is easier to install.

A.3.4 Intel RealSense SDK Scan3D sample

The Scan3D sample is a C# application that shows the 3D scanning capabilities of the Intel RealSense F200 and SR300 sensors. The sample is part of the 3D scan module of the Intel RealSense SDK for Windows.

It adopts an object detection method to automatically set the size and shape of the scanning volume around the target object, that must be placed on a flat surface, like a table. Once the scan starts, the system automatically removes the flat surface from the accumulated data and the resulting mesh. During the scan, the user needs to either turn the object in front of the sensor, or to move the sensor around the object so that the algorithm can scan the object from all angles. Anyway the first is the better scanning modality for the F200 and SR300 sensors, being them user facing cameras. The sample also warns the user when the object is too far/close to the sensor [2].

The mesh data output formats are OBJ, STL or PLY. If texture mapping option is enabled, the final 3D model is stored in the OBJ format preserving the color features of the target object. Otherwise color information is stored through per vertex format (PLY), that supplies low quality color rendering. Furthermore the user can also enable the solidification option to generate a closed mesh (e.g. for printing or simulation): it extends the color and surface curvature to close holes in areas that were not visible to the sensor during the scanning process [2].

A.3.5 Occipital applications

Occipital provides three different iOS scanning apps for its Structure Sensor, the first 3D sensor for mobile devices. Two of them, the Scanner and Room Capture apps, are an integral part of the Structure SDK: their source code is available in the form of samples and can be customized by the developers. The last one, the Canvas app, can be downloaded from the App Store and it is a closed source commercial application.

Since the Structure Sensor does not have its own colour camera, all of the 3D scanning apps developed for it, not only those provided by Occipital, exploit the iOS device colour camera at which the sensor is connected in order to retrieve also the color information of the target object/scene. Therefore, considering that the object geometry and the texture are captured from two different points of view, it is necessary to calibrate the precise alignment (reconstruct the geometric relationship) between the Structure Sensor infrared camera and the iOS device camera in order to accurately overlap the geometry and colour data in

the final 3D model.

The Calibrator app provided by Occipital can achieve this goal. Specifically designed for the Structure Sensor bracket accessory, it is the unique calibration app actually available on the Apple Store.

A.3.5.1 Scanner and Room Capture

The Scanner application is a powerful mobile 3D scanner, covering a majority of the Structure framework functionalities (high and low level) [13]. Simple and easy to use, it allows to capture 3D models of objects and people by simply walking around them with an iOS device (preferably a modern iPad) connected to the Structure Sensor. The Scanner app automatically places the scanning volume, visualized on the iOS device screen by exploiting the AR potentialities of the Structure Sensor, over flat surfaces such as floors or walls and allows the user to adjust its size in order to fit over the target object dimensions using the typical "pinch" gesture of the iOS devices.

Concerning the Room Capture application, it demonstrates the acquisition of larger, textured environments [13]. Also in this case the size of the scanning volume can be adjusted in relation to the real dimensions of the room/environment, this time by simply moving the position of a slider placed in the lower right corner of the graphical interface (the values are already in meters). The model is built in real time, and it appears in the form of AR on the iOS device screen. To achieve the best quality scan, the user should try to turn slowly in a circle, not moving around too much. Once the acquisition is completed, the resulting mesh can be measured in metric scale.

For both the applications, the final 3D models can be explored on

the iOS device screen with natural touch gestures and exported for free via email in the OBJ format. The calibration is fundamental to improve the tracking process and the texture quality. The user can choose to use either the *Old Tracker*, that only leverages geometry information, or the *New Tracker* that also exploits color data to keep track of the object being scanned. Obviously it is hard or impossible to model uniform objects with few geometric details using the Old tracker.

A.3.5.2 Canvas

Released on November 2016, Canvas is a new iPad application for the Structure Sensor that allows to instantly capture a scale accurate 3D model of an apartment, one room at a time. Thus it is very similar to the Room Capture application, but if the sample is exclusively meant to show the Structure Sensor potentialities, Canvas is a commercial, closed source application.

The app guides the user through the scan, overlaying already scanned zones with a paint like filter that shows the missing areas. For the best quality scans Occipital recommends its optional 20$ wide angle attachment for the iPad camera. Once the scan is completed, a raw 3D model of the scene appears on the iPad screen and the distances between the objects in the apartment can be virtually measured on the final 3D model, that can be inspected from any angle and revisited at any time [74].

The most innovative feature of the Canvas app is the Scan To CAD service that semi-automatically converts the raw 3D scans into clean CAD files for a 29$ fee per scan. The CAD files take up to 48 hours

to create and get emailed back to the user account [74]. Anyway it is always possible to export the raw 3D model for free by connecting the iPad to a Mac computer and downloading it from the Canvas app document directory.

A.3.6 itSeez3D

Itseez3D was developed by Itseez Inc., a Russian company leader in implementing Computer Vision (CV) algorithms for embedded and specialized hardware and the main developer of the renowned open source computer vision library OpenCV. Itseez Inc. was acquired by Intel on 26 May 2016.

Designed to work with the Occipital Structure Sensor or the Intel RealSense R200 through an iPad or a Windows tablet respectively, the itseez3D application allows to reconstruct a 3D model of a person, object or environment simply by walking around it.

For object scanning, the tablet must be held in landscape mode and the object must be placed on a flat surface, like a table or a floor. The tracking process uses both shape and color information, so it is a good practice to put the target object on a surface with many color or contrast details. The AR functionality guides users to capture the highest quality scan possible drawing the virtual preview mesh on the object and an 360° indicator shows the overall progress. The flat surface is automatically detected and it does not appear in the preview model [68].

As regards the environment scanning, it is still an experimental feature, introduced by version 4.0, and designed to model small areas. The user experience is similar to object scanning [68]. The dimensions

of the volume can be selected by scaling the scanning cube with pinch gesture.

Once the scanning is completed, itseez3D delivers the data to the cloud and returns with a complete 3D model within minutes. The cloud processing permits to retrieve high resolution color and high quality geometry data, generally better than those obtained with the Occipital Scanner app for the Structure Sensor. Anyway, to export the models via mail in .ply or .obj formats, it is necessary to pay 7 $ per model (free Individual subscription) [68].

A.3.7 SCANDY

Founded in 2014, Scandy is an American start-up focused on the use of the 3D scanning technology for simplifying the 3D printing process of high fidelity color models of objects and/or landscape. Their Scandy mobile app allows users to model objects (Scandy objects mode) or landscapes in the form of panoramas (Scandy spheres mode) and then have them 3D printed by the Scandy 3D printing service.

In particular, Scandy (objects mode) is a 3D scanning application specifically designed for iPad that leverages the Structure Sensor capabilities to generate a 3D model of a person or object in real time and in a user-friendly way. The Structure Sensor retrieves the depth data while the iPad camera is used to overlay the colour data. Together they generate the full 3D scan. The AR bounding box controls permit to easily resize the scanning volume. The obtained 3D model can be then uploaded to the Scandy cloud that will make it watertight and 3D printable [11].

Furthermore, in August 2016 Scandy launched a beta program for

its own 500$ 3D scanning sensor, specifically designed to work with Android mobile devices. The company is using the 3D technology from PMD in order to obtain high precision 3D models.

A.3.8 SKANECT

Skanect was developed by ManCTL, a French-American company founded in late 2011 by Nicolas Tisserand and Nicolas Burrus, acquired by Occipital in 2013. Since then, Skanect Pro has been redesigned by the Occipital team to work seamlessly with the Structure Sensor. In addition, it supports also Kinect v1 device, Asus Xtion Pro Live and PrimeSense Carmine 1.08 and 1.09. Skanect is an easy to use software tool that allows users to capture a full color 3D model of an object, a person or a room in real time. It actually runs on Windows PCs and Mac [12].

Fig. A.14: A chair model created with the Skanect software [12].

The software can use either the GPU than the CPU reconstruction. The GPU reconstruction requires a top-end NVidia graphics card

with CUDA support. Thanks to the massive amount of computational power of these devices, a precise and smooth fusion can be performed in real time. The main limitation of this technique, a part from the GPU requirement, is its sensitivity to the *geometry* of the scene. So the user should avoid flat walls and other scenes with little geometry. On the other hand, the CPU reconstruction does not require a powerful graphics card, but usually gives lower quality results. It has a number of advantages though. It can work with little geometry, as long as there is enough *texture* information in the color image. For example, an homogeneous wall will not work, but a wall with paintings will. It also does not require to predefine the volume of the scene, and it is thus suitable for open spaces. For commercial use the PRO version has to be purchased, while the free version is available for a personal and hobbyist use.

A.3.9 ReconstructMe

ReconstructMe is an easy to use real time 3D scanning system. The scanning results are 3D models of everyday objects that can be accessed in memory or exported to various CAD format like STL, OBJ, 3DS and PLY. The software supports a growing selection of modern sensors such as Microsoft Kinect v1, Asus Xtion (Pro and Pro Live), PrimeSense Carmine, Intel RealSense (F200, R200) and Occipital Structure Sensor. As long as the used sensor provides the necessary color stream, the texture of the object being scanned can be processed and merged to the final 3D model [10].

ReconstructMe is split into an SDK and applications built upon this library (see Fig. A.16). The SDK supplies methods and types

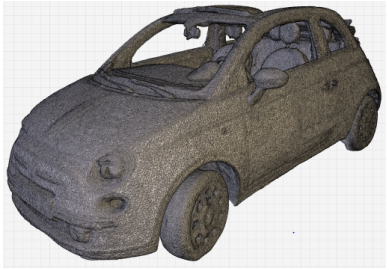


Fig. A.15: A FIAT 500 car model captured with the ReconstructMe software [10].

to control the real time 3D reconstruction process. It successfully abstracts the complexity of communicating with sensors and automatically utilizes high performance devices such as GPUs or CPUs for the reconstruction process.

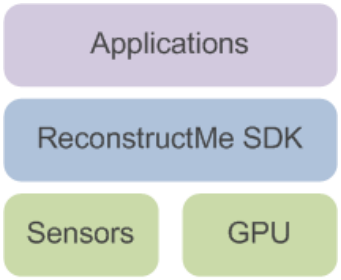


Fig. A.16: The ReconstructMe architecture [10].

In particular, the SDK offers a pure C-based API without additional compile time dependencies. The API is designed to provide a maximum performance for a smooth reconstruction experience. ReconstructMe currently runs on Windows 32 bit and 64 bit operating systems and it is available for free for non-commercial projects, but, for this last case, a commercial license must be purchased (179 €) [10].

3D scanning software	Platforms	Depends on	Sensors
PCL KinFu and KinFu Large Scale	Windows Mac Os Linux	PCL, Boost, eigen3, FLANN,VTK OpenNI, CUDA	OpenNI compliant sensors
Kinfu remake	Windows Linux	CUDA ($\geqslant$5), OpenCV 2.4.9 with Viz module enabled, OpenNI v1.5.4	OpenNI compliant sensors
ReconstructMe	Windows	information not available	Intel RealSense (R200 and F200), PrimeSense (1.08 and 1.09), Microsoft Kinect v1, Occipital Structure Sensor, Orbbec Astra (S and L), Asus Xtion (Pro and Pro Live)
Skanect	Windows Mac OS X	information not available	Asus Xtion, Occipital Structure Sensor, PrimeSense Carmine, Microsoft Kinect v1
Microsoft for Windows SDKs KinectFusion Samples	Windows	Kinect for Windows SDK v1.8 (Kinect v1) or v2 (Kinect v2)	Kinect v1 Kinect v2
Intel RealSense SDK Scan3D sample	Windows	Intel RealSense SDK for Windows 2016 R3	Intel RealSense F200 and SR300
Occipital Structure SDK samples	iOS	Structure SDK	Occipital Structure Sensor

Tab. A.1: Available 3D scanning software for range cameras.

3D scanning software	Remarks
PCL KinFu and KinFu Large Scale	BSD license, source code completely available, Tracking based exclusively on 3D geometry. difficult to build for not expert users.
Kinfu remake	BSD3 license, source code completly available, less difficult to build than the original PCL version anyway still complex to build for not expert users. Tracking based exclusively on 3D geometry.
ReconstructMe	Commercial closed source, free for non commercial use
Skanect	Two tracking possibilities: one based exclusively on geometry, one based on both geometry and color data
Microsoft for Windows SDKs KinectFusion Samples	Source code partially available, inner functions are placed in closed dlls. Tracking based exclusively on 3D geometry
Intel RealSense SDK Scan3D sample	Source code partially available, inner functions are placed in closed dlls
Occipital Structure SDK samples	Source code available. Two tracking possibilities: one based exclusively on 3D geometry, one based on both geometry and color data

3D scanning software	Platforms	Depends on	Sensors
Canvas	iOS	Probably Structure SDK	Occipital Structure Sensor
Itseez3D	iOS Windows (tablet)	information not available	Occipital Structure Sensor, Intel RealSense R200
Scandy	iOS	Scandy Core	Occipital Structure Sensor

3D scanning software	Remarks
Canvas	Commercial closed source for 3D scanning of large environments (home)
Itseez3D	Commercial closed source, unlimited scanning for individual subscription (0 $/month), but 7 $ per model export; considered the best by the 3D scanning community in its price range
Scandy	Commercial closed source for 3D printing purpose

Appendix B

Developed software

The code development was an important activity of the PhD research, fundamental in order to leverage all the capabilities that range cameras offer, as described in appendix A. In this appendix the implemented applications are shortly illustrated, after a brief overview of the IT facilities used. Finally a brief description of the software implemented by the author during the two Google Summer of Code programs is also given.

B.1 IT Equipment

In the following subsections the platforms, the networks and the programming languages used to code the developed applications are introduced.

B.1.1 .NET Platform and Framework

The .NET Platform is, in essence, a new development framework that provides a fresh application programming interface (API) to the services and APIs of classic Windows operating systems, while bringing together a number of disparate technologies that emerged from Microsoft during the late 1990s. The platform consists of four separate product groups [49]:

- a set of languages, including C# and VB .NET; a set of development tools, including Visual Studio .NET; a comprehensive class library for building web services and web and Windows applications; as well as the *Common Language Runtime* (CLR) to execute objects built within this framework;
- a set of .NET Enterprise Servers;
- an offering of commercial web services, called .NET My Services;
- new .NET-enabled non-PC devices.

Microsoft .NET supports not only language independence, but also language integration that means you can take advantage of polymorphism across different languages. The .NET Framework makes this possible with a specification called the *Common Type System* (CTS) that all .NET components must obey. Additionally, .NET includes a *Common Language Specification* (CLS), which provides a series of basic rules that are required for language integration. The CLS determines the minimum requirements for being a .NET language. Compilers that conform to the CLS create objects that can interoperate with one another [49].

B.1.2 C sharp (C#)

C sharp (C#) is a relatively new programming language, announced by Microsoft in July 2000. The goal of C# is to provide a simple, safe, modern, object-oriented, Internet-centric, high-performance language for .NET development. Although C# is a new language, it draws on the lessons learned over the past three decades. In fact, it is easy to see in C# the influence of Java, C++, Visual Basic (VB), and other languages. C# is learned specifically to create .NET applications and therefore this language is firmly placed in the context of Microsoft's .NET platform and in the development of desktop and Internet applications [49]. The C# language has only about 80 keywords and a dozen built-in data types, but C# is highly expressive when it comes to implementing modern programming concepts.

At the heart of any object-oriented language is its support for defining and working with classes. Classes define new types[1], allowing you to extend the language to better model the problem you are trying to solve. C# contains keywords for declaring new classes and their methods and properties, and for implementing encapsulation, inheritance, and polymorphism, the three pillars of object-oriented programming. It provides component-oriented features, such as properties, events, and declarative constructs (called *attributes*).

B.1.3 Windows Presentation Foundation

The Windows Presentation Foundation (WPF) is a graphical display system for Windows, and it means WPF applications cannot run on

[1] A *type* represents a thing. In C# a type is defines by a *class*, while the individual instances of that class are known as *objects*.

other operating systems. What is more, it is available only for C# language. WPF is designed for .NET, influenced by modern display technologies such as HTML and Flash, and hardware accelerated. It is also the most radical change to hit Windows user interfaces since Windows 95 [71].

In WPF, the underlying graphics technology is not GDI/GDI+, but it is DirectX. Remarkably, WPF applications use – and work through – DirectX no matter what type of the selected user interface. In simple words, rich effects such as transparency and anti-aliasing can be used even in the most mundane application. What is more, the user interface can also benefit from hardware acceleration, which simply means DirectX hands off as much work as possible to the graphics processing unit (GPU), which is the dedicated processor on the video card.

The goal of WPF is to off-load as much work as possible on the video card, so that complex graphics routines are render-bound (limited by GPU) rather than processor-bound (limited by CPU). The CPU is therefore kept free for other work and the best use of the video card and their performance increases is accomplished.

If the only thing WPF offered was hardware acceleration through DirectX, it would be a compelling improvement but not a revolutionary one. But WPF actually includes a basket of high-level services designed for application programmers. The most interesting to be mentioned here is the declarative user interface. Although you can construct a WPF window with code, Visual Studio takes a different approach. It serializes the content of each window to a set of XML tags in a XAML[1] document. In other words, XAML documents define the

[1]XAML is short for Extensible Application Markup Language

arrangement of panels, buttons and controls that make up windows in a WPF application. The advantage is that user interface is completely separated from code, and graphic designers can use professional tools to edit XAML files and refine the front end of the application.

B.1.4 Emgu CV and OpenCV

Emgu CV is a cross platform .NET wrapper to the OpenCV image processing library and it allows OpenCV functions to be called from .NET compatible languages such as C# , VB, VC++, IronPython etc.

B.1.5 Meta.Numerics

Meta.Numerics is a library for advanced scientific computation in the .NET Framework. It can be used from C# , Visual Basic, F♯, or any other .NET programming language. The Meta.Numerics library is fully object-oriented and optimized for speed of implementation and execution.

It is a math and statistics library that offers an API for matrix algebra, advanced functions of real and complex numbers, signal processing and data analysis

B.1.6 Objective C

$Objective - C$ is a general-purpose, object-oriented programming language that adds Smalltalk-style messaging to the C programming language. It is the main programming language used by Apple for the OS X and iOS operating systems, and their respective application programming interfaces (APIs), Cocoa and Cocoa Touch [33].

The programming language $Objective-C$ was originally developed in the early 1980s. It was selected as the main language used by NeXT for its NeXTSTEP operating system, from which OS X and iOS are derived. Portable Objective-C programs that do not use the Cocoa or Cocoa Touch libraries, or those using parts that may be ported or reimplemented for other systems, can also be compiled for any system supported by GCC or Clang. Objective-C source code implementation program files usually have *.m* filename extensions, while Objective-C header-interface files have *.h* extensions, the same as C header files. Objective-C++ files are denoted with a *.mm* file extension [33].

B.2 Kinect Measurement Tool

The *Kinect Measurement Tool* was developed in order to evaluate the precision and accuracy of the Kinect v1 and Kinect v2 sensors. Based on the APIs of the Kinect for Windows SDK (v1.8 e v2.0), it is a WPF application developed in C# with the Microsoft Visual Studio 2013 IDE. It exploits the functionalities of parallel computing (background worker) available in the C# libraries.

The Kinect Measurement Tool is based on a Graphical User Interface (GUI), which allows to:

- display in real time on the PC screen the depth, RGB or/and IR data captured by the Kinect (v1 or v2), respectively as a gray depth map and classical color images (see Fig. B.1);
- manually select some interest points or complete portions of the depth map;

- automatically collimate the corners of one or more checkerboard grids (thanks to the EmGU capabilities), both static and dynamic (on the color or the infrared image);
- compute and collect 3D coordinates (X, Y, Z) of the selected points in the local reference system of the Kinect (v1 or v2);
- automatically measure the distances between the selected points;
- collect and store a number n (decided by the user) of depth maps;

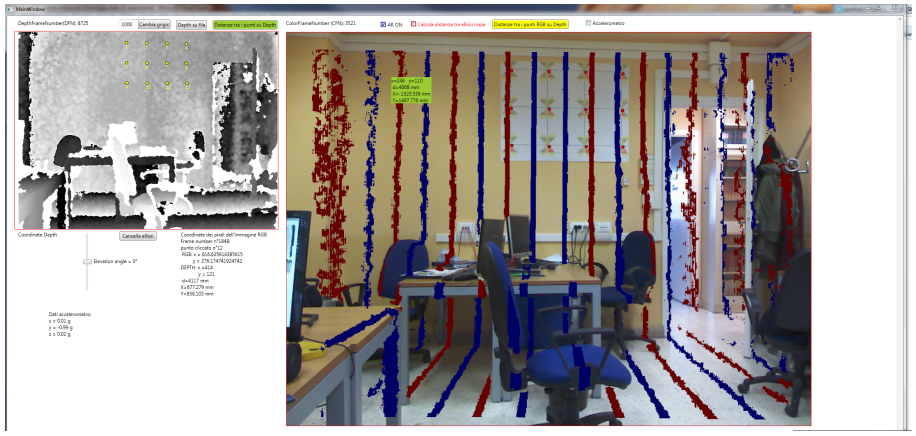


Fig. B.1: The application interface with the AR effect enabled: the depth image on the left side and the RGB image on the right side.

The two images were aligned using the CoordinateMapper method: in this way the depth data can be retrieved at every RGB pixel which is inside the overlapping region of the two images. So, clicking with the mouse left button on the RGB/depth image, two circles are drawn at the same time on the two images. At the same time, the circle

identification number is also drawn in a textblock under its own circle, assigned in chronological order of selection. The circles point out the selected points: clicking on them with the mouse right button, a popup appears showing the relative coordinate of the point at the given depth frame number. Once the points have been selected (manually or automatically), data capture begins pushing the red checkbox: the coordinates of the selected points and the distances them are saved in two different text files. The program presents also some additional features: the display of the depth data in a textblock next to the mouse pointer icon, the control of the Kinect elevation angle, the display of the accelerometer data (for the Kinect v1) and an augmented reality (AR) sub-application. The first shows the depth value of the pixel pointed by the mouse on the depth or on the RGB image in real time. The second allows to increase and/or to decrease the Kinect v1 elevation angle, in order to frame better the object to measure. The third shows the values of the three accelerometer coordinates in a textblock under the depth image and, by checking the accelerometer checkbox, it allows them to be saved in a textfile. The latter draws red and blue stripes on the RGB image in real time, checking the blue AR checkbox: they are perpendicular to the X axis, therefore they are vertical if the X axis is horizontal and so on (see Fig. B.1).

The application runs on Windows 7 (only Kinect v1) and Windows 8 and Window 10 operative systems.

B.2.1 Software architecture

To retrieve frame data from the Kinect streams, the polling model was used: a fast application was needed and, also if polling is more com-

plicated to implement, it removes the innate overhead associated with events, allowing performance gains [104]. Indeed, polling is a process by which an application manually requests a frame of data from a stream. Each Kinect data stream has a method named OpenNextFrame. When calling the OpenNextFrame method, the application specifies a timeout value, which is the amount of time the application is willing to wait for a new frame. The timeout is measured in milliseconds. The method attempts to retrieve a new frame of data from the sensor before the timeout expires. If the timeout expires, the method returns a null frame [104]. However, by using only the polling, the application remains tied to WPF's UI[1] thread: any long-running data processing or poorly chosen timeout for the OpenNextFrame method can cause slow, choppy, or unresponsive behaviour in the application, because it executes on the UI thread. So it was decided to implement all polling and data processing on a secondary thread, using the background worker class, which allows to work with threads in an easy way [104]. Therefore, the Backgroundworker's DoWork event handler contains the two main methods of the application: the DiscoverKinectSensor method, which initializes the sensor, and the PollImageStream, which is the core of the application.

B.3 Structure Sensor applications

Two different applications were developed for the Structure Sensor: the *Structure Sensor Calibration App* and the *Structure Sensor Measurement tool*. Based on the APIs of the Structure SDK, they are iOS

[1]The WPF's User Interface thread is the thread which physically draws the application window on the pc screen.

applications developed in Objective-C++ with the XCODE IDE. Both integrate the functionalities of OpenCV 3.1 library and run only on iOS devices.

B.3.1 Structure Sensor Calibration App

The Structure Sensor Calibration App was specifically designed to carry out the calibration of the Structure Sensor through the automatic detection and collimation of the corners of a generic chessboard on both IR and color images.

It is a simple and easy to use application that displays the color camera view and IR view, in order to allow the user to frame the calibration grid with both the cameras. The App returns the values of the interior and distortion parameters of IR and RGB cameras and roto-translation parameters of the sensor and display on the iPad screen the calibration reprojection errors. In this way the user can immediately understand if the error committed is below 1 pixel so as to carry out a better data collection phase.

B.3.2 Structure Sensor Measurement Tool

Very similar to Kinect Measurement Tool, but implementing less functionalities, the *Structure Sensor Measurement Tool* was specifically developed in order to evaluate the precision and accuracy of the Structure Sensor. On the left side there is the color image (up view) and the depth map (down view), whereas on the right side the collimated image is visualized, with a 640×480 resolution at 30 fps (see Fig. B.2).

The two images were aligned using *STStreamConfigDepth640×480*

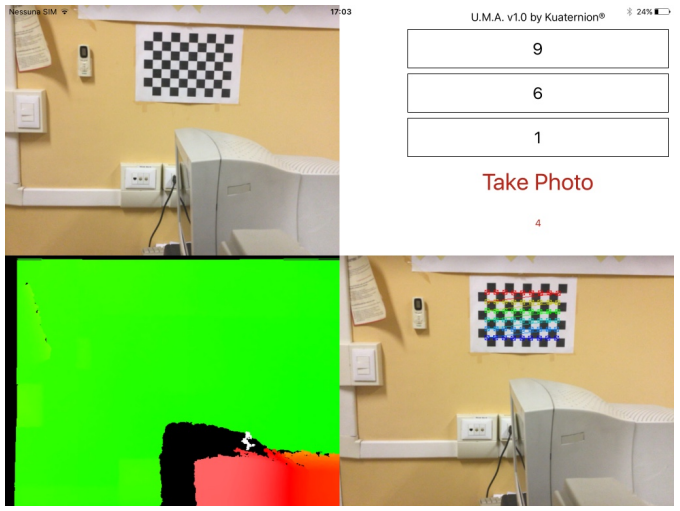


Fig. B.2: Main layout of the developed application.

method, which exploits the calibration parameters saved in the sensor by the Occipital calibrator App. Also displayed on the interface there are the controls to change the size of the grid to be acquired and the nr. of frame that will be captured.

The application allows to automatically collimate the corners of a chessboard grid and to measure the distances between them. Finally the counter below the button "Take Photo" keeps count of the number of acquired frames, since it is possible to collect and store up to 100 depth frames.

B.4 Google Summer of Code

During the summers of 2014 and 2015 the author was involved in the GSoC 2014 and GSoC 2015 programs. GSOC is an annual program

in which Google awards stipends to students who propose projects for free and open source software.

In particular, the LiDAR roof extraction Plug-In for Opticks and the porting of 3D UNDERWORLD-SLS algorithm inside OpenCV were respectively developed during the GSoC 2014 and the GSoC 2015. Both projects were developed using the C++ programming languages.

Being part of such important open source projects has enhanced the author expertise and has contributed in fostering the doctoral research activities.

B.4.1 GSoC 2014: LiDAR roof extraction Plug-In for Opticks

This Plug-In implements a RANSAC-based technique for extracting roof planes of buildings from LiDAR point clouds. It consists of three different stages: raw LiDAR data are first interpolated over a grid with the nearest neighbor interpolation method, in order to generate a DEM raster; then the watershed segmentation algorithm and the connected components approach, which rispectively find and classify the DEM pixels which belong to the buildings, are applied (the DEM must be first divided into a rectangular grid of n $\times$ m tiles in order to improve the results of the segmentation process); finally, the extraction of roof planes is obtained by recursively applying the RANSAC algorithm.

Fig. B.3, Fig. B.4, Fig. B.5 show into details the results for some identified buildings: we can see that *LiDAR Roof Extraction Plug-In* works well on some buildings (for example buildings 128 and 230), but it doesn't work on others (for example building 171). So further developments are needed to improve the results, for example to use

different RANSAC thresholds for every building, to improve the segmentation results (trees and also airplanes are classified as buildings). Moreover, the Plug-In works enough well on this sample point cloud, but it doesn't work so well on others: a generalization of the threshold selection in the watershed algorithm must be also done. Further information can be found at [88].

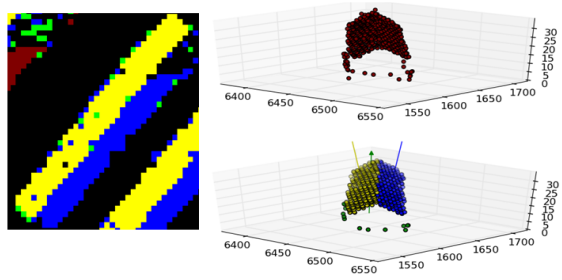


Fig. B.3: LiDAR Roof Extraction Plug-In results for the building 128.

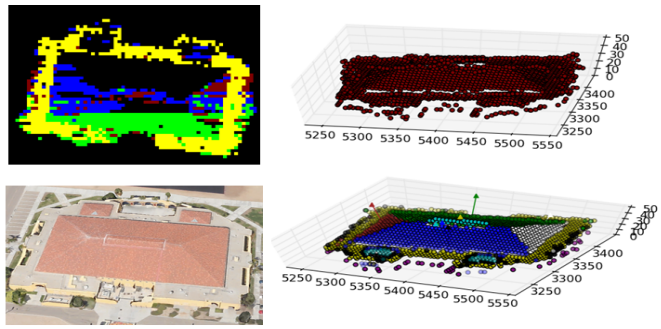


Fig. B.4: LiDAR Roof Extraction Plug-In results for the building 230.

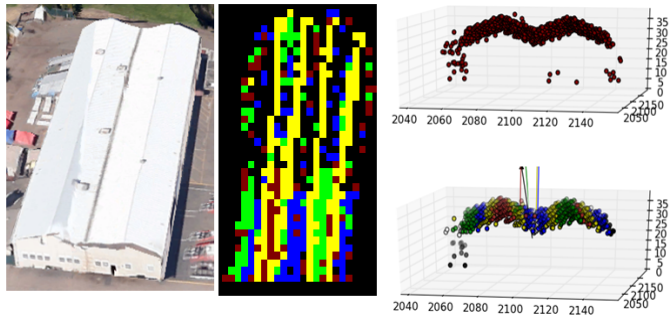
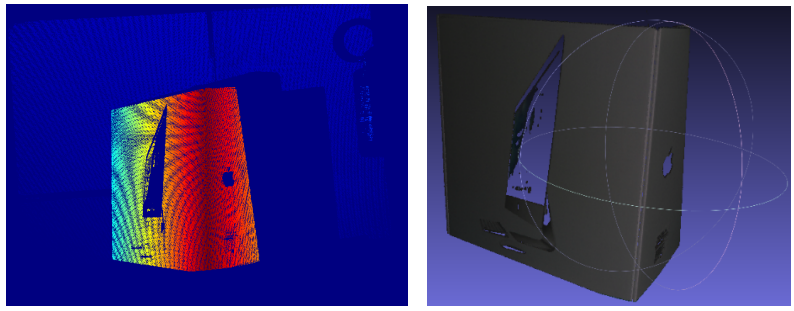


Fig. B.5: LiDAR Roof Extraction Plug-In results for the building 171.

B.4.2 GSoC 2015: Structured Light module for OpenCV

The 3D UNDERWORLD-SLS algorithm [53], an open source structured-light scanning system for rapid geometry acquisition, was ported inside the OpenCV library, developing the structured-light module.



(a) Disparity map.

(b) Dense point cloud with texture.

Fig. B.6: A cardboard modelled with the OpenCV implementation of the 3D UNDERWORLD-SLS algorithm.

In particular this algorithm implements a stereo approach where:

- the generation of the pattern images is performed with Gray encoding using the traditional white and black colors;
- the information about the two image axes x, y is encoded separately into two different pattern sequences (one for the columns and one for the rows);
- the generated pattern sequence consists of both regular color and color-inverted images: effective method for easily determining the intensity value of each pixel when it is lit (highest value) and when it is not lit (lowest value);
- each pixel in the captured images is decoded into its corresponding decimal numbers, respectively representing the projector column and row;
- mapping between the pixels in the captured images which correspond to the same projector pixel (disparity computation);
- the acquired pattern images must be previously rectified: 3D reconstruction is performed using OpenCV reprojectImageTo3D method.

Fig. B.4.2 show some results. More details can be found in [89].

References

[1] Intel RealSense Cross Platform API. https://github.com/IntelRealSense/librealsense. Last edited: December 2016. 141

[2] Intel RealSense SDK 2016 R3 Documentation: 3D scan module. https://software.intel.com/sites/landingpage/realsense/camera-sdk/v2016r3/documentation/html/index.html?doc_scan_3d_scanning.html. Last edited: 2016. 159, 160

[3] Intel RealSense SDK Details. https://software.intel.com/en-us/intel-realsense-sdk/details. Last edited: 2016. 140

[4] Kinect for Windows SDK 1.8 Architecture. https://msdn.microsoft.com/en-us/library/jj131023.aspx. Last edited: September 2013. xiv, 136, 137, 138

[5] Kinect for Windows SDK 2.0 API Overview. https://msdn.microsoft.com/en-us/library/dn782033.aspx. Last edited: October 2014. xiv, 138, 139

[6] Kinect Fusion. https://msdn.microsoft.com/en-us/library/dn188670.aspx. Last edited: September 2013. xv, 146, 149, 156, 157

[7] OpenKinect website. https://openkinect.org/wiki/Main_Page. Last edited: March 2012. 135

[8] OpenNI programmer's guide. http://com.occipital.openni.s3.amazonaws.com/OpenNI_Programmers_Guide.pdf. Last edited: April 2013. 133, 134

[9] PCL web site. http://pointclouds.org/. xiv, xv, 142, 144, 158

[10] ReconstructMe web site. http://reconstructme.net/. xv, 166, 167

[11] SCANDY web site. https://scandy.co/welcome. 164

[12] SKANECT web site. http://skanect.occipital.com/. xv, 165

[13] Structure SDK Reference. Structure SDK documentation. Last edited: September 2016. 139, 140, 161

[14] Unofficial OpenNI website. http://www.openni.ru/about/index.html. Last edited: April 2014. xiv, 132, 133

[15] Agisoft PhotoScan web page. http://www.agisoft.com/, 2017. 106, 119

[16] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. Point Cloud Library: Three-Dimensional Object Recognition and 6 DoF Pose Estimation. *IEEE Robotics & Automation Magazine*, 1070(9932/12), 2012. 142, 145

[17] A. Armstrong. OpenNI To Close. `http://www.i-programmer.info/news/194-kinect/7004-openni-to-close-.html`. Last edited: February 2014. 132

[18] A. Baksheev. KinFu remake. `https://github.com/Nerei/kinfu_remake`. Last edited: March 2016. 158, 159

[19] E. Benedetti, R. Ravanelli, M. Moroni, A. Nascetti, and M. Crespi. Exploiting Performance of Different Low-Cost Sensors for Small Amplitude Oscillatory Motion Monitoring: Preliminary Comparisons in View of Possible Integration. *Journal of Sensors*, 2016:10, 2016. 94

[20] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer graphics forum*, 21(2):149–172, 2002. 46, 145

[21] P. J. Best and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992. 107, 120, 143, 149

[22] F. Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1), 2004. 12

[23] J. Blake, F. Echtler, and C. Kerl. libfreenect2. `https://github.com/OpenKinect/libfreenect2`. Last edited: March 2012. 135

[24] W. Böhler and A. Marbs. 3d scanning and photogrammetry for heritage recording: a comparison. In *Proceedings of the 12th International Conference on Geoinformatics*, pages 291–298. Citeseer, 2004. 12

[25] A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shakensense: Reducing interference for overlapping structured light depth cameras. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, May 2012. 153

[26] B. Büttgen, T. Oggier, M. Lehmann, R. Kaufmann, and F. Lustenberger. CCD/CMOS Lock-In Pixel for Range Imaging: Challenges, Limitations and State-of-the-Art. *1st Range Imaging Research Day*, pages 21–32, 2005. 17, 20

[27] B. Carrihill and R. Hummel. Experiments with the intensity ratio depth sensor. *Computer Vision, Graphics, and Image Processing*, 32(3):337–358, 1985. x, 32, 33

[28] D. Cathue. *Programming with the Kinect for Windows Software Development Kit*. Microsoft Press, 2012. 52

[29] E. Charbon, M. Fishburn, R. Walker, R. K. Henderson, and C. Niclass. Spad-based sensors. In *TOF range-imaging cameras*, pages 11–38. Springer, 2013. 16

[30] F. Chiabrando, R. Chiabrando, D. Piatti, and F. Rinaudo. Sensors for 3D Imaging: Metric Evaluation and Calibration of

a CCD/CMOS Time-of-Flight Camera. *Sensors*, 9(12):10080, 2009. 23

[31] Chipworks. chipworks web page. `http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-xbox-one-kinect/`, 2013. 53

[32] J. Chow, K. Ang, D. Lichti, and W. Teskey. Performance Analysis of a Low-Cost Triangulation-Based 3d Camera: Microsoft Kinect System. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:175–180, 2012. 45, 61

[33] W. contributors. Objective-C. `https://en.wikipedia.org/w/index.php?title=Objective-C&oldid=754691927`. Last edited: December 2016. 177, 178

[34] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo. *Time-of-Flight Cameras and Microsoft Kinect^{TM}*. Springer Science & Business Media, 2012. ix, x, xi, 6, 7, 13, 17, 18, 22, 23, 28, 29, 30, 31, 32, 39, 41, 42, 45, 56, 59

[35] L. Della Ventura. Low cost range camera geomatics applications: an innovative calibration method for Microsoft Kinect. M.S. Thesis, University of Rome "La Sapienza", Faculty of Civil and Industrial Engineering, 2012-2013. 84

[36] S. Dixon-Warren. Inside the Intel RealSense Gesture Camera. `http://www.chipworks.com/about-chipworks/overview/blog/inside-the-intel-realsense-gesture-camera`. Last edited: July 2015. 40, 41

[37] N. G. Durdle, J. Thayyoor, and V. Raso. An improved structured light technique for surface reconstruction of the human trunk. In *Electrical and Computer Engineering, 1998. IEEE Canadian Conference on*, volume 2, pages 874–877. IEEE, 1998. x, 36

[38] T. Etzion. Constructions for Perfect Maps and Pseudorandom Arrays. *IEEE Transactions on information theory*, 34(5):1308–1316, 1988. 36

[39] G. D. Evangelidis, M. Hansard, and R. Horaud. Fusion of Range and Stereo Data for High-Resolution Scene-Modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(11):2178–2192, 2015. 106

[40] G. P. F. Jurado and F. Flores. 3d color-based tracking system for real-time using kinect sensor. In *SOMIXXVII Congreso de Instrumentacin*, 2012. 94

[41] D. Fofi, T. Sliwa, and Y. Voisin. A comparative survey on invisible structured light. In *Electronic Imaging 2004*, pages 90–98. International Society for Optics and Photonics, 2004. 24, 25

[42] S. Foix, G. Alenyà, and C. Torras. Lock-in Time-of-Flight (ToF) Cameras: A Survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011. 18

[43] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns, Apr. 3 2012. US Patent 8,150,142. 42

[44] J. Geng. Rainbow three-dimensional camera: new concept of high-speed three-dimensional vision systems. *Optical Engineering*, 35(2):376–383, 1996. x, 33

[45] J. Geng. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011. x, 26, 27, 31, 34, 35, 37

[46] G. Gerig. Structured Lighting. 3D Computer Vision lecture at University of Utah, 2012. ix, 25, 33

[47] D. Girardeau-Montaut. Cloud Compare3D Point Cloud and Mesh Processing Software - Version 2.7.0. *Open Source Project*, 2016. 107, 118, 120

[48] P. M. Griffin, L. S. Narasimhan, and S. R. Yee. Generation of uniquely encoded light patterns for range data acquisition. *Pattern recognition*, 25(6):609–616, 1992. x, 38

[49] I. Griffiths. *Programming C# 5.0*, chapter 1, Introducing C#. O'Reilly Media Inc., March 2012. 174, 175

[50] M. Grzegorzek, C. Theobalt, R. Koch, and A. Kolb. *Time-of-Flight and Depth Imaging - Sensors, Algirithms, and Applications*. Springer, 2012. 13

[51] S. Á. GuOmundsson, H. Aanæs, and R. Larsen. Environmental effects on measurement uncertainties of time-of-flight cameras. In *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, volume 1, pages 1–4. IEEE, 2007. 68

[52] M. Hansard, S. Lee, O. Choi, and R. P. Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Science & Business Media, 2012. 13

[53] K. Herakleous and C. Poullis. 3DUNDERWORLD-SLS: An Open-Source Structured-Light Scanning System for Rapid Geometry Acquisition. *arXiv preprint arXiv:1406.6595*, 2014. x, 33, 34, 186

[54] T. Hoffman. Occipital Structure Sensor. `http://www.pcmag.com/article2/0,2817,2463858,00.asp`. Last edited: August 2014. 40

[55] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011. xv, 46, 145, 146, 147, 148, 149, 150

[56] T. Kahlmann, F. Remondino, and H. Ingensand. Calibration for increased accuracy of the range imaging camera swissrangertm. *Image Engineering and Vision Metrology (IEVM)*, 36(3):136–141, 2006. ix, 14

[57] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg. Omnikinect: Real-time dense volumetric data acquisition and applications. *Proceedings of the 18Th Acm Symposium*

On Virtual Reality Software And Technology (VRST), December 2012. xv, 152

[58] M. Kalantari and M. Nechifor. Accuracy and utility of the Structure Sensor for collecting 3D indoor information. *Geo-spatial Information Science*, pages 1–8, 2016. xi, 44

[59] S. Kean, J. Hall, and P. Perry. *Meet the Kinect: An Introduction to Programming Natural User Interfaces*. Apress, 2011. 40, 135, 136

[60] K. Khoshelham and S. O. Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, 2012. 41, 43

[61] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight cameras in computer graphics. In *Computer Graphics Forum*, volume 29, pages 141–159. Wiley Online Library, 2010. 20, 21, 22, 23

[62] K. Konolige and P. Mihelich. OpenKinect: Ros Technical Description of Kinect Calibration. `http://wiki.ros.org/kinect_calibration/technical`. Last edited: December 2012. 42

[63] E. Lachat, H. Macher, M. Mittet, T. Landes, and P. Grussenmeyer. First experiences with Kinect v2 sensor for close range 3D modelling. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):93, 2015. 23, 61

[64] R. Lange. *3D Time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology*. PhD thesis, University of Siegen, Germany, 2000. x, 26, 27, 28

[65] L. Lastilla. Structure Sensor and RGB cameras integration for point cloud enhancement: first archaeological applications. B.S. Thesis, University of Rome "La Sapienza", Faculty of Civil and Industrial Engineering, 2015-2016. 119

[66] J. Le Moigne and A. M. Waxman. Structured light patterns for robot mobility. *IEEE Journal on Robotics and Automation*, 4(5):541–548, 1988. x, 38

[67] L. Li. Time-of-flight camera–an introduction. *Technical White Paper, May*, 2014. ix, 14, 15, 16, 21

[68] N. Lievendag. Itseez3D (4.1) 3D scanner app review. `http://3dscanexpert.com/structure-sensor-review-part-2-itseez3d/`. Last edited: November 2016. 163, 164

[69] T. Liu, A. W. Burner, T. W. Jones, and D. A. Barrows. Photogrammetric techniques for aerospace applications. *Progress in Aerospace Sciences*, 54:1–58, 2012. ix, 8, 9, 10

[70] T. Luhmann, S. Robson, S. Kyle, and I. Harley. *Close range photogrammetry: principles, techniques and applications*. Whittles, 2006. ix, 8, 9, 10

[71] M. MacDonald. *Pro WPF in C ♯ 2010*, chapter 1, Introducing WPF; chapter 2, XAML. Apress, 2010. 176

[72] M. Maruyama and S. Abe. Range sensing by projecting multiple slits with random cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):647–651, 1993. x, 36

[73] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissanov. Structured light using pseudorandom codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):322–327, 1998. x, 38

[74] T. Moynihan. This New iPad App Makes Futuristic 3D Scans of Your Home. `https://www.wired.com/2016/11/occipital-canvas-app-for-structure-camera/`. Last edited: November 2016. 162, 163

[75] F. Mufti and R. Mahony. Statistical analysis of measurement processes for time-of-flight cameras. In *SPIE Optical Engineering+ Applications*, pages 74470I–74470I. International Society for Optics and Photonics, 2009. ix, 18, 19

[76] R. Nair, K. Ruhl, F. Lenzen, S. Meister, H. Schäfer, C. S. Garbe, M. Eisemann, M. Magnor, and D. Kondermann. A Survey on Time-of-Flight Stereo Fusion. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 105–127. Springer, 2013. 106

[77] T. Nakamura. Real-time 3-d object tracking using kinect sensor. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, pages 147–153, 2011. 94

[78] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgib-

bon. Kinectfusion: Real-time dense surface mapping and tracking. *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, October 2011. 46, 145, 146, 147, 148

[79] E. Nibi. From computer vision to geomatics: high precision calibration and metric potentialities of the "Occipital Structure Sensor©". M.S. Thesis, University of Rome "La Sapienza", Faculty of Civil and Industrial Engineering, 2014-2015. 88

[80] Occipital. VR DEV KIT: OUR NEXT GENERATION OF MOBILE 6-DoF POSITIONAL TRACKING. `http://structure.io/use/vr-positional-tracking`. Last edited: December 2016. 129

[81] D. Pagliari, F. Menna, R. Roncella, F. Remondino, and L. Pinto. Kinect fusion improvement using depth camera calibration. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):479–485, 2014. 68

[82] D. Pagliari and L. Pinto. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. *Sensors*, 15(11):27569–27589, 2015. 67

[83] D. Piatti and F. Rinaudo. SR-4000 and CamCube3.0 Time of Flight (ToF) Cameras: Tests and Comparison. *Remote Sensing*, 4(4):1069, 2012. 15, 23

[84] PrimeSense. PrimeSense 3D Sensors. `http://www.i3du.gr/pdf/primesense.pdf`. Last edited: November 2013. xi, 40

[85] X. Qi, D. Lichti, M. El-Badry, J. Chow, and K. Ang. Vertical dynamic deflection measurement in concrete beams with the microsoft kinect. *Sensors*, 14(2):3293–3307, 2014. 94

[86] D. Quaglio. Potenzialitá dello Structure Sensor per la realizzazione real time di planimetrie di ambienti indoor. B.S. Thesis, University of Rome "La Sapienza", Faculty of Civil and Industrial Engineering, 2014-2015. 111

[87] J.-Y. Rau and P.-C. Yeh. A semi-automatic image-based close range 3d modeling pipeline using a multi-camera configuration. *Sensors*, 12(8):11271, 2012. 11, 12

[88] R. Ravanelli. Documentation of LiDAR segmentation Plug-In for Opticks. `https://github.com/RobertaRavanelli/Opticks_GSoC2014/blob/master/GSoC_2014_Ravanelli.pdf`. Last edited: August 2014. 185

[89] R. Ravanelli. GSoC 2015 OpenCV Structured Light. `https://www.youtube.com/watch?v=O5P65CYqo_Q`. Last edited: August 2015. 33, 187

[90] R. Ravanelli, A. Nascetti, and M. Crespi. Kinect V2 And RGB Stereo Cameras Integration For Depth Map Enhancement. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B5:699–702, 2016. 104

[91] F. Remondino and S. El-Hakim. Image-based 3D modelling: A review. *The Photogrammetric Record*, 21(115):269–291, 2006. 6, 7, 11

[92] F. Remondino, A. Guarnieri, and A. Vettore. 3d modeling of close-range objects: photogrammetry or laser scanning? In *Electronic Imaging 2005*, pages 216–225. International Society for Optics and Photonics, 2005. 11, 12

[93] F. Remondino and D. Stoppa. *TOF Range-Imaging Cameras*. Springer, 2013. 13, 22

[94] J. M. Rüeger. *Electronic distance measurement - An introduction*. Springer Science & Business Media, 1996. 21

[95] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D Model Acquisition. *ACM Transactions on Graphics (TOG)*, 21(3):438–446, 2002. ix, 24

[96] R. B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011. 142, 143, 144

[97] J. Salvi, J. Pagés, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, April 2004. x, 29, 32, 34, 35, 36, 37

[98] H. Sarbolandi, D. Lefloch, and A. Kolb. Kinect range sensing: Structured-light versus Time-of-Flight Kinect. *Computer Vision and Image Understanding*, 139:1–20, 2015. 22, 23, 45, 46, 61

[99] P. Seitz. Unified analysis of the performance and physical limitations of optical range-imaging techniques. *1st Range Imaging Research Day*, pages 9–17, 2005. 17

[100] L. Shindler, M. Moroni, and A. Cenedese. Using optical flow equation for particle identification and velocity prediction in particle tracking. *Applied Mathematics and Computation*, 218:8684–8694, 2012. 98

[101] A. Shpunt and B. Pesach. Optical pattern projection, Feb. 26 2013. US Patent 8,384,997. xi, 41

[102] vgleaks. vgleaks web page. `http://www.vgleaks.com/durango-next-generation-kinect-sensor`, 2013. 51

[103] W. Wang, W. Zhao, L. Huang, V. Vimarlund, and Z. Wang. Applications of terrestrial laser scanning for tunnels: a review. *Journal of Traffic and Transportation Engineering (English Edition)*, 1(5):325–337, 2014. 12

[104] J. Webb and J. Ashley. *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012. 181

[105] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. *Computer Science and Artificial Intelligence Laboratory Technical Report*, July 2012. xv, 150, 151

[106] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555. IEEE, November 2013. 151, 152

[107] O. Yilmaz and F. Karakus. Stereo and kinect fusion for continuous 3d reconstruction and visual odometry. In *Electronics, Computer and Computation (ICECCO), 2013 International Conference on*, pages 115–118. IEEE, 2013. 106

[108] L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 24–36. IEEE, 2002. x, 37

[109] J. Zhu, L. Wang, R. Yang, and J. Davis. Fusion of Time-of-Flight Depth and Stereo for High Accuracy Depth Maps. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 106