Correction and Interpolation of Depth Maps from Structured Light Infrared Sensors

Simone Milani and Giancarlo Calvagno

Dept. of Information Engineering, University of Padova
via Gradenigo 6/B, 35131 Padova, Italy
Tel.: +39-049-827-{7641,7731}
Fax: +39-049-827-7699
e-mail: `simone.milani@dei.unipd.it,calvagno@dei.unipd.it`

Abstract

Infrared structured light sensors are widely employed for control applications, gaming, acquisition of dynamic and static 3D scenes. Recent developments have lead to the availability on the market of low-cost sensors, like Kinect devices, which prove to be extremely sensitive to noise, light conditions, and the geometry of the scene.

The paper presents a quality enhancement strategy for Kinect-acquired depth maps that corrects depth values via a set of local differential equations and interpolates the missing depth samples. The approach improves both density and accuracy of the generated 3D model under different light conditions and in the presence of cross-talk noise derived from other devices. Moreover, the paper introduces a new experimental reference dataset for Kinect denoising algorithms consisting in multiple acquisition under different noise conditions associated to a laser scan model of the scene (ground truth).

Keywords: DIBR denoising, MS Kinect, structured light camera, infrared sensor, 3D acquisition

1. Introduction

The recent availability of low-cost range cameras has shaken the ICT world leading to a flourishing of new object recognition applications, human-computer interfaces, and acquisition systems of dynamic 3D scenes. Time-of-Flight cameras [1], structured light 3D scanners [2], and multicamera systems allow easy real-time acquisition of dynamic 3D scenes with both static and dynamic elements [3].

Among these, the Xbox Kinect sensor [4], which includes a standard RGB camera together with an infrared (IR) structured light scanner (see Fig. 1), has recently proved to be one of the most widely-used depth sensors thanks to its versatility, the limited cost and the improved performance in a wide range of possible applications.

In [5] Leyvand *et al.* discuss new possibilities in tracking persons and identifying their identity. Suma *et al.* present a novel toolkit to model human gestures and poses [6]. In [7] Wilson presents a touch interface implemented using a depth camera. Moreover, several navigation tools employ kinect sensor to accurately control and drive robots in an indoor environment (see Turtlebot by Willow Garage [8], as an example). Other approaches employ Kinect for modeling ground surface in precision agriculture [9], providing a reliable visual feedback to the user in virtual dressing rooms [10], managing interaction in augmented reality [11] or in virtual environments [12].

The structure of the Xbox Kinect device is reported in the diagram of Fig. 1. The implemented IR depth sensor consists in an IR projector, an IR CMOS camera, and a processing unit that controls them and elaborates the acquired signal. An IR pattern of dots is projected by the IR projector on the scene, and the IR CMOS camera acquires the reflected pattern, which will be distorted according to the geometry of the objects. The central processing unit estimates the distance of each point from the depth camera considering the distortions in the acquired dot pattern with respect to the projected one. Color information is available as well since an RGB CMOS camera permits obtaining a standard picture of the acquired scene. This information permits building a point cloud model of the 3D scene by mapping depth pixels into color pixels with a warping operation.

Unfortunately, despite the strong versatility and the wide range of new applications of these devices, the resulting depth signal is affected by a significant amount of noise.

Like many imaging systems, depth sensors present shot noise related to the radiation, A/D conversion quantization noise, and thermal noise. Moreover, in a real scene IR sensors receive a significant amount of radiation that has not been generated by the the associated projector. This is the case of ambient illumination (e.g., sun or artificial lights whose electromagnetic radiations also span the IR frequency of MS Kinect) or, in case multiple similar IR structured light cameras are present in the same scene, the dot patterns produced by a different device [13]. In a controlled environment, this inconvenience can be mitigated by time division multiplexing [14], motion (shake-and-sense [15]) or by sample removal and hole filling [16]. In a distributed system, e.g., an environment where multiple Kinect sensors are operated independently, like in case of multiple Kinect-controlled robots [17, 18], this inconvenience must be addressed by effective interpolation and denoising algorithms.

The paper presents a joint denoise-interpolation algorithm for MS Kinect sensor that aims at correcting the computed depth values and interpolate them whenever they are not available because of the noise conditions. The approach relies on an initial denoising performed by matching borders between range and color images. The mismatches between edge information from RGB data and depth maps can be corrected by a set of differential local equations. Then, the resulting data are propagated to neighboring locations where depth samples are not available according to the edge information. The proposed solution improves the approach in [19] by introducing a new set of denoising and in-

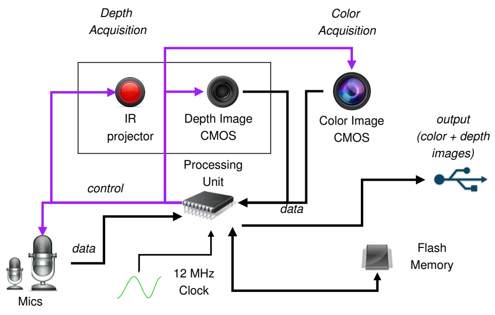


Figure 1: Block diagram of the MS Kinect sensor.

terpolating equations for depth samples and optimizing their implementation in order to run in real time. Experimental testing was improved as well using a high-quality laser-scanned 3D model as ground truth reference in order to evaluate the algorithm performance. This activity led to the creation of a downlodable dataset with multiple acquisitions under different illumination conditions. The reported results show that the algorithm increases the number of available depth samples and refines the accuracy of the generated point clouds. Moreover, the proposed solution can run in real time allowing a frame rate of approximately 17 frame/s. In the following, Section 2 overviews some of the existing works in literature, while Section 3 presents the characteristics and the effects of noise on depth signals. Section 4 describes the proposed algorithm in detail, with Subsection 4.3 reporting the depth correction process and Subsection 4.5 presenting the interpolation strategy. Experimental results (Section 5) and conclusions (Section 6) end the paper.

2. Related works

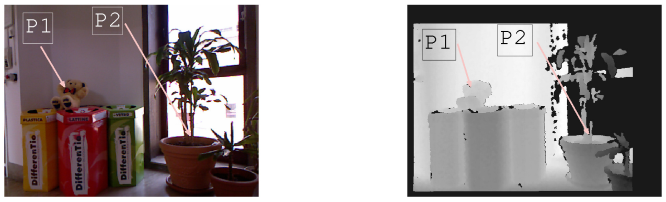
Several works have proposed novel denoising algorithms to improve the quality of the acquired depth maps. This is severely impaired by three main issues: the inaccuracy of depth samples due to the presence of noise, the irregularity of object boundaries, and the presence of missing depth samples (holes) in the acquired depth maps.

Depth samples are usually corrected by filtering the depth data in order to reduce the noise level. The work in [20] reviews the performance of different types of filters (median, bilateral, joint bilateral, non-local means, with adaptive thresholds, and temporally smoothing) of the data generated by a Kinect sensor. Experimental results show that temporal denoising performs extremely well in most of the situations although it requires multiple acquisition along time. For single-acquisition depth maps, joint bilateral filters perform well enough thanks to their edge-preserving properties and require a limited computational complexity [21]. They rely on tuning the weights of a standard bilateral filter [22] according to the color information. Joint bilateral filters have been evaluated in many works considering different devices and different data format [23]. As an example, the work by Huhle *et al.* [24] targets Time-of-Flight (ToF) devices coupled with an RGB camera. In the case of ToF cameras, some solutions also consider the confidence values to denoise depth acquired via a ToF camera [25]. Side RGB color information have also been effectively employed to enhance the resolution of depth images, which is usually much lower with respect to that of color data [26]. The solution in [27] performs an edge detection approach on both color and depth images, discards irrelevant edges, and applies a directional joint bilateral filter along the object borders. In [28] filter weights are generated using a Common Distance Transform (CDT), which models the degree of pixel-modal similarity between depth pixels and the corresponding color pixels. The solution in [29] employs a joint bilateral filter that refines the object boundaries and fills holes by adapting the filtering parameters according to depth values. The resulting depth map permits improving the smoothness of object contours and the performance of object detection algorithms that exploit the RGBD signal.

Inpainting and interpolation algorithms are usually employed to estimate the missing data samples. Like in the previously-mentioned approaches, most of the proposed solutions jointly process both depth and color signals. The solution in [30] adopts a structure-guided fusion strategy that enacts a weighted estimation of the missing depth samples starting from the neighbouring available ones. Weights are computed according to geometrical distance, depth similarity and color data. Other solutions segment color information in order to find the boundaries where interpolation of valid depth samples can take place [31]. This strategy relies on the fact that object shapes must be the same both for color and depth data: such assumption can also be used to perform depth superresolution [32] or no-reference quality evaluation [33]. Since good segmentation algorithms require some computational effort, it is possible to reduce the computing effort by employing edge information to drive the inpainting operations. In the paper [34], edges are computed to generate a higher resolution depth image. The solution in [19] enacts an edge-constrained hole filling interpolation to improve the quality of Kinect-acquired depth maps. Other solutions combines non-linear processing strategies with traditional wavelet decomposition in order to suit the piecewise-smooth characteristics of depth maps [35].

In case time delay is not an issue, it is possible to merge multiple acquisitions to improve the quality of the acquired 3D model [36]. This solution permits obtaining good performances, as shown in [20] but introduces an additional delay in outputting the acquired depth map since more than one acquisition is required. In a Depth Image Based Rendering (DIBR) video with high requirements in terms of frame rate, the denoising approach must be faster and avoid processing multiple depth frames.

To this purpose, the proposed strategy processes single couples of RGB and depth images. It inherits the estimation strategy of [19] in order to find the mismatches between color and depth edges. With respect to [19], depth correction and fusion strategies are improved while avoiding an increase in terms of computational complexity.



(a) (b)

Figure 2: Example of kinect acquisition. Images show (a) color and (b) depth components. Corresponding points P1 and P2 are highlighted in the images related to `bearbins`.

3. Problem statement

As mentioned before, the MS Xbox Kinect device implements a low-cost 3D sensor based on an IR structured light camera. The device acquires an RGB color image of the scene together with a depth map of the acquired scene. This information permits building a point cloud model of the 3D scene by mapping depth pixels into color pixels with a warping operation. Unfortunately, the acquired depth maps present several artifacts depending on possible calibration errors, lighting conditions and errors in depth estimation by the processing unit.

Some of the errors occur since the point cloud model is obtained from a warping operation which relies on camera parameters estimated from a calibration process. The accuracy of warping depends on the region of the image where the object is projected, the resolution of the image and the number of images used in the calibration [37]. Moreover, the estimation process that infers scene geometry from the deformations of the acquired dot pattern with respect to its projected version proves to be less accurate along object boundaries. Figure 2 shows that in the warped depth map the borders of the objects are highly irregular and sometimes prove to be shifted with respect to the corresponding color signal. Several holes and missing depth values can be noticed since for those points depth estimation can not provide a sufficiently accurate value. As an example, the difference between coordinates for point P1 is $(+3, +1)$ (no direct light), while it is $(-10, +3)$ for point P2 (direct light from window).

From these premises, it is possible to decompose the IR image acquired by the depth sensor into a desired component, which has been generated by the IR projector and reflected by the objects in the scene, and a noise component produced by surrounding light sources or other IR depth sensors in the scene. In Figure 2, it is possible to notice that most of the holes and the irregularities lie along the borders directly illuminated by the sun light coming from the window.

This leads to noisy signals that impair the performance of depth processing application and the accuracy of the generated 3D model. In the following section, we will present a depth denoising strategy that aims at mitigating these effects.

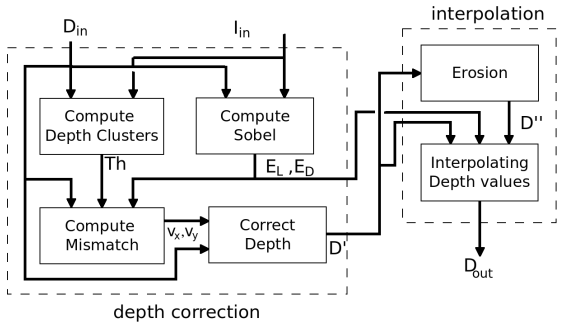


Figure 3: Block diagram of the proposed algorithm.

4. Description of the proposed algorithm

The structure of the algorithm is summarized in Fig. 3 and consists in two main operating blocks: a denoising unit that corrects mismatches between the color image I_{in} and the warped depth map D_{in}, and an interpolating strategy that fills holes and missing pixels in the range image.

The values of depth samples $D_{in}(x, y)$ are initially clustered into multiple classes, and the mismatches between color edges and depth edges are then compensated by correcting the depth values. Then, noisy depth samples along object borders are removed, and the missing depth data is recovered via an edge-driven interpolation.

The following subsections will describe each step in detail.

4.1. Clustering depth values

At the beginning of the depth correction strategy, the depth values $D_{in}(x, y)$ (where (x, y) are the pixel coordinates) are clustered into a set of $N_c = 10$ classes R_k, $k = 0, \ldots, 9$, according to their distance from the IR camera using the k-means algorithm [38]. The choice of using k-means algorithm and computing N_c classes was driven by the need of having a low complexity architecture.

Each class is characterized by its centroid and two threshold values that defines the upper and the lower bounds for depth values, which are grouped into the set of thresholds $\mathbf{Th} = [Th(k)]_k$. Therefore, region R_k can be written as

$$R_k = \{(x, y) : Th(k-1) \geq D_{in}(x, y) < Th(k)\}. \tag{1}$$

Note that thresholds adapt to the specific depth map values.

4.2. Computing mismatches

At the beginning of the error correction unit, 3×3 Sobel operators S_x and S_y are applied to both the luminance component L of the color image and the warped depth image D_{in}. Let $S_x * L$ and $S_y * L$ be the convolutions of L with the horizontal and the vertical Sobel operators, respectively, and $S_x * D_{in}$ and

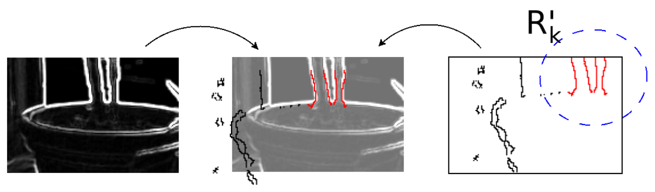


Figure 4: Example of computation of $\mathbf{v}^*$ for the class R'_k (detail from the scene `bearbins`).

$S_y * D_{in}$ be the convolution of the same operators with D_{in}. Then, the edge images E_L and E_D are computed as

$$\begin{aligned} E_L &= round\left(\frac{|S_x * L| + |S_y * L|}{2}\right) \\ E_D &= round\left(\frac{|S_x * D_{in}| + |S_y * D_{in}|}{32}\right) \end{aligned} \tag{2}$$

where quantization steps 2 and 32 have been chosen from a set of experimental trials. In equation (2), coordinates (x, y) have been omitted for the sake of conciseness. For all the pixel positions (x, y) such that $D_{in}(x, y)$ is not valid, $E_D(x, y)$ is set to 0.

In a second step, the mismatches between I_{in} and D_{in} are computed for each class R_k independently generating the pixel sets

$$R'_k = \{(x, y) \in R_k : E_D(x, y) > 0\} \tag{3}$$

which comprises points in depth layer k with edge strength greater than 0. For each class R'_k, the algorithm computes the displacement vector $\mathbf{v}^* = [v_x^*, v_y^*]$ in the search window W_{SR} such that

$$\mathbf{v}^* = \arg\max_{\mathbf{v} \in W_{SR}} \sum_{(x,y) \in R'_k} E_L(\overline{x}, \overline{y}). \tag{4}$$

where the coordinates $(\overline{x}, \overline{y})$ depend on the coordinates of the principal point $C = (C_x, C_y)$ so that

$$\overline{x} = x + (2\,\mathcal{I}(x > C_x) - 1) \cdot v_x$$

$$\overline{y} = y + (2\,\mathcal{I}(y > C_y) - 1) \cdot v_y$$

($\mathcal{I}(\cdot)$ is the indicating function).

In this way, the algorithm models the mismatch between edges of the color component and edges of the depth information (see Figure 4). Despite object profiles result irregular in the depth component, the algorithm assumes that errors vary symmetrically with respect to borders and maximizing edge matching permits a correct alignment.

4.3. Correcting depth values

Given the mismatch $\mathbf{v}^*$, it is possible to compensate it computing new values for the related depth samples.

This correction can be obtained differentiating the equations of the pinhole camera model

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{5}$$

and combining it with the function that maps values $D(x, y)$ into distance values

$$z' = 1/(t_1 - t_2\ D(x, y)). \tag{6}$$

Coordinates (x', y', z') identifies the location of the 3D point associated to $D_{in}(x, y)$, f_x, f_y are the focal lengths, and constants t_1, t_2 are obtained from the calibration.

In this way it is possible to model how a change in the edge position may affect the values of depth samples.

In our implementation, the algorithm corrects the positions of pixels in $\mathcal{R}_k$ and replaces the associated depth values $D_{in}(x, y)$ with the value

$$D'(\overline{x}, \overline{y}) = D_{in}(x, y) + \frac{\delta_x D(x, y) + \delta_y D(x, y)}{2} \tag{7}$$

where

$$\delta_x D(x, y) = \frac{\partial D(x, y)}{\partial x} v_x^* \qquad \delta_y D(x, y) = \frac{\partial D(x, y)}{\partial y} v_y^*, \tag{8}$$

and $\overline{x}^* = x + (2\ \mathcal{I}(x > C_x) - 1)\ v_x^*$, $\overline{y}^* = y + (2\ \mathcal{I}(y > C_y) - 1)\ v_y^*$.

In order to simplify the calculation, we approximated $\partial D(x, y)/\partial x$ as

$$\frac{\partial D(x, y)}{\partial x} = \frac{\partial D(x, y)}{\partial(1/z)}\ \frac{\partial(1/z)}{\partial x} \tag{9}$$

(and similarly for $\partial D(x, y)/\partial y$). The partial derivative $\partial D(x, y)/\partial(1/z)$ was obtained from equation (6), while $\partial(1/z)/\partial x$ was derived from the pinhole camera equation $z(x - C_x) = f_x\ x'$. Combining these derivatives into equation (8), it is possible to obtain

$$\begin{aligned} \delta_x D(x, y) &= \frac{-v_x^*/t_2}{f_x\ x'} \simeq v_x^* \frac{t_2\ D_{in}(x, y) - t_1}{t_2(x - C_x)} \\ \delta_y D(x, y) &= \frac{-v_y^*/t_2}{f_y\ y'} \simeq v_y^* \frac{t_2\ D_{in}(x, y) - t_1}{t_2(y - C_y)}. \end{aligned} \tag{10}$$

Note that the amount of depth correction is inversely proportional to the distance: the further the points, the smaller the change. This can be related to the adaptive strategies reported in [20], where the amount of tolerated difference between temporally-adjacent samples depends on the depth value.

At this point, the resulting depth map D' has to be extended in order to fill holes and gaps that lie wherever the depth values estimated by the sensor are not sufficiently reliable.

4.4. Removal of noisy depth samples

In order to improve the accuracy of the estimated 3D model, noisy depth samples need to be removed from the input data and replaced with more accurate estimates that have been obtained interpolating reliable depth information. As observed in Section 3, object borders in depth maps are highly irregular and often neighbor unavailable depth samples. On the contrary, depth data inside objects are more stable and precise because of the characteristics of the depth estimation process. As a matter of fact, it is reasonable to use the latter to correct the first. This operation is performed in two steps. At first, border pixels neighboring with regions of unavailable pixels are removed and labelled as unavailable. Then, unavailable pixels are interpolated from the valid ones according to the edge information computed from the color component. In this section, we present the removal strategy.

Given the depth map D', a binary mask $M(D)$ is generated placing "1" values in the locations of valid depth samples in D' and "0" otherwise. An erosion operation is applied to the mask using as structuring element a diamond with radius 3. In this way, the regions of unavailable depth sample grow including part of the samples along object boundaries.

The resulting depth map D'' is obtained taking from D' only the samples which corresponds to a "1" in the eroded mask. The map D'' is then processed by the interpolation routine that estimates the value of unavailable depth pixels when this is possible. The details of this process will be presented in the following subsection.

4.5. Depth interpolation

Like other algorithms for depth processing [39], our approach resorts to segmentation in order to partition the input depth map into segments where depth signal is assumed to be planar. From this assumption, it is possible to interpolate the missing depth samples from the available valid ones. Unfortunately, segmentation routines require significant additional computational complexity. In order to mitigate the computational load, it is possible to replace the segmentation and interpolation strategy with a simpler interpolation where the weights are computed according to the edge information.

In case the depth sample $D''(x, y)$ at position (x, y) need to be interpolated, the algorithm identifies the locations of the upper, lower, left, and right neighboring valid depth samples, referenced as (x_u, y_u), (x_b, y_b), (x_l, y_l), and (x_r, y_r), respectively. Note that $x_u = x_d = x$ and $y_l = y_r = y$. A valid upper neighbor $D(x_u, y_u)$ is obtained as

$$y_u = \min\{y' : y' > y \ \wedge \ (E_L(x, y') > T_L \vee D(x, y') > 0)\}, \tag{11}$$

and the corresponding distance is computed as

$$d_u = |x_u - x| + |y_u - y|. \tag{12}$$

Note that in case a valid upper neighbor is not available, the algorithm sets d_u to 0. Lower, left, right neighbors and the corresponding distances d_b, d_l, d_r are defined similarly.

From the distance values, it is possible to compute the weights $w_n = d_n/(d_u + d_b + d_l + d_r)$, $n = u, b, l, r$. Then, in case $w_n > 0$, the algorithm sets

$$w_n = \frac{1/w_n}{\sum\limits_{n' : w_{n'} > 0} 1/w_{n'}}; \tag{13}$$

otherwise $w_n = 0$.

The final estimated depth sample is then computed as

$$D_{out}(x, y) = w_u D''(x_u, y_u) + w_b D''(x_b, y_b) + w_l D''(x_l, y_l) + w_r D''(x_r, y_r). \tag{14}$$

In case the depth sample $D''(x, y)$ does not need to be interpolated, the algorithm set $D_{out}(x, y) = D''(x, y)$.

4.6. Differences with respect to prior work

The proposed solution was developed starting from the approach presented in [19], but it currently departs from it in many of its aspects. It is worth underlying that the depth correction strategy (equations (2) and (9-13)) was completely changed computing the depth error propagation from pinhole camera model equations. All the mathematical derivations have been presented in the previous sections. Moreover, depth interpolation does not rely on segmentation any more since the interpolation filter parameters are computed from edge information. This has allowed to improve the performance of the whole strategy and mitigate the computational complexity. Note that the proposed solution operates in real time and requires a limited amount of calculation with respect to other denoising solutions like the joint bilateral filter. In order to provide an experimental evidence for this, the following section presents a performance evaluations for different strategies in terms of accuracy, density, and processing time.

5. Experimental results

The performance of the proposed approach was compared with that obtained by the JBF[21] (with $W = 7$ and $\sigma = 20$ as suggested in [28] and by a set of experimental trials) and that obtained by the TD algorithm in [20]. Four different scenarios were considered in order to test the performance of the algorithm with different geometries: `bear`, `plants`, `pillows`, and `ballbook` (see Fig. 5). Different data sets were acquired changing the illumination of the rooms, and

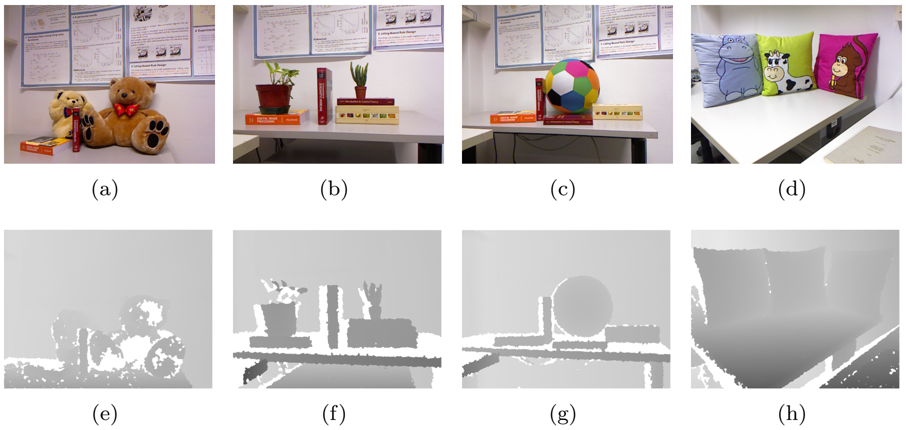


Figure 5: Color and depth components for different acquisition scenarios. a,e) `bear`; b,f) `plants`; c,g) `ballbook`; d,h) `pillows`.

Table 1: Different light conditions.

Label	**Description**
gas	gas light
low	low natural light
strong	strong halogen light
no light	no illumination

the different light conditions are reported in Table 1. Moreover, a second set of acquisitions was taken with a second Kinect projecting its dot patterns on the subject from different angles a (see Fig. 6).

In order to evaluate the performance of the proposed algorithm, we performed 10 different acquisitions of distinct 3D scenes using the MS Kinect sensor under different noise conditions. The resolution of color and depth images are 640×480, and the device was calibrated using the Kinect Calibration Toolbox [40]. The noise level was modified by varying the light conditions and by adding a second Kinect sensor in the scene with different viewing angles (in order to change the amount of interference with our device).

Every scene was also acquired using a NextEngine 2020i 3D laser scanner in order to generate a reference high-quality point cloud model P_0 of the scene as comparison term. Datasets are available at [41].

The quality of the Kinect signal was measured converting depth and color images into a 3D point cloud model P and comparing it with P_0. The comparison was performed by mapping the 3D points of P into the points of P_0 via the ICP algorithm [42]. Each 3D point $\mathbf{p} \in P$ was then associated to the closest point $\mathbf{p}_0 \in P_0$, and the final Mean Square Error (MSE)

$$MSE = E\left[\|\mathbf{p} - \mathbf{p}_0\|^2\right] \tag{15}$$

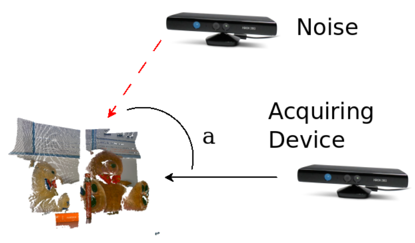


Figure 6: Viewing angles for acquisition sets using an additional Kinect sensor.

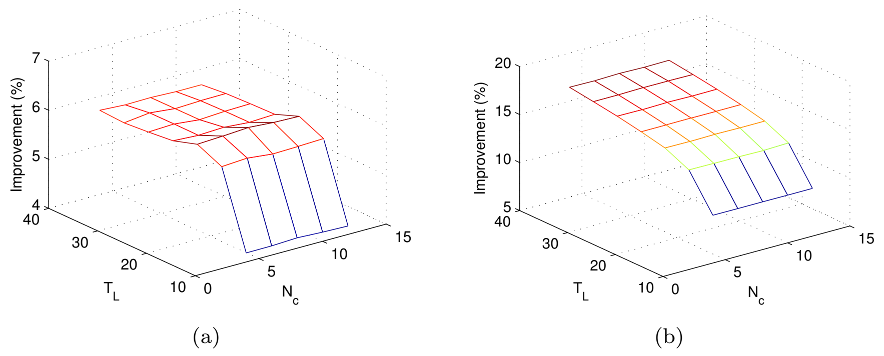


Figure 7: Results for `plant gas` acquisition set for different N_c and T_L values. a) MSE, b) Number of 3D points.

was computed. In addition, the number of valid depth samples is also used as a density measurements for the generated 3D models. Both parameters were averaged over 10 different acquisitions. This averaging was performed in order to smooth the performance of the different algorithms over different noise realizations.

At first, we evaluated the performance of the algorithm for different parameter values. Figures 7 (a) and (b) report the average MSE and the number of points obtained on the acquisition set `plants gas` changing the values of N_c and T_L. It is possible to notice that the influence of the number of classes N_c on the final performance is quite limited. Experimental results show that the optimal values range from 8 to 10. On the other hand, the threshold T_L affects both MSE and the number of interpolated points. It is possible to notice that the higher the threshold, the higher the number of points in the final model.

Figures 8 and 9 report the average MSE values and the number of points for different scenarios. It is possible to notice that the proposed denoising strategy improves the quality of the 3D model under different illumination conditions. The displayed data shows a 12 % reduction of the average MSE value for the acquisition set `plants`. Note also that the difference between the performances of JBF is limited with respect to that of the proposed approach (as it was also noticed in [28]). It is also possible to notice that the density of 3D points

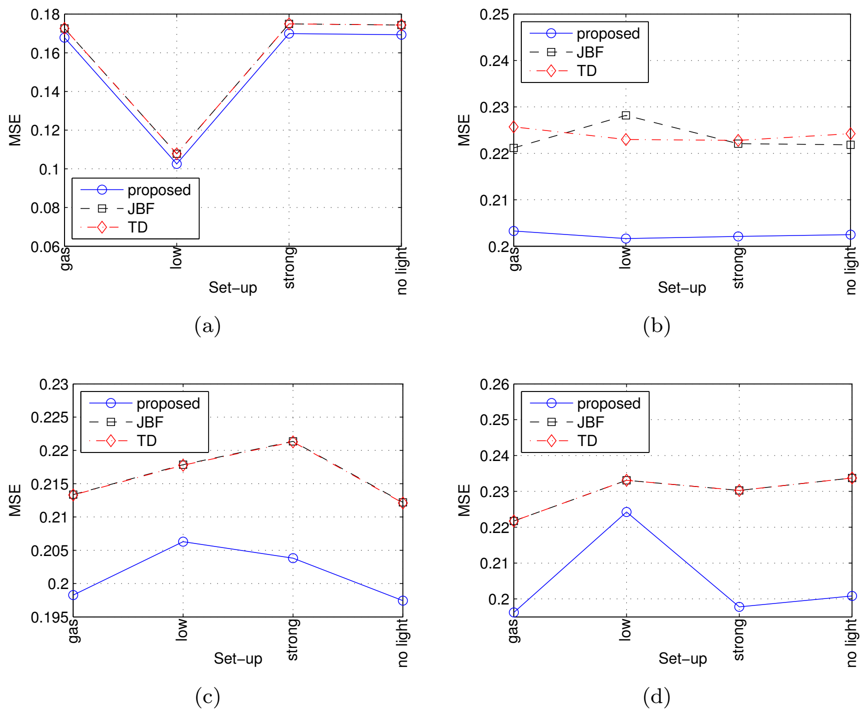


Figure 8: Average MSE values for different scenarios under different illumination conditions and noise conditions. a) `bear`, b) `plants`, c) `pillows`, d) `ballbook`.

increases as well (see Fig. 9). This effect was possible thanks to the interpolation of the missing depth samples.

The displayed results are averaged (for the different acquisition scenarios) and summarized in Table 2.

Figure 10 shows the average MSE for the reconstructed model when the acquisition process is disturbed by a side Kinect device with a viewing angle a with respect to the objects. The proposed solution proves to be helpful also in this case since the average MSE values are reduced. It is possible to notice that the average precision depends on the angle. Whenever devices are positioned at 90 degrees with respect to the scene, the interference is minimal since the amount of radiation emitted by the second Kinect and acquired by the first one is very low. In case the angle decreses, the superposition between different IR patters gets maximum and therefore, the amount of noisy samples increases. Anyway, the fact that the dot patterns of the two devices change and are uncorrelated mitigates the cross-talk interference granting a minimum quality level.

The displayed results are averaged (for the different acquisition scenarios) and summarized in Table 3.

It is also possible to notice that the acquisition set `plants` shows smaller

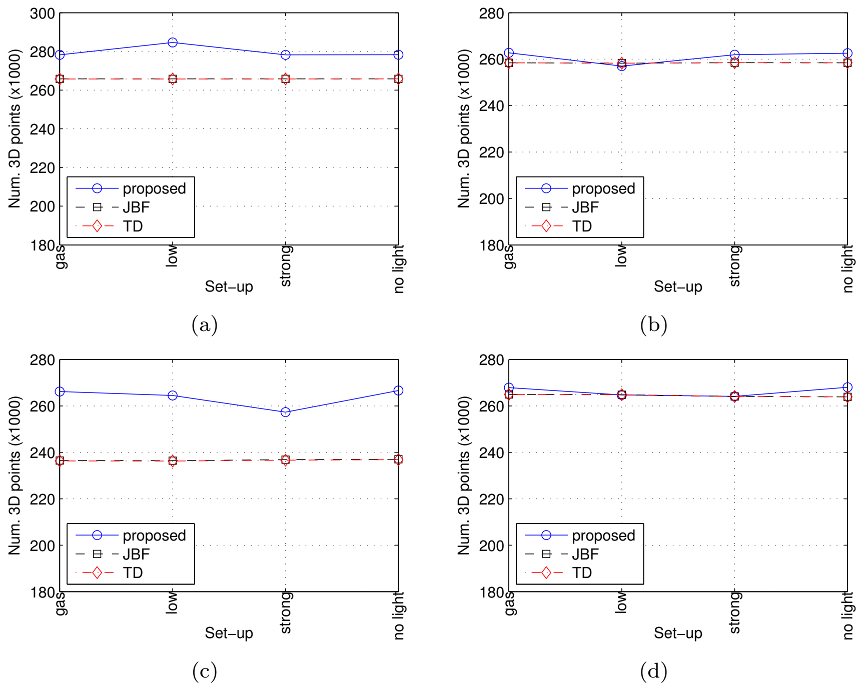


Figure 9: Average number of 3D points for different scenarios under different illumination conditions and noise conditions. a) `bear`, b) `plants`, c) `pillows`, d) `ballbook`.

differences since most of the objects present sharp edges, and therefore, a limited amount of lateral IR radiation produced by the side device if reflected towards the acquiring Kinect. Instead, the objects in `bear` acquisition sets are rounded and highly reflecting. As a matter of fact, it is possible to appreciate a more evident difference in the results.

In order to provide a visual evidence for the adopted algorithms, Fig. 11 shows some details of the processed depth maps taken from different scenarios. It is possible to notice that the object borders of depth maps processed by the proposed solution are sharper and more defined.

In the end, we tested the performance improvements with respect to the solution in [19]. Table 4 reports the average MSE and number of 3D points for the acquisition scenarios `plants` and `pillows`. It is possible to notice that the proposed solution permits obtaining a higher accuracy although the solution in [19] generates denser point clouds thanks to the adoption of segmentation. The segmentation algorithm permits defining interpolation areas more accurately but requires a significant computational effort with respect to the overall calculation. This fact will be evaluated in the following paragraph.

As for the computational complexity, we evaluated the calculation effort of the algorithm for the acquisition set `bear - gas` averaging the execution time for 10 realizations. The average computational time on a Intel® QuadCore i7

Table 2: Average performance for different algorithms.

	Avg. MSE			**Num. 3D points ($\times 10^5$)**		
Dataset	**JBF**	**TD**	**prop.**	**JBF**	**TD**	**prop.**
`gas`	0.207	0.208	**0.191**	2.564	2.564	**2.688**
`low`	0.197	0.195	**0.184**	2.563	2.563	**2.677**
`strong`	0.212	0.212	**0.193**	2.562	2.563	**2.654**
`no light`	0.211	0.211	**0.192**	2.562	2.563	**2.689**

Table 3: Average performances for different algorithms with side device noise.

	Avg. MSE			**Num. 3D points ($\times 10^5$)**		
Dataset	**JBF**	**TD**	**prop.**	**JBF**	**TD**	**prop.**
90	0.220	0.216	**0.206**	2.622	2.622	**2.668**
72	0.216	0.211	**0.200**	2.618	2.618	**2.665**
54	0.227	0.221	**0.210**	2.621	2.621	**2.663**
36	0.231	0.225	**0.215**	2.622	2.622	**2.658**
18	0.242	0.235	**0.222**	2.619	2.619	**2.643**
0	0.250	0.241	**0.228**	2.615	2.615	**2.630**

CPU running at 1.80 GHz with 12 GB RAM was about 58 ms, which allows denoising at approximately 17 frame/s. The computational effort is partitioned as reported in Table 5. It is possible to notice that interpolation is the most time consuming operation, but further optimizations are still possible. With respect to the JBF approach, the computational complexity of the proposed solution is much lower. Moreover, the TD strategy requires more than one depth frame, while our approach can be applied instantaneously.

As regards the solution in [19], the average running time on the same machine for the `bear - gas` data is approximately 136 ms (which allows acquiring and denoising 7 frame per second). Most of the computational effort is dedicated to the segmentation and interpolation of the available samples. Therefore, a real time implementation becomes quite difficult.

6. Conclusion

The paper presents a joint denoising and interpolation approach for the MS Kinect sensor. The algorithm is based on an initial correction of depth values in the sequence, which then will be interpolated in order to fill holes and missing depth values. The proposed solution permits obtaining significant improvements for 3D models acquired under different light conditions. Experimental results also show that the proposed solution permits rejecting the cross-talk noise which can be generated by a second Kinect device operating in the same environment. Future work will be devoted to optimize the computational complexity of the approach and extend the denoising strategy to Kinect v2.0 devices. Moreover, additional improvements can be obtained by integrating the depth information along the time.

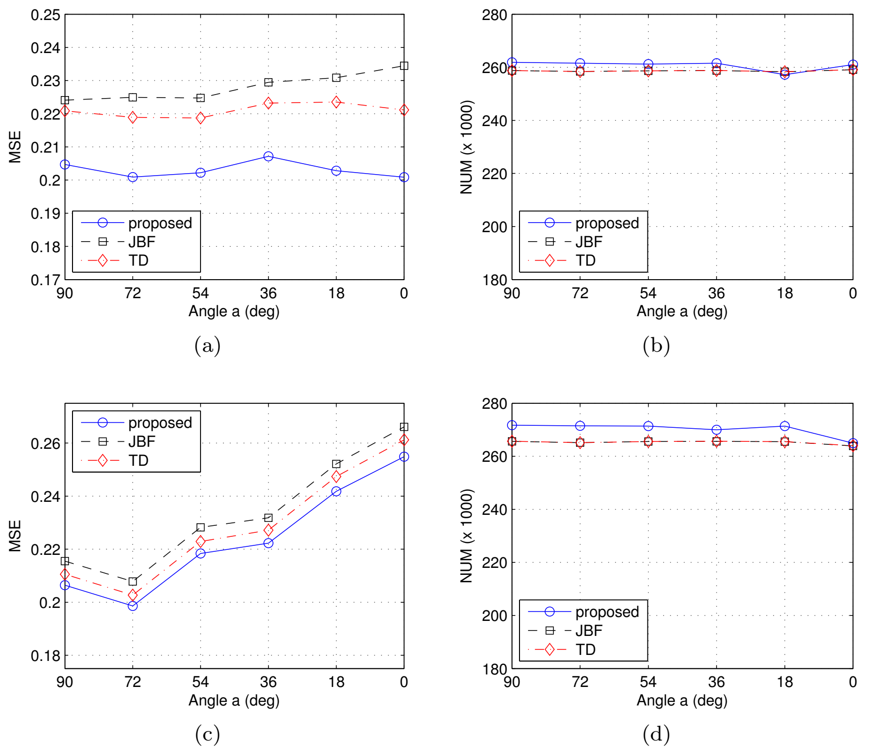


Figure 10: Average MSE and number of 3D points for scenarios `bear` and `plants` with different cross-talk noise levels. The noise level is changed varying the angle a which determines the position of the second Kinect device. a) `plants` MSE, b) `plants` NUM, c) `bear` MSE, d) `bear` NUM.

Acknowledgements

This work has been partially supported by the University of Padova projects 3DCloudVision prot. CPDA142399 and Robotic3D .

[1] S. Gokturk, H. Yalcin, C. Bamji, A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions, in: Proc. of CVPRW 2004, Vol. 3, 2004, p. 35.

[2] IHS iSuppli, The teardown: The kinect for xbox 360, IET Engineering Technology 6 (3) (2011) 94 –95.

[3] J. Hartmann, D. Forouher, M. Litza, J. H. Kluessendorff, E. Maehle, Real-time visual slam using fastslam and the microsoft kinect camera, in: Proc. of ROBOTIK 2012, 2012, pp. 1–6.

[4] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with mi-

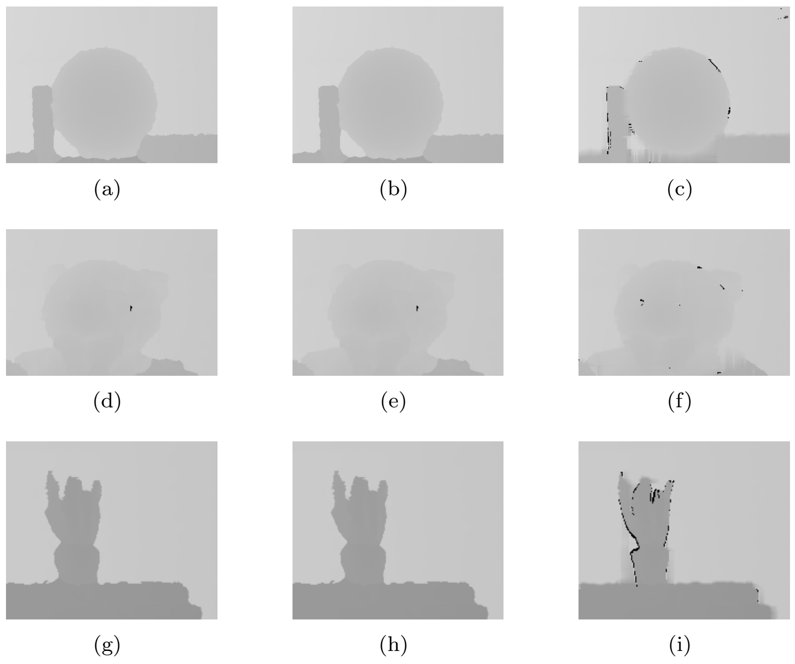


Figure 11: Details from acquisition sets `ballbook gas` (a,b,c), `bear gas` (d,e,f), and `plants gas` (g,h,i). Methods JBF (a,d,g), TD (b,e,h), and proposed (c,f,i).

crosoft kinect sensor: A review, Cybernetics, IEEE Transactions on 43 (5) (2013) 1318–1334.

[5] T. Leyvand, C. Meekhof, Y.-C. Wei, J. Sun, B. Guo, Kinect identity: Technology and experience, Computer 44 (4) (2011) 94 –96.

[6] E. Suma, B. Lange, A. Rizzo, D. Krum, M. Bolas, Faast: The flexible action and articulated skeleton toolkit, in: Proc. of IEEE VR 2011, Singapore, 2011, pp. 247 –248.

[7] A. D. Wilson, Using a depth camera as a touch sensor, in: Proc. of ACM ITS 2010, Saarbrucken, Germany, 2010, pp. 69–72.

[8] W. Garage, Turtlebot overview (Jul. 2015).
URL `http://www.willowgarage.com/turtlebot`

[9] F. Marinello, A. Pezzuolo, F. Gasparini, J. Arvidsson, L. Sartori, Application of the kinect sensor for dynamic soil surface characterization, Precision Agriculture (2015) 1–12.

[10] H.-T. Chang, Y.-W. Li, H.-T. Chen, S.-Y. Feng, T.-T. Chien, A dynamic fitting room based on microsoft kinect and augmented reality technologies, in: M. Kurosu (Ed.), Human-Computer Interaction. Interaction Modalities

Table 4: Performance improvements with respect to strategy in [19] for different algorithms.

	Avg. MSE		**Num. 3D points ($\times 10^5$)**	
Dataset	**prop.**	**[19]**	**prop.**	**[19]**
`plants gas`	0.203	0.209	2.627	2.906
`plants low`	0.202	0.216	2.570	2.864
`plants strong`	0.202	0.208	2.619	2.943
`plants no light`	0.202	0.210	2.625	2.910
`pillows gas`	0.198	0.205	2.662	2.696
`pillows gas`	0.206	0.209	2.645	2.691
`pillows gas`	0.204	0.209	2.573	2.775
`pillows gas`	0.197	0.204	2.666	2.705

Table 5: Profiling of the execution time for the algorithm

Operation	**Time (%)**
Threshold computation	8.67
Edge computation	5.47
Edge mismatch compensation	16.42
Erosion & Interpolation	69.44

and Techniques, Vol. 8007 of Lecture Notes in Computer Science, 2013, pp. 177–185.

[11] A. Wilson, H. Benko, S. Izadi, O. Hilliges, Steerable augmented reality with the beamatron, in: Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12, ACM, New York, NY, USA, 2012, pp. 413–422.

[12] B. R. Jones, H. Benko, E. Ofek, A. D. Wilson, Illumiroom: Peripheral projected illusions for interactive experiences, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13, ACM, New York, NY, USA, 2013, pp. 869–878.

[13] K. Essmaeel, L. Gallo, E. Damiani, G. D. Pietro, A. Dipand, Multiple structured light-based depth sensors for human motion analysis : a review, LNCS ; 7657, Springer, Berlin, 2012, pp. 240 – 247.

[14] Y. Schroder, A. Scholz, K. Berger, K. Ruhl, S. Guthe, M. Magnor, Multiple kinect studies, in: Computer Graphics Technical Report, 2011.

[15] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, D. Kim, Shake'n'sense: Reducing interference for overlapping structured light depth cameras, in: Proc. of the ACM CHI 2012, Austin, Texas, USA, 2012, pp. 1933–1936.

[16] A. Maimone, H. Fuchs, Encumbrance-free telepresence system with real-

time 3d capture and display using commodity depth cameras, in: Proc. of ISMAR 2011, 2011, pp. 137–146.

[17] M. LaMonica, Microsoft's Kinect: A robot's low-cost, secret weapon, CNET (Dec. 6, 2011). URL `http://www.cnet.com/news//microsofts-kinect-a-robots-low-cost-secret-weapon/`

[18] H. J. Min, D. Fehr, N. Papanikolopoulos, A solution with multiple robots and kinect systems to implement the parallel coverage problem, in: Proc. of MED 2012, 2012, pp. 555–560.

[19] S. Milani, G. Calvagno, Joint denoising and interpolation of depth maps for ms kinect sensors., in: Proc. of 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012), Kyoto, Japan, 2012, pp. 797–800.

[20] K. Essmaeel, L. Gallo, E. Damiani, G. De Pietro, A. Dipanda, Comparative evaluation of methods for filtering kinect depth data, Multimedia Tools and Applications (2014) 1–24.

[21] O. Gangwal, B. Djapic, Real-time implementation of depth map post-processing for 3d-tv in dedicated hardware, in: Proc. of ICCE 2010C, Las Vegas, NV, USA, 2010, pp. 173–174.

[22] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: Proc. of ICCV 1998, Bombay, India, 1998, p. 839.

[23] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, ACM Trans. Graph. 22 (3) (2003) 950–953.

[24] B. Huhle, T. Schairer, P. Jenke, W. W. Straíer, Fusion of range and color images for denoising and resolution enhancement with a non-local filter, Comput. Vis. Image Underst. 114 (2010) 1336–1345.

[25] M. Frank, M. Plaue, F. A. Hamprecht, Denoising of continuous-wave time-of-flight depth images using confidence measures, Optical Engineering 48 (7).

[26] Q. Yang, R. Yang, J. Davis, D. Nister, Spatial-depth super resolution for range images, in: Proc. of IEEE CVPR 2007, Vol. 0, Los Alamitos, CA, USA, 2007, pp. 1–8.

[27] A. V. Le, S.-W. Jung, C. S. Won, Directional joint bilateral filter for depth images, Sensors 14 (7) (2014) 11362–11378.

[28] A. Koschan, M. Abidi, Depth map denoising via cdt-based joint bilateral filter, in: L. Shao, J. Han, P. Kohli, Z. Zhang (Eds.), Computer Vision and Machine Learning with RGB-D Sensors, Advances in Computer Vision and Pattern Recognition, Springer International Publishing, 2014, pp. 65–89.

[29] M. Camplani, T. Mantecon, L. Salgado, Depth-color fusion strategy for 3-d scene modeling with kinect, Cybernetics, IEEE Transactions on 43 (6) (2013) 1560–1571.

[30] F. Qi, J. Han, P. Wang, G. Shi, F. Li, Structure guided fusion for depth map inpainting, Pattern Recogn. Lett. 34 (1) (2013) 70–76.

[31] Y. Wang, F. Zhong, Q. Peng, X. Qin, Depth map enhancement based on color and depth consistency, The Visual Computer 30 (10) (2014) 1157–1168.

[32] V. Garro, C. dal Mutto, P. Zanuttigh, G. M. Cortelazzo, A Novel Interpolation Scheme for Range Data with Side Information, in: Proc. of CVMP 2009, London, United Kingdom, 2009, pp. 52 – 60.

[33] S. Milani, D. Ferrario, S. Tubaro, No-reference quality metric for depth maps, in: Image Processing (ICIP), 2013 20th IEEE International Conference on, 2013, pp. 408–412.

[34] E. Ekmekcioglu, M. Mrak, S. Worrall, A. Kondoz, Edge adaptive upsampling of depth map videos for enhanced free-viewpoint video quality, Electronics Letters 45 (7) (2009) 353 –354.

[35] L. Jovanov, A. Pižurica, W. Philips, Fuzzy logic-based approach to wavelet denoising of 3d images produced by time-of-flight cameras, Opt. Express 18 (22) (2010) 22651–22676.

[36] K. Essmaeel, L. Gallo, E. Damiani, G. De Pietro, A. Dipanda, Temporal denoising of kinect depth data, in: Proc. of SITIS 2012, 2012, pp. 47–52.

[37] Z. Zhang, Flexible camera calibration by viewing a plane from unknown orientations, in: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, Vol. 1, 1999, pp. 666–673 vol.1.

[38] C. M. Bishop, Pattern Recognition and Machine Learning, Springer Science+Business Media, LLC, 2006.

[39] S. Milani, G. Calvagno, A depth image coder based on progressive silhouettes, IEEE Signal Process. Lett. 17 (8) (2010) 711 –714.

[40] D. Herrera C, J. Kannala, J. Heikkila, Joint depth and color camera calibration with distortion correction, IEEE Trans. Pattern Anal. Mach. Intell. 34 (10) (2012) 2058–2064.
URL `www.ee.oulu.fi/~dherrera/kinect/`

[41] S. Milani, MS Kinect Dataset (2014).
URL `www.dei.unipd.it/~sim1mil/materiale/kinect_data/`

[42] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, International Journal of Computer Vision 13 (2) (1994) 119–152.