

Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras

Søren Maagaard Olesen · Simon Lyder ·
Dirk Kraft · Norbert Krüger · Jeppe Barsøe Jessen

Received: 15 November 2011 / Accepted: 14 June 2012 / Published online: 6 July 2012
© Springer-Verlag 2012

Abstract In this paper, we present our work on GPU-based real-time extraction of surface patches by means of Kinect cameras. This paper makes four contributions: (1) we derive an uncertainty model for pixel-wise depth reconstruction on Kinect cameras; (2) we implement a real-time algorithm for surface patch (here called ‘textlet’) extraction based on Kinect depth data on a GPU. For that we compare and evaluate different implementation alternatives. (3) Based on (1) we derive and implement an appropriate uncertainty model for textlets which is also computed in real-time. (4) We investigate and quantify the effect of interferences on the depth extraction process when using multiple Kinect cameras. By these contributions we present insights into the processing of depth and how to achieve higher precision reconstructions by means of Kinect cameras as well as extend their use for higher level visual processing. The introduced algorithms are available in the C++ vision library CoViS.

Keyword Surface patch · Kinect · Stereo · Reconstruction · Uncertainty

1 Introduction

The Microsoft Kinect¹ sensor is an RGB-D sensor which provides RGB and depth information (see Fig. 1). It is developed by the Israeli company PrimeSense and was originally intended to be used with the Microsoft Xbox 360

gaming console. However, very soon after the launch of the Kinect in November 2010 it became possible to gain access to the Kinect functionality from a PC.² The possibility of using the Kinect with normal PCs made the Computer Vision community highly interested because of its impressive performance on depth extraction and relatively low cost. Since the Kinect actively emits texture in the infrared domain (see Fig. 1b, c), matching is possible on homogeneous surfaces. This solves a major pitfall of classical stereo vision approaches, and allows a more complete 3D representation of the scene.

Meanwhile, a large amount of work has been done utilizing the Kinect camera to its full extent, e.g., both PCL³ (see [25]) and ROS⁴ (see [23]), two major contributors in the 3D Computer Vision and Robotics community, do support the Kinect device. Recent application areas of the Kinect include range flow estimation [5], human detection using contours and 3D surface models [29], 3D object detection [26], 3D modeling of indoor environments [7] and dense surface mapping and tracking of scenes [19].

Recently, a significant number of papers have addressed the calibration of Kinect cameras and their use for the processing of depth descriptors. ROS and [16] contributed general knowledge about Kinect cameras as well as calibration procedures required. Calibration and the geometric properties of depth measurements by the Kinect are also investigated in [11, 28] (for a list of general parameters derived from the Kinect calibration process see, e.g., [28]). In [2], a Kinect calibration scheme is proposed which uses a checkerboard with special materials that either reflects or

S. M. Olesen · S. Lyder · D. Kraft · N. Krüger ·
J. B. Jessen (✉)
The Mærsk Mc-Kinney Møller Institute, University of Southern
Denmark, Odense, Denmark
e-mail: jeje@mmmi.sdu.dk

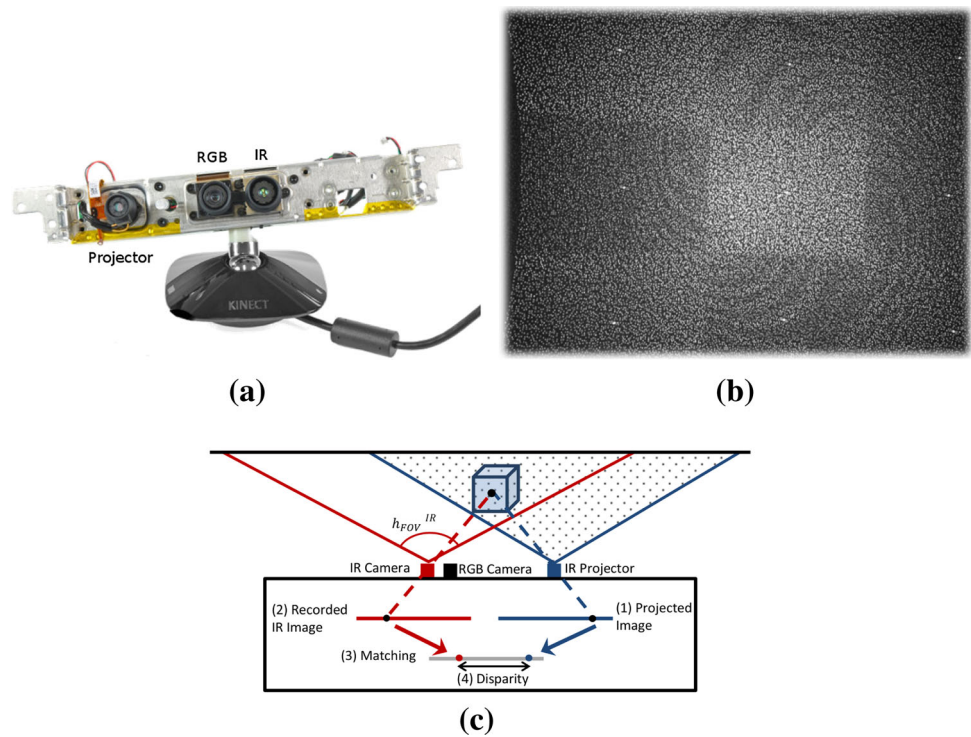
¹ See <http://www.xbox.com/kinect>.

² See e.g., the LibFreenect driver <http://openkinect.org>.

³ Point Cloud Library, <http://pointclouds.org>.

⁴ Robot Operating System, <http://www.ros.org>.

Fig. 1 **a** Kinect device without cover (image from <http://www.ifixit.com/>) and **b** recording of projected IR pattern. **c** Triangulation with the Kinect: (1) the known IR pattern is projected on the scene. (2) The IR image is recorded. (3) Matching of (1) and (2) is performed to achieve (4) a disparity value, which is transformed to a depth value



deflects the infrared light, to make the pattern visible in both the depth and RGB image. By that a normal stereo calibration can be done using a binary version of the depth image and the RGB image as left and right input. They also systematically evaluate the degradation of the matching and reconstruction under concurrent use of multiple Kinects and show that blocking the infrared emitters, enabling only one Kinect pattern at a single point in time will actually degrade the results significantly. This indicates that the Kinect depth processing relies on temporal consistency in the IR image and are disturbed by the IR patterns of the other Kinects depending on their relative position. Calibration of time-of-flight (ToF) cameras and sensor fusion systems have been studied extensively in, e.g., [13, 27]. This is relevant here because many concepts of the calibration can be transferred from sensors that combine ToF and an RGB camera to Kinect cameras. Both sensors can provide a dense depth image with the inherent challenges of aligning that to a color image. Another known problem of Kinect cameras is the problem of dealing with depth discontinuities and surfaces with poor reflectivity, which have also been addressed in studies of ToF [24].

A primary goal of the Kinect is to be a low-cost device for ‘natural interaction’ (according to PrimeSense) rather than a precision tool in computer vision and robotics research. OpenNI⁵ has been used to interface with the

Kinect sensor. However, the OpenNI driver handles the modeling of the reconstruction process and the calibration of the involved parameters in a suboptimal way. In order to achieve more precise reconstructions, the pinhole model used for back-projection in the OpenNI driver needs to be extended and some imprecisions are identified and corrected.

In this paper, we present a real-time algorithm for the computation of local surface patches (in the following called ‘textlets’, see Fig. 2c, d) with Kinect cameras. A textlet shares many properties with the patchlet feature, presented in [18], and the surfel feature presented in [3] and used in [7, 20].

In the context of deriving an uncertainty model for textlets, we give both a qualitative and quantitative estimate of the uncertainty distribution of point-wise depth reconstruction with the Kinect camera. As a first contribution of the paper, we provide a quantification of the uncertainty of point reconstructions by deriving a polynomial approximation of the 3D point noise, and by means of Monte Carlo simulation we can use this to compute textlet uncertainties.

A textlet is a local surface descriptor consisting of a 3D center point \mathbf{P} and a surface normal \mathbf{N} , both with associated uncertainties (see Fig. 2c). In addition, information about size and appearance are associated. Textlets allow for a semantically richer and more condensed representation at smooth 3D surfaces than point clouds [21]. The textlet descriptor is part of the Early Cognitive Vision (ECV) system [22]. The ECV system is a hierarchical system

⁵ See <http://www.openni.org/>.

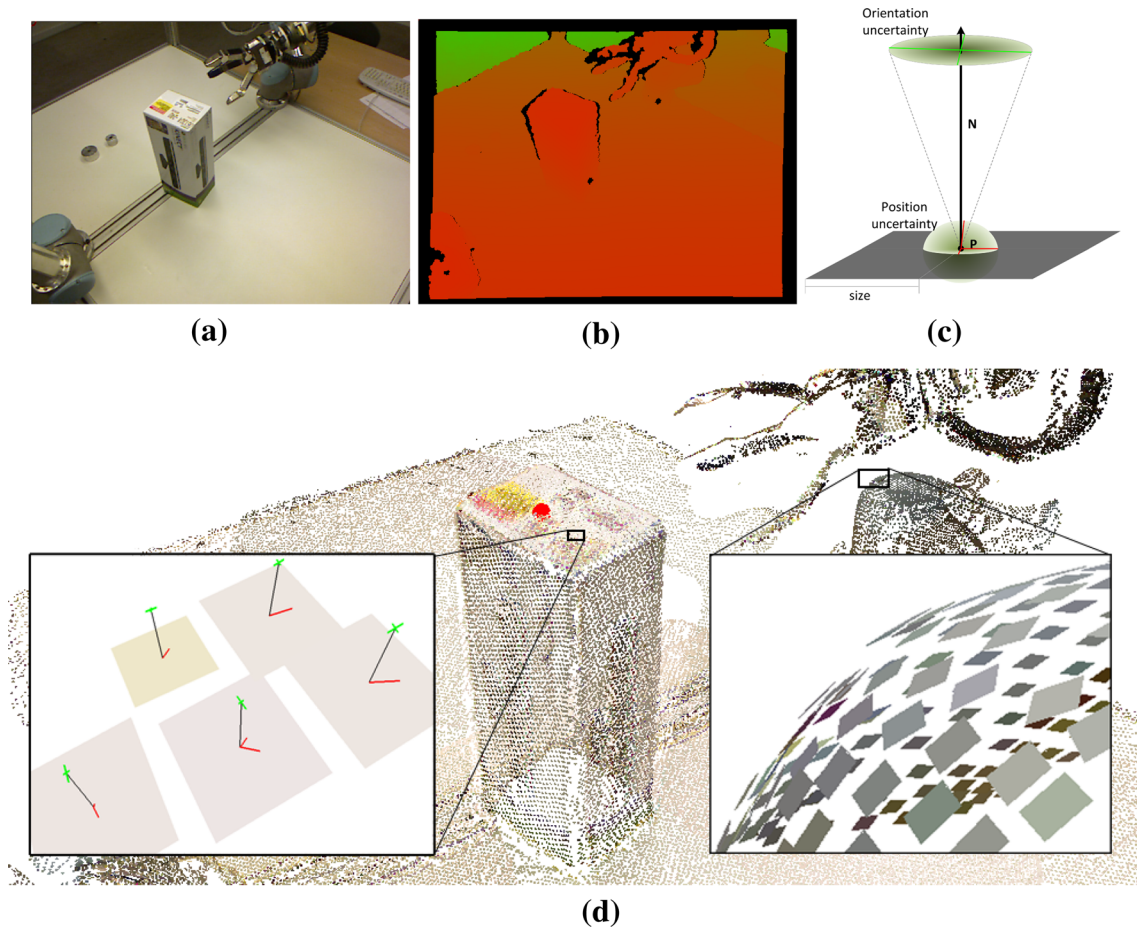


Fig. 2 **a** RGB scene image obtained by Kinect and **b** the corresponding depth map (colored for visualization purposes) from the OpenNI driver. **c** Textlet representation and parameterization. *Red* and

green lines indicate principal axes of position and orientation uncertainty, respectively. **d** Example of extracted textlets, with details of a flat surface (including uncertainty visualization) and curvature

consisting of visual descriptors of different abstraction level (such as textlets, surflets and surfaces) and has been used in particular in robotic applications (see, e.g., [21]). Figure 3 shows a robot platform realizing an intelligent work-cell with two arms which is also used for quantifications in this paper. The workspace is observed by three pairs of Kinect and stereo cameras (see sub-figure on bottom left in Fig. 3).

Textlets as described here can be extracted both from Kinect cameras as well as stereo cameras. Since entities of higher abstraction level in the ECV system are dependent on the textlets, the derivation of an appropriate uncertainty model of textlets as well as real-time capabilities are of importance. Real-time processing is achieved by means of a Graphics Processing Unit (GPU) which is now widely used in computer vision.

We use GPUs to extract textlets in real-time from the 3D information provided by the Kinect camera. Our approach can be used also for any other 3D sensing device—such as a standard stereo cameras—provided an appropriate noise model exists. We compare three different methods to

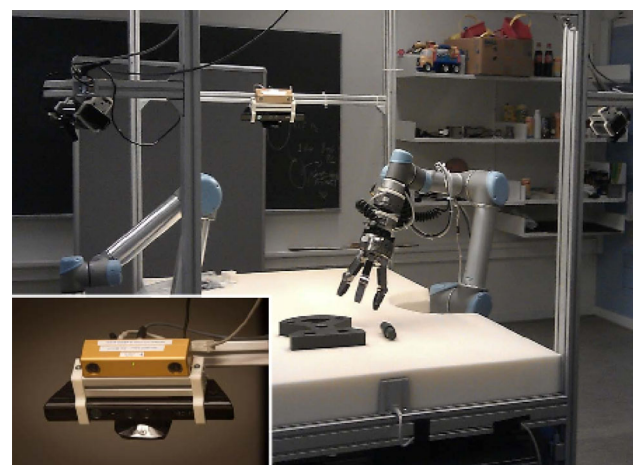


Fig. 3 Robotic setup with multiple stereo and Kinect cameras

compute the textlet position \mathbf{P} and the associated surface normal \mathbf{N} . By this, we can indicate two methods with similar reconstruction quality from which one is more

suitable to be transferred to a GPU due to its non-iterative nature. As a second contribution, we achieve a processing speed for texlet computation (including an associated uncertainty based on the uncertainty model of point reconstruction) of up to 30 Hz, thus utilizing the full frame rate of the Kinect. As a reference, the same algorithm as described here applied to 3D point clouds extracted from stereo camera input (with approx. double resolution compared to the Kinect camera) achieves a processing speed of up to 10 Hz. Based on the pixel-wise uncertainty model and the texlet extraction mechanism, we—as a third contribution—derive an uncertainty model for texlets (also computed in real-time) which is crucial for higher level processes such as temporal accumulation (see [9]) and optimal parameter estimation (see, e.g., [1]).

The fact that depth reconstruction by means of Kinect cameras, is based on a projection of an infrared pattern, results in interferences when multiple Kinect cameras are used at the same time, since the projected patterns overlap. This potentially reduces the matching quality, and as a consequence also the reconstruction quality. In this context, as a fourth contribution we quantify—in terms of known geometric relations of a test object—the degradation of the quality of the texlet reconstruction on an example scene when using one, two and three Kinect cameras simultaneously. Thus, we are able to investigate the effect of the interference of multiple Kinect cameras for depth extraction.

The paper is structured as following: In Sect. 2, we describe the geometry and reconstruction with Kinect cameras including the indicated simplifications in the available drivers and the way we correct these. In this context, we also perform a modeling of the uncertainty of point reconstruction. In Sect. 3, we describe our work on texlet reconstruction and its associated uncertainty as well as the real-time implementation on GPUs. In Sect. 4, we then discuss and quantify the effect of the interference when using multiple Kinect sensors for scene reconstruction.

2 Reconstruction with Kinect cameras

Multiple drivers are available for using the Microsoft Kinect. The first major initiative resulted in the unofficial Open Source driver LibFreenect. PrimeSense later on released their own driver and initiated a new Open Source framework called OpenNI, which easily enables the functionality of their RGB-D sensors including the one in the Kinect. These drivers are still under constant development. LibFreenect has the advantage of providing support for the hardware not related to the PrimeSense chip (e.g., the

motors and LEDs of the Kinect), whereas OpenNI shows a more complete interface to the features of the PrimeSense chip. We prefer the latter to have easy access to both RGB image, IR image and depth image, hence this is considered to be the driver of choice. In the following section, we deal with the reconstruction process realized in Kinect cameras and in Sect. 2.2 we address the point-wise uncertainty computation.

2.1 Reconstruction of 3D points

We first describe the 3D reconstruction process realized in the OpenNI driver in the following section and then describe our approach to correct inaccuracies in Sect. 2.1.2.

2.1.1 Reconstruction of 3D points using the OpenNI driver

The Kinect consists of two sensors and one emitter in order to provide color and depth information. Figure 1a shows the different components. It possesses a standard RGB camera, which is horizontally aligned to an IR camera with a baseline of approximately 2.5 cm. It has an IR projector which emits a pattern of speckles as shown in Fig. 1b. The IR projector and IR camera have a baseline of approximately 7.5 cm.

Since the projected IR pattern is known to the Kinect, this serves as a virtual second image. The recorded IR image will contain a distorted version of the projected image allowing for calculations of disparities between the image pairs by means of matching. The exact process of matching and the extraction of depth information through triangulation (see Fig. 1c) is undisclosed and protected by a PrimeSense patent. Reverse engineering efforts have revealed some of the details though (see [16, 28]).

When using the OpenNI driver, the depth map (Fig. 2b) can be acquired directly. The depth map is a per pixel map containing a 16-bit depth value z_{depth} (with 11-bit resolution) at each pixel point $(x_{\text{pixel}}, y_{\text{pixel}})$ of the IR camera. If coordinates are needed in 3D space and not as image coordinates, they can be transformed by the driver to real-world points relative to the IR camera coordinate system. OpenNI uses a simplified pinhole model (assuming undistorted images and that the principal point is at the image center) when doing the conversion from $(x_{\text{pixel}}, y_{\text{pixel}}, z_{\text{depth}})$ to (x, y, z) coordinates as described in Eqs. (1–3).

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_{\text{norm}} \cdot \tan\left(\frac{f_{\text{IR}}}{2}\right) \\ y_{\text{norm}} \cdot \tan\left(\frac{f_{\text{IR}}}{2}\right) \\ 1 \end{pmatrix} \cdot z_{\text{depth}} \quad (1)$$

Where:

$$x_{\text{norm}} = 2 \times \left(\frac{x_{\text{pixel}}}{r_x} - 0.5 \right) \tag{2}$$

$$y_{\text{norm}} = 2 \times \left(0.5 - \frac{y_{\text{pixel}}}{r_y} \right) \tag{3}$$

Here r_x and r_y are the resolution of the depth map in x and y , respectively. $h_{\text{FOV}}^{\text{IR}}$ and $v_{\text{FOV}}^{\text{IR}}$ are the horizontal and vertical field of views of the IR camera, which are stored in the Kinect devices. $h_{\text{FOV}}^{\text{IR}}$ is displayed in Fig. 1c.

2.1.2 Improving reconstruction quality

Equation (1) assumes that the principal point of the IR camera is in the center of the image. This simplification leads to some inaccuracy in the reconstruction of real-world points derived from the depth map.

The driver also provides the functionality to align the depth map and the RGB image, a process called registration, giving the ability to associate a color to each real-world point. Registration is a well-known concept from the calibration of, e.g., ToF cameras, and is studied in, e.g., [12].

When aligning the depth map obtained from the IR camera with the RGB image using OpenNI, the depth map coordinates $(x_{\text{pixel}}, y_{\text{pixel}}, z_{\text{depth}})^T$ are first projected into the RGB camera coordinate system. This is necessary since the distance between the RGB camera and the IR sensor gives slightly different views. The FOV value is used, as this can be read from the sensor device. One error we encountered arose when these new re-projected depth map coordinates were back-projected into real-world points. Here, the field of view of the IR camera $h_{\text{FOV}}^{\text{IR}}$ and $v_{\text{FOV}}^{\text{IR}}$ from Eq. (1) should be replaced with the FOV of the RGB camera ($h_{\text{FOV}}^{\text{RGB}}$ and $v_{\text{FOV}}^{\text{RGB}}$).

The identified problems can be solved in several ways. One method is to simply use the correct FOV when back-projecting the aligned points, which however is still a simplification.

When we are anyway calibrating the Kinect, the precision of reconstruction can be further increased by doing individual monocular calibration of the RGB camera. The depth data, once registered to the RGB camera, are in this viewpoint, and the reconstruction must thus be done with the RGB camera intrinsics. Having the calibration matrix of the RGB camera, the vector representing the back-projection ray going from the focal point through the image point $(x_{\text{pixel}}, y_{\text{pixel}})$, can be calculated (see, e.g., [6]). Since the depth is known, the originating 3D point can be reconstructed. Calibration of the RGB camera was done as discussed in [28].

2.2 Uncertainty model for point reconstructions with Kinect cameras

In the ECV system [21, 22] the texlets represent only the initial state of a processing hierarchy. For the computation of entities at higher levels, it is crucial to propagate the uncertainty associated with the entities at lower levels of the hierarchy to higher levels for spatial-temporal disambiguation (see [9]), as well as numerical optimization purposes when estimating properties of higher level entities (see [1]).

Since the details of the matching process as well as the reconstruction process are not completely known, it is difficult to derive an analytical model for the uncertainty of depth reconstruction with Kinect cameras and we rely on a Gaussian distribution model estimated experimentally for our uncertainty computations. A series of experiments were conducted to estimate the varying noise at all positions in the depth image returned from a Kinect sensor. The Kinect was pointed towards a flat wall and for a number of specified distances and orientations 100 depth maps were computed. From these depth recordings a standard deviation (SD) was calculated for each pixel position. Figure 4 shows the SD images obtained at 0° at 2 different depths. The SDs computed are for visual purposes normalized to gray-scale values (with white as zero deviation and black as the maximum deviation recorded) and printed for each pixel position.

From the recordings it is clearly visible how the SD increases with distance. Furthermore, it seems that there is a radial dependency, where the noise increases with larger distance from the center of the depth map. This is expected since the reconstruction process is based on the very same geometric relations as a normal stereo setup. It is, however, also noticeable that straight vertical lines appear in the images in Fig. 4a, b. These vertical artifacts are also observable in the raw depth map. The depth map seems to be divided into vertical slices. Between two such slices there seems to be a static depth offset. Since the size and appearance of these slices are temporally dependent they also appear in the SD images, generating larger variations in the transition areas. The vertical slices can also be seen in Fig. 4c. Here, the Kinect is again pointed against a flat wall and all valid points are fitted to a plane. The errors in each point are then depicted with green representing points in front of the plane and red representing points behind.

It has not been possible to determine a deterministic behavior of these lines as they seem to show up at different, seemingly random, places (see the difference in Fig. 4a, b) and are temporally dependent. The static difference between two slices, however, seems to be at the magnitude

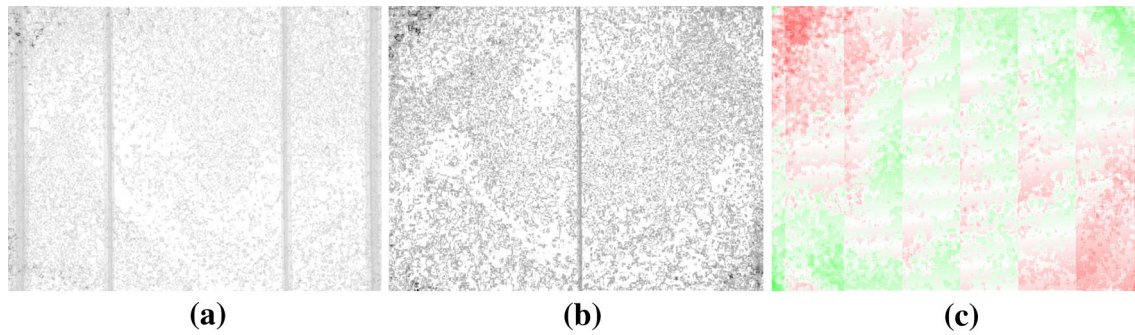


Fig. 4 **a** and **b** Recorded image of SD. *Black* means higher. **a** Distance between camera and wall 1.5 m. **b** Distance between camera and wall 2.5 m. **c** Residuals of plane fitting against flat wall

of the discretization level. These lines are not accounted for in the final uncertainty model for the Kinect since their appearance is not predictable. For a further investigation in the residual errors see [28], where a learning process is utilized to learn tendencies, and correct some of the static error present.

Since the reconstruction process on the Kinect cameras is comparable to a standard stereo reconstruction (see Sect. 2.1), an uncertainty model $n = f(r, d)$ which models the variance of Gaussian noise depending on the radial distance r from the center and the depth d were chosen. We made experiments for fitting the function f using different polynomial degrees which shows that a second order polynomial approximation is optimal.

Altogether six sets of recordings were made at different depths, starting from around 70 cm and up to approx. 175 cm. It was not possible to include sets at larger distances, i.e., utilize the full range of the Kinect, since this requires a very large planar surface. Each set includes measurements where the Kinect were placed in seven different orientation configurations.

The first configuration is frontal parallel to the wall, referred to as 0° . In another two configurations the Kinect is tilted 15° relative to the wall, in either the horizontal or vertical direction. The next two configurations are similar, but the angle is increased to 30° . In the final two configurations the Kinect was tilted both horizontally and

vertically so the combined angle relative to the wall would be either 15° or 30° . The polynomial fit of some of the data sets sorted by orientation configurations (0° , 30° horizontal and vertical and 30° diagonal) are shown in Fig. 5. Interestingly, we see that the 3D reconstruction quality does not significantly depend on the orientation of the wall, within the tested orientations, indicating that the angle is not required as a third parameter of the noise model. This might be connected to the (unfortunately undisclosed) matching function used in the Kinect camera, which possibly has an inbuilt invariance towards changes in the infrared pattern caused by slanted surfaces in the scene. Figure 6 shows the fitted uncertainty model based on all orientation measurements. The parametrized model is as follows:

$$f(r, d) = p_{00} + p_{10} \times r + p_{01} \times d + p_{20} \times r^2 + p_{11} \times r \times d + p_{02} \times d^2 \quad (4)$$

where

$$p_{00} = 2.344, \quad p_{10} = -1.202e-2, \quad p_{01} = -2.734e-3, \\ p_{20} = 1.818e-5, \quad p_{11} = 6.516e-6, \quad p_{02} = 1.233e-6.$$

Figure 7 shows the depth quantization as a function of distance, which was measured by recording a number of Kinect images of a frontal parallel wall. The smallest

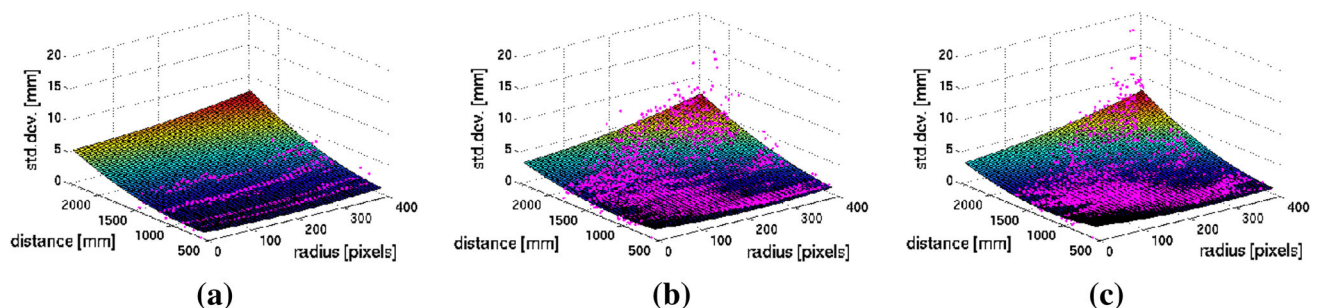


Fig. 5 Polynomial fittings using data acquired with **a** 0° measurements, **b** 30° measurements in either horizontal or vertical direction, **c** 30° diagonal (combined vertical and horizontal) measurements

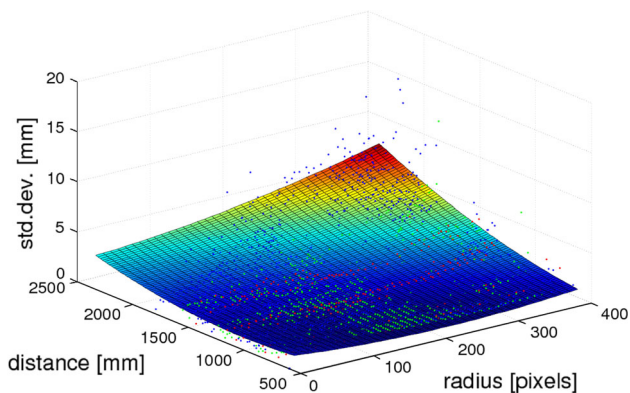


Fig. 6 Uncertainty model using two second degree polynomials to fit radius in pixels (x -axis) and distance in mm (y -axis). z -axis is the SD of the uncertainty in mm. Sub-sampled original data are marked with dots (red 0° , green 15° and blue 30°)

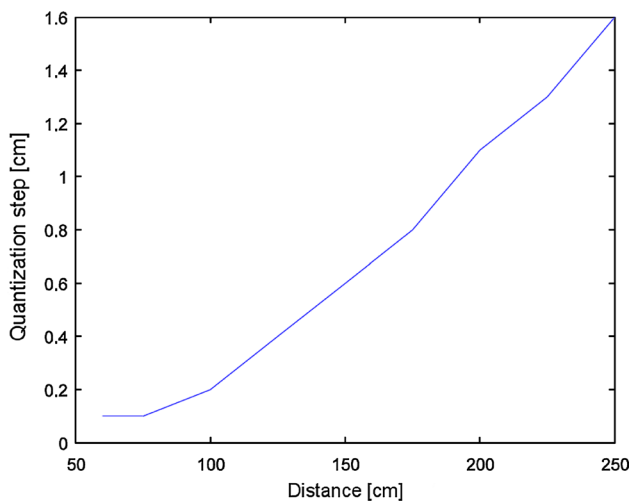


Fig. 7 Depth quantization as a function of distance

change of depth value of a pixel at a given distance are regarded as the quantization step. Comparing Figs. 6 and 7, we conclude that a significant part of the measured noise are in the vicinity of the discretization step.

3 Textlet extraction with Kinect cameras

The textlet feature consists of appearance and geometry information and is extracted from a hexagonal grid of 2D locations in the image (see [22]). The hexagonal grid is chosen since it, contrary to a square grid, ensures a more uniform distance to all neighboring textlets [17]. Appearance is not of focus here and can be expressed in different ways (e.g., by using a color histogram computed from the RGB image or even simpler a mean color value, see e.g., [21]). The geometric information include the surface patch

parameters \mathbf{P} and \mathbf{N} , which is the 5D pose, represented here by a point in 3D space, and a normal vector of the plane, respectively. For each parameter we associate an uncertainty estimate represented using covariance matrices $\Sigma_{\mathbf{P}} \in \mathbb{R}^3 \times \mathbb{R}^3$ for the position and $\Sigma_{\mathbf{N}} \in \mathbb{R}^2 \times \mathbb{R}^2$ for the angular equivalent representation of the orientation vector. The textlet parameterization is shown in Fig. 2c.

To extract the textlet geometry from 3D point data, usually total least squares methods are used (see e.g., [18]) to compute the surface patch parameters. Such methods minimize the orthogonal distance between the plane and every point associated with the patch. This least square optimization is often computed by SVD or other iterative algorithms. Unfortunately, such iterative approaches do not map very well to GPUs except for very large design matrices.⁶ Two non-iterative methods—in the following called ‘normal vector averaging’ [15] and ‘PCA’ [8]—are compared in the following section, with an SVD approach as a reference. In Sect. 3.2, we describe our GPU implementation.

3.1 Comparison of methods for least square fitting of planes

The averaging method [15] is a very lightweight implementation. It uses a center point and the six surrounding neighbors in the hexagonal grid. The normal vectors of the six patches, created by two adjacent neighbor points and the center point, are found and the average is calculated and used as the orientation \mathbf{n} .

$$\mathbf{n} = \frac{1}{n} \sum_{i=0}^{n-1} (p_i - p_c) \times (p_{i+1} - p_c) \tag{5}$$

where p_c is the center point, p_0 to p_{n-1} are the neighboring points in a counter clockwise direction around p_c , with the last point p_n wrapping around and being equal to p_0 . This vector combined with the center point constitutes the textlet orientation \mathbf{N} and position \mathbf{P} (see Fig. 2c). Note that this approach is only suitable for small neighborhood sizes.

The PCA method minimizes the variances of the points instead of the orthogonal distance. All 3D points within a certain neighborhood of the center point are used, and the covariance matrix is calculated for those. Since the covariance matrix is symmetric and positive semi-definite, the eigenvalue decomposition gives real and non-negative Eigenvalues. The Eigenvalue decomposition of a matrix A can be written as

$$A = V \Sigma V^T \tag{6}$$

where Σ contains the Eigenvalues and V contains the Eigenvectors of A . The covariance matrix represents the

⁶ see e.g., CULA tools (<http://culatools.com>).

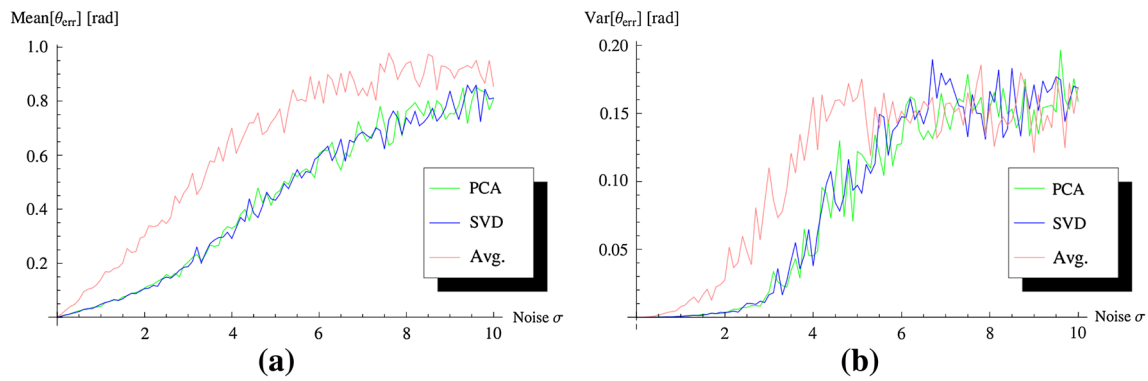


Fig. 8 Mean angle error and variance of unit normal vectors with varying noise

variance ellipsoid spanned by the points. The Eigenvectors are the principal components of this ellipsoid and the Eigenvalues represent the variances in these directions. Assuming that the noise perpendicular to the true plane is smaller than the in-plane variance (which should be determined by the neighborhood size), the Eigenvector corresponding to the smallest Eigenvalue is the normal vector of the plane patch. The Eigenvalues of a matrix A can be found by solving

$$\det(A - \sigma I) = 0 \quad (7)$$

with σ being the variable (Eigenvalues), and I being the identity matrix. Since the points are in \mathbb{R}^3 , Eq. (7) can be solved as a cubic function, and since A was symmetric and positive semi-definite Eq. (7) gives 3 non-negative solutions⁷ for σ . This reduces the computation complexity. The textlet orientation \mathbf{N} is thus the Eigenvector corresponding to the smallest Eigenvalue. During the covariance calculation the average of the neighborhood points are already calculated. This is more suitable to be used as the textlet position \mathbf{P} than just the center point, as it represents the point's center of gravity.⁸ As the average is already calculated, this choice does not impose extra computational complexity on the algorithm.

An emulation of the two algorithms, with SVD as a reference has been done in order to investigate the stability of the calculated orientation \mathbf{N} , when exposed to noise on the input points. The mean and variance of the angular difference between ground truth plane normal and estimated plane normal over 100 runs for increasing added noise can be seen in Fig. 8a, b.

It is clear that the non-iterative PCA method can compete with regular SVD when calculating the normal vector. The very lightweight normal vector averaging is too

⁷ Some of the solutions might be coinciding or zero.

⁸ Some points in a neighborhood might be discarded as a result of outlier rejection.

Table 1 Performance parameters of the GPU implementation when varying grid spacing or neighborhood radius

Variable parameter	Total time	Texlets created	Inliers (%)
Texlet spacing			
2 px	288.7 ms	3.5 Hz	3,8321
6 px	89.8 ms	11 Hz	9,814
8 px	33.1 ms	30 Hz	2,430
Neighborhood radius			
4 px	68.1 ms	15 Hz	10,715
6 px	89.8 ms	11 Hz	9,814
8 px	127.7 ms	8 Hz	9,060

influenced by even small amounts of noise to be used in our context.

3.2 GPU implementation

In order to acquire the textlets in real-time, the CUDA platform associated with NVIDIA graphics cards has been utilized. The data independence of the computation of each textlet fits very well to the SIMT (Single Instruction Multiple Threads) architecture used in CUDA. The calculation is done in three steps as outlined below. Here, we mention a few parameters that have an impact on the performance (see also Table 1). First, the grid spacing between textlets is relevant, since with smaller spaces more textlets are computed. Each textlet is then calculated at the chosen location using a surrounding neighborhood of image points. The second parameter is thus a radius that defines the size of the textlet neighborhood. This determines the number of points used in the computation of each textlet. This is further discussed after the description of the textlet extraction process based on RGB-D data from the Kinect.

Step 1 Transformation to real-world coordinates The depth map coordinates acquired from the Kinect (aligned to the RGB image) is transformed into real-world coordinates, using intrinsics and distortion coefficients for the RGB camera to obtain a back-projection ray, and intersect this at the known depth. If multiple Kinects are used, the data sets are aligned to a common reference coordinate system. This can be done pixel-wise, and thus, in this step, one pixel calculation maps to one thread on the CUDA GPU.

Step 2 Removing outliers from texlet neighborhoods Before the actual reconstruction is done, RANSAC [4] is used to remove any of the 3D points which are classified as outliers. This is done locally using the 3D point associated with each of the pixels in the neighborhood; we iteratively choose three points randomly as shown in pseudo code in Algorithm 1. Each set of three points span a plane, and the outlier criterion for the rest of the points is defined as the distance between a point and the plane. If the distance exceeds a threshold value, the point is considered as an outlier. To achieve good performance when mapping the RANSAC algorithm to the GPU architecture, we choose a finite number of iterations and omit a step of refitting the plane to the current set of inlier points. For GPU execution, the responsibility of each texlet is divided amongst threads. This means that a thread will be responsible for the processing of the RANSAC algorithm for one texlet neighborhood.

Algorithm 1 RANSAC — GPU optimized approach

```

1: for all texlets  $t$  do
2:   repeat
3:     draw random set of 3 points, construct vectors  $v_1$  and  $v_2$ 
4:     calculate plane  $P$  with normal  $N = v_1 \times v_2$ 
5:     for all points  $p$  in neighborhood do
6:       calculate distance  $d$  from point  $p$  to plane  $P$ 
7:       if  $d < \text{minDistance}$  then
8:         add  $p$  to  $\text{inlierPoints}$ 
9:         add  $d^2$  to  $\text{totalError}$ 
10:      end if
11:    end for
12:    if  $\text{totalError} < \text{minTotalError}$  and
        $\text{size}(\text{inlierPoints}) > \text{minNumberOfInliers}$  then
13:      Save  $\text{inlierPoints}$  as current best set
14:       $\text{minTotalError} = \text{totalError}$ 
15:    end if
16:  until  $\text{numberOfIterations}$ 
17: end for
    
```

Step 3 Calculating texlets from neighborhood points The orientation \mathbf{N} of the texlet is found by applying PCA on the 3D points still remaining after applying RANSAC in each texlet neighborhood. The position \mathbf{P} is set to the average of these remaining points. The uncertainties associated to each texlet’s orientation and position are calculated by Monte Carlo simulation [14], which is a computationally heavy method, compared to an analytical derivation. Monte Carlo on the other hand, gives us the advantage of easy

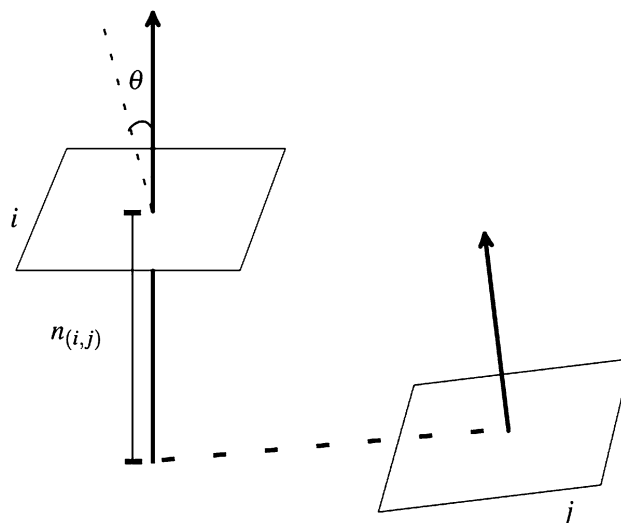


Fig. 9 Texlet relation $(n_{(i,j)}, \theta)$ (normal distance and angle) between texlet i and j

adaptation to other feature types or input data where we want to propagate input uncertainties that might be given in another space, e.g., 2D points of stereo images.⁹ This is the case in our setup shown in Fig. 3, where the amount of input data makes a fast GPU implementation beneficial. On the GPU Monte Carlo is preferred over, e.g., SVD for propagating uncertainties, because of the complexity of the algorithms and how they map to the architecture.

To compute the uncertainties of the texlet parameters, the texlet is created multiple times, each time with noise from the uncertainty model derived in Sect. 2.2 added to each 3D point. The uncertainty representation for the texlet is the covariance matrices $\Sigma_P \in \mathbb{R}^3 \times \mathbb{R}^3$ and $\Sigma_N \in \mathbb{R}^2 \times \mathbb{R}^2$ calculated from this Monte Carlo set. They are visualized in Fig. 2c, and the left highlight in Fig. 2d, using red lines and green lines for the principal axes of the Gaussian distribution model, for position and orientation, respectively.

The spacing of the hexagonal grid (i.e., the distance between the texlet center pixels) can be adjusted given the desired resolution or performance demands. The impact of spacing on the performance and number of created texlets can be seen in Table 1. The effect of changing the neighborhood size is also shown. All the data are obtained using 20 RANSAC iterations and 50 Monte Carlo simulations which has been found experimentally to be adequate. The data with varying spacing (rows 1–3) are obtained with a neighborhood radius of 6 pixels, and the test of neighborhood radius (rows 4–6) are obtained with a spacing of 6 pixels.

⁹ This is similar to the Unscented Transform method [10].

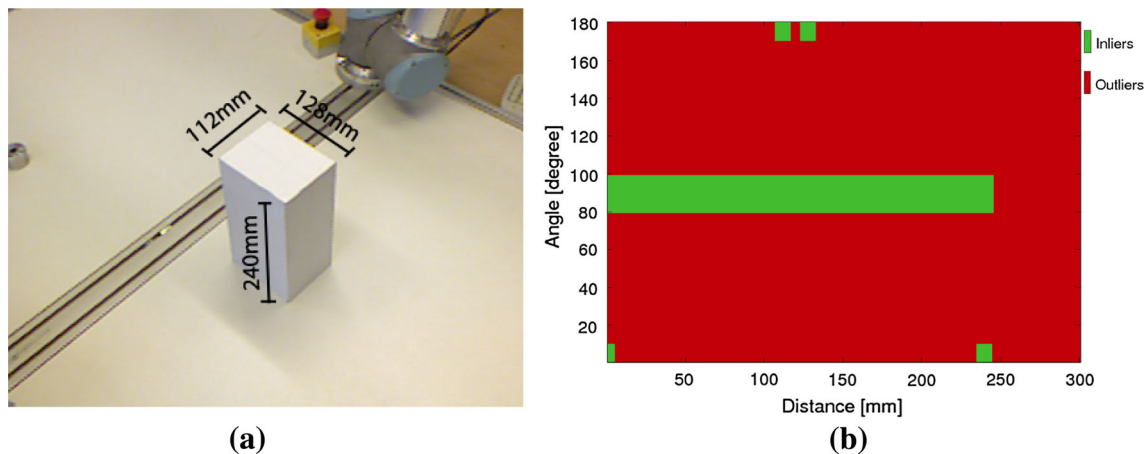


Fig. 10 **a** Known object used in test setup. **b** Bounding boxes used to classify inliers

Table 2 Textlet relation constraints in test setup

Relation	Angle (°)	Distance (mm)
Top of box vs. ground	0	240
Opposite side 1	180	128
Opposite side 2	180	112
In same plane	0	0
From orthogonal planes	90	0–240

4 Quantification of textlet reconstruction with Kinect cameras

As a final step we want to give a qualitative and quantitative evaluation of the reconstruction precision of textlets in a well defined setup. In addition, we want to measure the effects of interference when using multiple (up to three) Kinect cameras. As already mentioned in the introduction, the fact that the Kinect camera is an active sensor which emits light in the infrared domain might affect the matching quality when multiple projection patterns overlay. Since multi-view reconstruction is a common method in computer vision (e.g., to deal with occlusions), it is of interest to measure these effects of interference.

In order to be able to evaluate the quality of extracted textlets they need to be extracted from a known object. A test setup was made, with three Kinect sensors, in a triangular setup, observing a rectangular box, with known side lengths as shown in Fig. 10a. Each textlet extracted from the scene has a relation $(n_{(i,j)}, \theta)$ to any other textlet in the scene. $n_{(i,j)}$ is the normal distance (for parallel planes the normal distance corresponds to the distance between the planes) from textlet i to j , i.e., the orthogonal projection of the center of one plane to the other plane (see Fig. 9). θ is the angular difference in the orientations. The computed relations are expected to agree with the constraints shown in Table 2.¹⁰ This is equivalent to the center of the green

areas in Fig. 10b: the two peaks at the top correspond to the two side-widths of the object (textlets being parallel but normal vectors pointing in opposite directions) while the left bottom peak corresponds to all textlets in the same plane, the bottom right peak corresponds to the relation between the table and the top surface (textlets being parallel and pointing in the same direction). The horizontal line at 90° angle corresponds to all 90° angles of textlet relations at different non-parallel surfaces and naturally ends at a normal distance corresponding to the height of the object.

The normal distances and angles from each textlet to all other extracted textlets are plotted as 2D histograms. This test was conducted with three Kinects calibrated to the same reference frame. First each Kinect were turned on individually in order to get a sample without interference. This can be seen for neighborhood radius 2, 6 and 10 in Fig. 11 a, b, c. For visual purposes, logarithmic values of the histogram bins are plotted. As expected the larger the radius of the textlets, the better the actual extraction quality and hence the more entries in the histograms are positioned in the expected areas indicated in Fig. 10b. The inlier percentages can also be seen in Table 1. Figure 11d–f shows the same diagram but with an additional Kinect turned on to generate noise. The last row of Fig. 11g–i is conducted with all three Kinects turned on at the same time. The degeneration of precision of the reconstruction is clearly visible in the larger variance when multiple Kinect cameras are used at the same time.

In order to quantitatively compare the measurements the threshold bounding boxes shown in Fig. 10b is applied. All relations between textlets within the bounding boxes shown in Fig. 10 (threshold of 10° and 5 mm), are defined as inlier relations. The inlier relations counted from scenarios with one, two and three simultaneously

¹⁰ If only one or two Kinects were used, it would not be possible to get all relations, due to an incomplete scene representation.

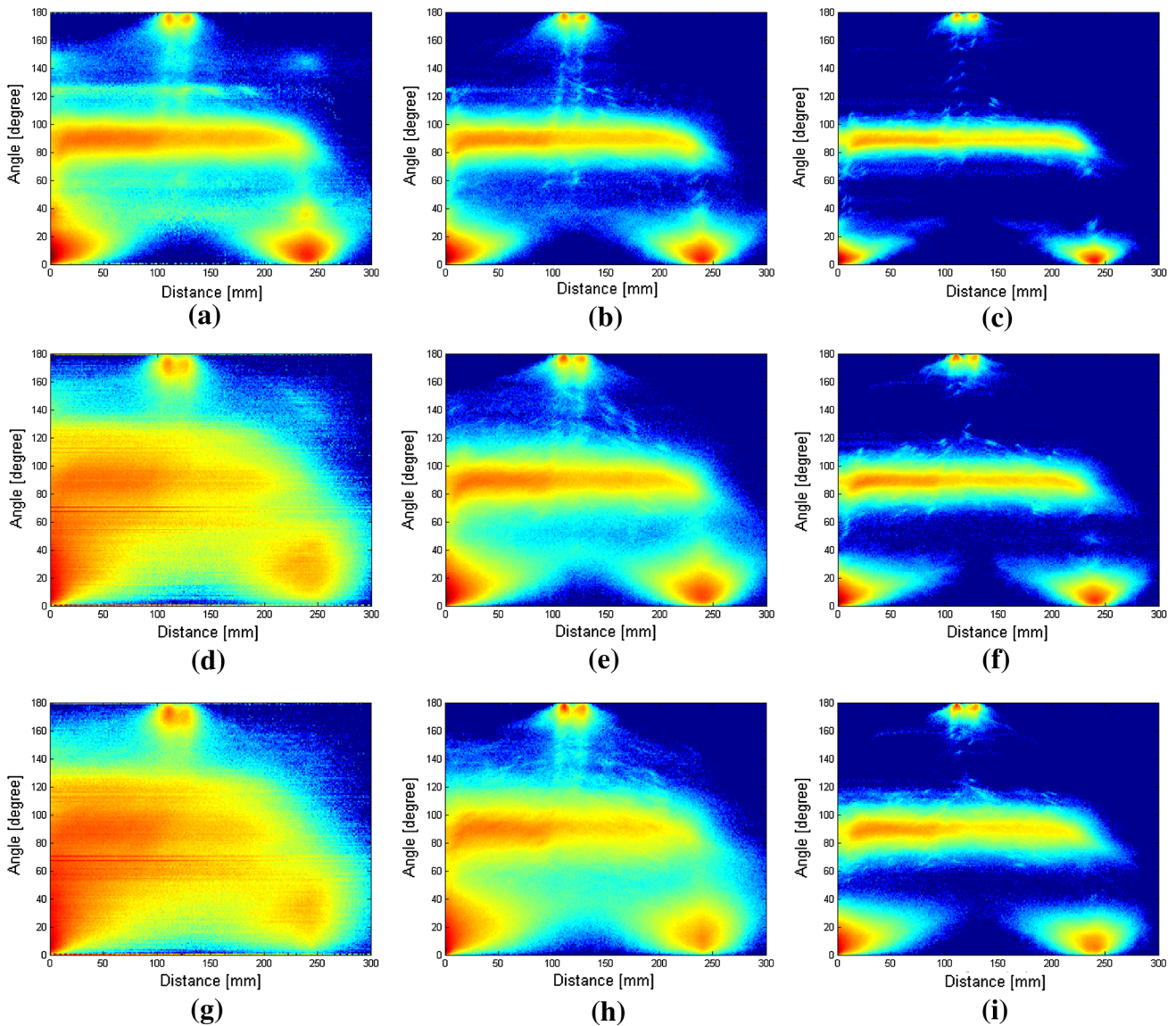


Fig. 11 Quality histograms. **a, b, and c** Each Kinect individually (neighborhood radius 2, 6 and 10). **d, e and f** Two Kinects at the same time (neighborhood radius 2, 6 and 10). **g, h and i** Three Kinects at the same time (neighborhood radius 2, 6 and 10)

enabled Kinect cameras can be seen in Fig. 13. The number of inliers are shown as a function of the size of the textlet (neighborhood radius). It shows clearly that using multiple Kinects synchronously degrades the overall quality of 3D textlets. The lower inlier count when using two or more Kinects are expected due to the overlap and interference of the two infrared patterns. Accordingly, more Kinects will add more interference. This interference is also evident when looking at the raw depth maps returned from Kinects in Fig. 12.

Figure 11 also indicates an important fact on the complexity of generating an appropriate model for the interference of multiple Kinect cameras: since the number of projections influences the reconstruction precision, the effect of interference depends on the actual 3D geometry of

the scene. For example, the top surface and also most of the base plane is subject to the infrared projections of all three cameras. As a consequence, textlets extracted from these areas are in particular affected. This is most visible (as being expressed in the high variance) in the peak on the bottom right (see Fig. 11i) which corresponds to the relation between the top surface and the base plate (see difference to Fig. 11c). For the side surfaces of the box, the corresponding peaks (the two peaks at the top in Fig. 11i) have a lower increase in variance (compared to Fig. 11c) since the surfaces are only subject to projections by one or maximally two Kinect cameras. This conclusion is also in agreement with the work in [2] which states that the reconstruction degradation depends on the relative orientation of the concurrently running Kinects. Hence, any

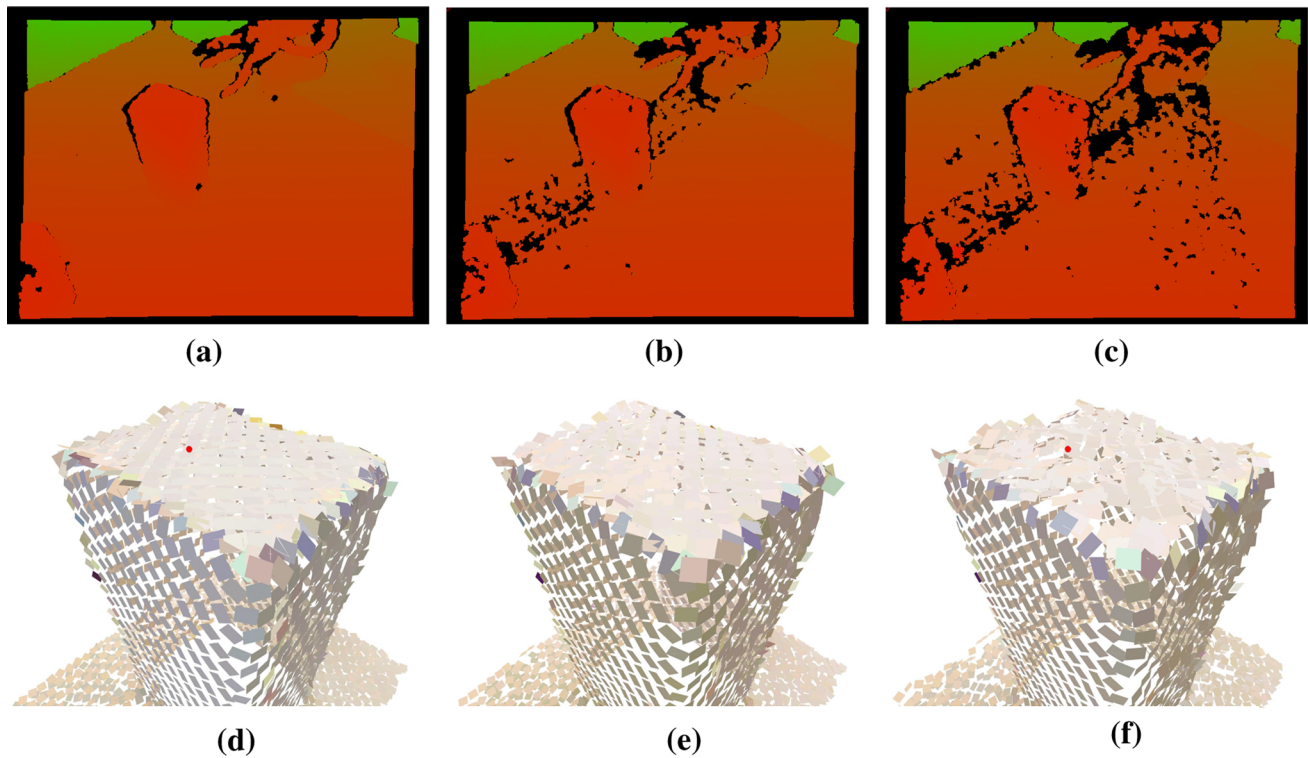


Fig. 12 Interference from multiple Kinects shown on colored depth map and texlets. **a, d** No interference. **b, e** Interference from one additional Kinect. **c, f** Interference from two additional Kinects. Note

that in the top surface the degradation effect is largest since all Kinect cameras project on the top surface while—due to occlusion—only one or two Kinect cameras project on the surfaces on the side

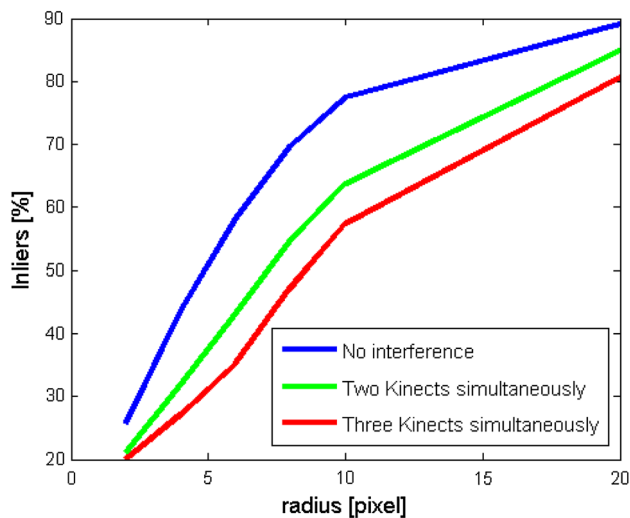


Fig. 13 Percentage of inliers with one, two and three Kinects

uncertainty model would need to explicitly model the scene geometry and the relative camera object relations.

5 Conclusion and future work

In this paper, we have described an algorithm for the real-time extraction of 3D surface patches (called ‘texlets’) with

an associated uncertainty model. To achieve high precision it was required to be able to describe the reconstruction process and the associated calibration problems in detail. We pointed out a number of issues that lead to degeneration of reconstruction quality which need to be taken care of before using Kinect cameras when high precision of the reconstruction is required. This requires in particular the understanding of the geometry behind the reconstruction process realized on Kinect cameras. To derive the uncertainty model for surface patches we used an approximate model for the uncertainty associated to a point-wise reconstruction with Kinect cameras. Hence, besides reaching the actual goal of real-time texlet extraction and the associated uncertainty we could provide information which will be of general use for the vision community and which will further facilitate applications of Kinect cameras requiring precise 3D reconstruction. This also includes the analysis of the interference effects when using multiple Kinect cameras. Here it would be interesting if new versions of the Kinect cameras would allow for a better control of the infrared projector to minimize these effects by switching the projector on and off easily and fast. All algorithms described in this paper are available via the C++ vision library CoViS.¹¹

¹¹ See <http://www.covis.org/>.

In Fig. 12 the effect of interference on the reconstruction is shown. When multiple Kinects are introduced the matching process seems to fail more often, giving empty spots in the depth map as seen in Fig. 12a–c. The interference also propagates to the textlets giving a visual degradation. This can be seen in 12d–f. Notice that the degradation is not present on the leftmost visible side of the box. Due to the triangular placement of the Kinects, only one pattern is projected to this surface in all cases. The rightmost visible side of the box suffers from the pattern of up to two Kinects, where the top of the box may have all three patterns interfering.

A particular challenge when using Kinect cameras is to get reliable information at depth discontinuities, which is also a problem with ToF cameras [24]. In [22], we have presented a 3D edge descriptor (covering 3D point and orientation information as well as a basic appearance description in terms of phase and color information for the left and right side of the edge) as part of our ECV system which is so far based on stereo information only. Integration of Kinect cameras and stereo information will allow to extend this edge descriptors by associating surface normals on both sides of the edge structure. In this context, we aim at addressing the issue of higher uncertainty at depth discontinuities.

Acknowledgments This work has been supported by the IntellAct project (FP7-ICT-269959).

References

- Başeski, E., Pugeault, N., Kalkan, S., Bodenhausen, L., Piater, J.H., Krüger, N.: Using multi-modal 3D contours and their relations for vision and robotics. *J Vis Commun Image Represent* **21**(8), 850–864 (2010)
- Berger, K., Ruhl, K., Brümmer C., Schröder, Y., Scholz, A., Magnor, M.: Markerless motion capture using multiple color-depth sensors. In: *Proceedings of Vision, Modeling and Visualization (VMV) 2011*, pp. 317–324, October 2011
- Carceroni, R.L., Katalakos, K.N.: Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. In: *Proceedings of Eighth IEEE International Conference on Computer Vision, ICCV*, vol. 2, pp. 60–67 (2001)
- Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
- Gottfried, J.-M., Fehr, J., Garbe, C.: Computing range flow from multi-modal Kinect data. In: *Advances in Visual Computing*, vol. 6938 of *Lecture Notes in Computer Science*, pp. 758–767. Springer, Berlin (2011)
- Hartley, R.I., Zisserman, A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge (2000)
- Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: *RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS* (2010)
- Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: *Proceedings of ACM SIGGRAPH* (1992)
- Jessen, J.B., Pilz, F., Kraft, D., Pugeault, N., Krüger, N.: Accumulation of different visual feature descriptors in a coherent framework. In: *Scandinavian Conference on Image Analysis (SCIA)* (2011)
- Julier, S.J., Idak Industries: The scaled unscented transformation. In: *Proceedings of IEEE Amer. Control Conf*, pp. 4555–4559 (2002)
- Khoshelham, K.: Accuracy analysis of Kinect depth data. In: *ISPRS Workshop Laser Scanning*, vol. XXXVIII (2011)
- Kim, Y.M., Chan, D., Theobalt, C., Thrun, S.: Design and calibration of a multi-view TOF sensor fusion system. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08*, pp. 1–7 (2008)
- Kim, Y.M., Chan, D., Theobalt, C., Thrun, S.: Design and calibration of a multi-view TOF sensor fusion system. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08* (2008)
- Kitagawa, G.: Monte Carlo filter and smoother for non-gaussian nonlinear state space models. *J. Comput. Graph. Stat.* **5**(1), 1–25 (1996)
- Klasing, K., Althoff, D., Wollherr, D., Buss, M.: Comparison of surface normal estimation methods for range sensing applications. In: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA'09*, pp. 1977–1982, IEEE Press, Piscataway, NJ, USA (2009)
- Kramer, J., Burrus, N., Herrera D.C., Echtler, F., Parker, M.: *Hacking the Kinect*. Apress (2012)
- Middleton, L., Sivaswamy, J.: *Hexagonal Image Processing : A Practical Approach*. Springer, London (2005)
- Murray, D., Little, J.J.: Patchlets: Representing stereo vision data with surface elements. In: *Seventh IEEE Workshops on Application of Computer Vision, 2005 (WACV/MOTIONS '05)*, vol. 1, pp. 192–199 (2005)
- Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: *ISMAR '11 Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136 (2011)
- Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: Surface elements as rendering primitives. In: *ACM SIGGRAPH*, pp. 335–342 (2000)
- Popović, M., Kootstra, G., Jørgensen, J.A., Kragic, D., Krüger, N.: Grasping unknown objects using an early cognitive vision system for general scene understanding. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, September 25–30, San Francisco, CA (2011)
- Pugeault, N., Krüger, N.: Temporal accumulation of oriented visual features. *J. Vis. Commun. Image Represent.* **22**(2), 153–163 (2011)
- Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: An open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*
- Reynolds, M., Dobos, J., Peel, L., Weyrich, T., Brostow, G.J.: Capturing time-of-flight data with confidence. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011)
- Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13 2011
- Saenko, K., Karayev, S., Jia, Y., Shyr, A., Janoch, A., Long, J., Fritz, M., Darrell, T.: Practical 3-D object detection using category and instance-level appearance models. In: *IROS* (2011)

27. Schuon, S., Theobalt, C., Davis, J., Thrun, S.: LidarBoost: depth superresolution for ToF 3D shape scanning. In: Proceedings of IEEE CVPR (2009)
28. Smisek, J., Jancosek, M., Pajdla, T.: 3D with Kinect. In: IEEE Workshop on Consumer Depth Cameras for Computer Vision, November 2011
29. Xia, L., Chen, C.-C., Aggarwal, J.K.: Human detection using depth information by Kinect. In: International Workshop on Human Activity Understanding from 3D Data in conjunction with CVPR (HAU3D), June 2011

Author Biographies

Søren Maagaard Olesen received his B.Sc. and M.Sc. degree in Computer Systems Engineering from the University of Southern Denmark, Denmark, in 2009 and 2011, respectively. He is currently a research assistant at the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interests include real-time computer vision and GPU computing.

Simon Lyder received his B.Sc. and M.Sc. degree in Computer Systems Engineering from the University of Southern Denmark, Denmark, in 2009 and 2011, respectively.

Dirk Kraft obtained a diploma degree in computer science from the University of Karlsruhe (TH), Germany in 2006 and a Ph.D. degree from the University of Southern Denmark in 2009. He is currently employed as an assistant professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interests lie within cognitive systems, robotics and computer vision.

Norbert Krüger is a Professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. He holds a MSc from the Ruhr-Universität Bochum, Germany and his Ph.D. from the University of Bielefeld. Norbert Krüger is leading the Cognitive Vision Lab which is focussing on computer vision and cognitive systems, in particular sensory-motor learning in the context of robot manipulation. He has also been working in the areas of computational neuroscience and machine learning.

Jeppe Barsøe Jessen received his B.Sc. and M.Sc. degree in Computer Systems Engineering from the University of Southern Denmark, Denmark, in 2007 and 2009, respectively. He is currently a Ph.D. student at the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interests include computer vision and GPU computing.