Extracting Sensor Noise Models – considering X / Y and Z Noise

Thomas Köppel *
Supervised by: Dr. Stefan Ohrhallinger †

Institute of Visual Computing and Human-Centered Technology
Technical University Vienna
Vienna / Austria

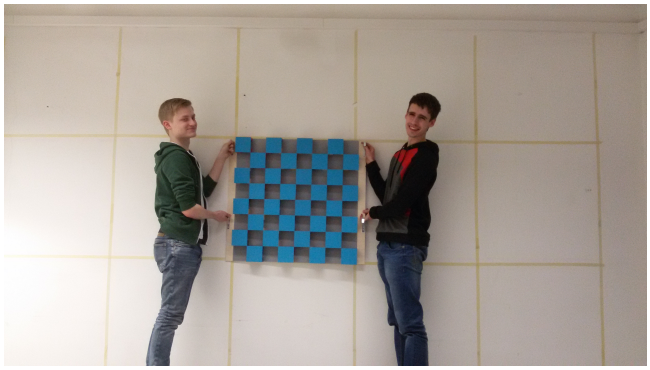

Figure 1: Conducting the Test Setup

Abstract

Depth maps or point clouds extracted by depth sensors are prone to have errors. The goal of this work is the extraction of these errors (the noise) and a statistical estimation using a noise model. We have combined the sensor noise model described in this work and the noise model described in the work by Grossmann et al. [4], generating a single noise model allowing a prediction of the amount of noise in specific areas of an image at a certain distance and rotation. We have conducted two test setups and measured the noise from 900 mm to 3.100 mm for the generation of the noise models. The test setup of this work focuses on determining the noise in X, Y and Z direction, covering the whole frustum of the respective depth sensor. Z noise was measured against a wall and X and Y noises were measured using a 3D chequerboard that was shifted through the room, allowing the above-mentioned coverage of the whole frustum. Along the edges of the cells of the chequerboard, the X and Y noise was measured. The combined model was evaluated by using a solid cube to classify the quality of our noise model. The estimation of the noise is important for applications like robot navigation that use data from depth sensors [2] or when reconstructing a 3D scene captured by a depth sensor. [7]

*e1327052@student.tuwien.ac.at
†ohrhallinger@cg.tuwien.ac.at

Keywords: noise model, surface reconstruction, sensor noise

1 Introduction

Depth sensors are used in various areas nowadays like computer vision, augmented reality, human computer interaction, the gaming industry and many more [7]. Over the last few years, many affordable depth sensors like the KinectV1, KinectV2 and the Lenovo Phab2Pro arrived on the market, offering depth sensors to a wider audience. But the resulting captures from depth sensors (depth maps or point clouds) suffer from noise. The identification and estimation of this noise is the main task of this work.

In general, our project consists of two parts focusing on different test setups. These two setups got combined and evaluated. The setup of Grossmann et al. [4] focused on axial and lateral noise measurement, using a plane placed in the centre of the captures with different rotations. My setup focused on the work by Choo et al. [1]. The main contribution of this paper was to identify the behaviour of noise when considering the whole frustum of the KinectV2.

Our main contributions are:

- Presenting an extraction algorithm for locating squares of a chequerboard in depth images
- Extracting sensor noise in X, Y and Z direction covering the whole frustum of the respective depth sensor
- Generating a sensor noise model out of the noise data and combining it with the work of Grossmann et al. [4]

2 Previous Work

”Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking”, published by Nguyen et al. [7] (based on the work by Khoshelham et al. [5]), was one of the first projects not only concentrating on axial noise in Z direction but also considering the lateral component, leading to better results especially in surface reconstruction. The main focus and contribution of their work was

identifying axial as well as lateral noise in the centre of the depth maps by considering different rotation angles and distances. The results of their measurements were included into the KinectFusion pipeline for 3D reconstruction, leading to better results, less holes and less noise – which increased the reconstruction accuracy. The main findings were the linear increase of lateral noise with distance and a quadratic increase of axial noise. [7]

"Statistical Analysis-Based Error Models for the Microsoft KinectTM Depth Sensor" [1], by Choo et al., proposed a new technique of acquiring noise models by considering the whole frustum of the KinectV2 depth sensor. Their main contribution was including the position of an object in the depth image to the noise model. The axial noise was measured against a flat surface. The lateral part was extracted using a chequerboard made from LEGO which covered the whole frustum of the captures, allowing lateral noise calculation at the borders of the cells [1]. The generated noise model, which described their measurements, showed a larger lateral noise in both X and Y direction compared to the model of Nyguen et al. [7]. Choo et al [1] showed the importance of the pixel's location relative to the centre, potentially increasing the quality of applications using the KinectV2.

3 Axial and Lateral Noise

The main focus of this work is the generation of a noise model that describes the noise of the KinectV2 and the Phab2Pro depth sensors considering the axial part in Z direction and the lateral part in X and Y direction. The noise is calculated as the standard deviation in mm based on differences between the ground truth and the measurements. Figure 2 shows the difference between axial and lateral noise.

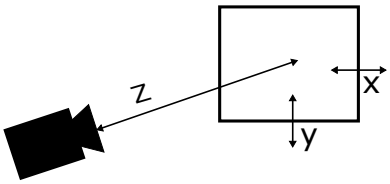

Figure 2: Explanation – Axial and Lateral Noise

Axial noise is measured along the viewing direction of the respective depth sensor. Deviations along this axis are considered being axial in Z direction. It describes the difference between the measured depth and the actual depth. We used a plane modelled according to the data and measured the deviations of each pixel to that plane for the noise calculation.

Lateral noise is measured at the axes perpendicular to the Z axis. The lateral noise in X direction is measured at the left and right edges of a rectangle, while the lateral noise in Y direction is measured at the top and bottom edges.

4 Our Setup

Our test setup used the method explained by Choo et al. [1] using a flat plane and a 3D chequerboard for the noise extraction. The depth sensors were placed onto a table in the respective distance, facing a wall. The chequerboard had a size of 76x76 cm. In order to allow a coverage of the whole frustum of the sensors, this board had to be moved along the wall. Therefore, a grid of masking tape was attached to the wall to avoid irregularities while measuring.

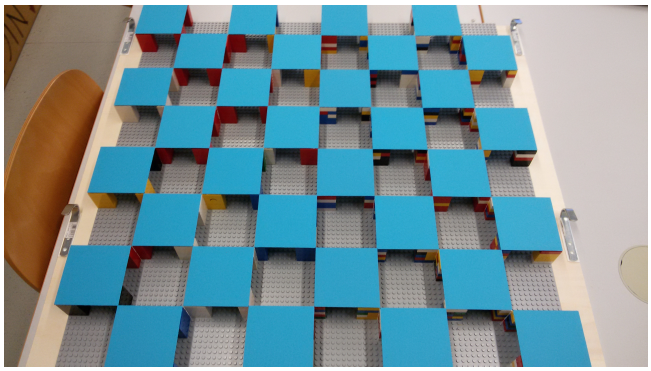

Figure 3: Chequerboard

Figure 3 shows the chequerboard. It consists of 32 heightened squares allowing measurements along the edges. The basis consists of a plywood plane with 4 LEGO baseplates. The blue carton squares with a width of 9.2 cm were elevated using LEGO bricks plugged on top of each other. The squares were fixed using double faced adhesive tape (see Figure 4). Metallic handles were attached to the sides in order to allow proper holding of the board when moving it across the wall.

In order to handle different cameras, we split the application into two parts, a depth sensor specific application for the data extraction and a Matlab application for the noise calculation and noise model generation. The two sensors were produced by two different companies allowing data extraction over their own APIs. We created two applications that use the respective APIs to extract the depth information needed for our calculations.

5 Axial Noise Calculation

The axial noise is the deviation in the direction of the Z axis. To achieve the extraction, point clouds were used. In

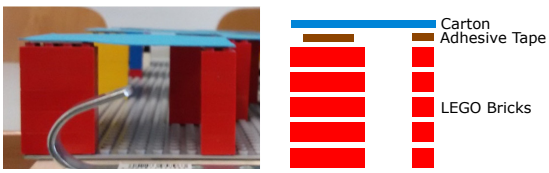

Figure 4: Single Square from the Side

our setup, the point clouds showing only the wall, covering the whole frustum were used. These point clouds were extracted at the distances from 900 mm to 3100 mm each 200 mm – captured 10 times. For the actual calculation of the axial noise, a plane got fitted through each of the point clouds. This plane was used as a ground truth of the wall. Afterwards, the deviation from the distance of the plane to the distance of the respecting points of the point clouds was calculated for each of the 10 captures. Considering these 10 point clouds, the standard deviation of each of these 10 captures was calculated for each pixel coordinate (X, Y). The standard deviation for each pixel is the axial noise in Z direction.

Figure 5 shows a point cloud of the KinectV2 where the sensor was placed at a distance of 900 mm away from the wall. The black surface is the fitted plane – the ground truth of the wall.

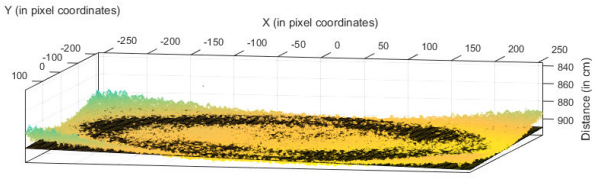

Figure 5: Point cloud of the KinectV2 at 900 mm distance with the fitted plane

5.1 Ground Truth Estimation using Plane Fitting

Our implementation used the Matlab function `fit` for the plane fitting estimating the ground truth of the wall. This function fits a plane trough the point cloud with minimal distances over all points. The result of the `fit` function is an equation that calculates Z dependent on the X and Y values.

$$z = a * x + b * y + c \tag{1}$$

The function above is a linear function in dependence of the variables X and Y. The Matlab function `fit` calculates the best coefficients for the variables a, b and c. This fitted plane was used as a ground truth for the noise calculation.

5.2 Noise Calculation

For calculating the axial noise, we used the Matlab function `feval`. This function used the fitted plane that was extracted before. For each data point with X and Y pixel coordinates, the distance from the plane was calculated, resulting in a distance value for each pixel coordinate (X, Y) for each distance for each of the 10 captures. The axial noise was the calculated standard deviation of all the distance values.

6 Lateral Noise Calculation

Like mentioned beforehand, our test setup also calculated the lateral noise in both X and Y direction. To achieve this, a 3D chequerboard was used. The chequerboard was measured, covering the whole frustum of the sensor at the respective distance. The board had to be moved across the wall and several captures had to be taken in each cell of the grid. Noise was extracted at the edges of each chequerboard field. Depending on how far the depth sensor was away from the wall, the chequerboard needed to be placed in a 3x3 up to a 5x4 grid. In each cell of the grid, 10 measurements were taken be the respective depth sensor.

6.1 Step 1 - Thresholding and Area Selection

Firstly, the depth maps needed to be transformed into a binary image to allow proper noise calculation along the edges. Figure 6 shows the obtained depth maps at a distance of 1100 mm normalized between 800 mm and 1200 mm. All of these depth maps cover the whole frustum of the sensor. In order to transform the depth maps into binary images, a threshold needed to be selected leading through the LEGO pillars. Using this threshold, objects closer to the camera – especially the blue carton on top of the LEGO pillars – were assigned 0 and all values behind – like the wall – 1. This threshold needed to be slightly adjusted for each distance and capture. Figure 6 shows the resulting binary images after thresholding at a distance of 1100 mm.

In order to improve the automatic detection of the rectangles of the chequerboard, a preprocessing step needed to be done because the structures of my colleague's and my body were disturbing the detection, leading to false positives. Therefore, a manual rectangle selection via a Matlab script was added. For each image, a rectangle was selected covering the position of the chequerboard in the respective image.

6.2 Step 2 - Square Detection

In order to detect the squares for the lateral noise calculation, we used the previously selected rectangular areas of the chequerboard and applied a Scale-Invariant Blob Detection in form of a Laplacian blob detector [6]. These detected blobs could be transformed into the desired squares.

The scale (σ) – the level where the blobs were found according to the Scale-Invariant Blob Detection – represents the radius of the blob, which was considered half the width of the square. The left upper corner of the squares was calculated via subtraction of the radius from the centre in x and y direction. From this left upper point, a square with a side length of 2*radius was constructed to represent a detected square of the chequerboard. Figure 7 shows this transformation. Figure 8 shows the detected squares

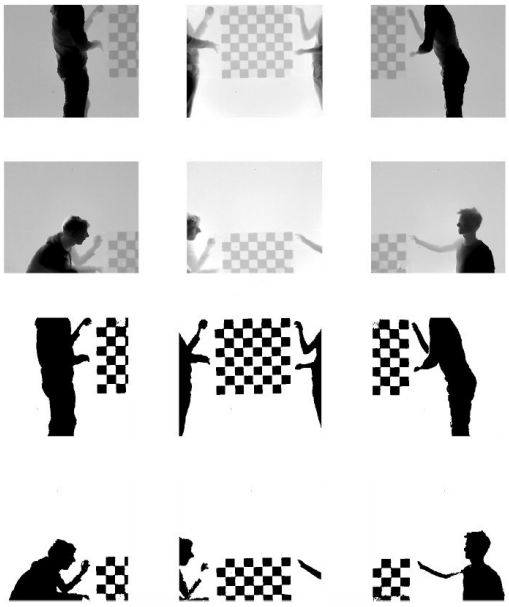

Figure 6: Depth Maps of the KinectV2, Distance = 1100 mm, Binary Images after Thresholding are beneath

at a distance of 1100 mm of the respective location of the chequerboard. These squares were used for further calculations.

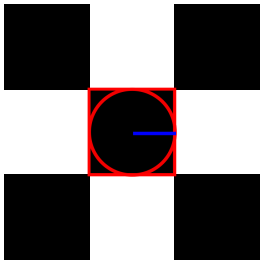

Figure 7: Transformation of the detected blobs to squares

6.3 Step 3 - Region of Interest Selection for X and Y Noise

The previously detected squares were used to select the Regions of Interest where the lateral noise can be calculated for the X and Y direction. Lateral noise in X direction was measured at the vertical edges of the detected squares and lateral noise in Y direction at the horizontal ones. Along these edges, rectangles were constructed with an extended width or height in both directions orthogonal to the detected edges, surrounding the edges. Figure 9 shows the constructed rectangles along the edges. They covered a part of the black coloured chequerboard's square and a part outside in the white background region. Blue rectangles mark the area for X noise while green rectangles mark the area for the Y noise calculation. Blue rectangles had been decreased in height and green rectangles

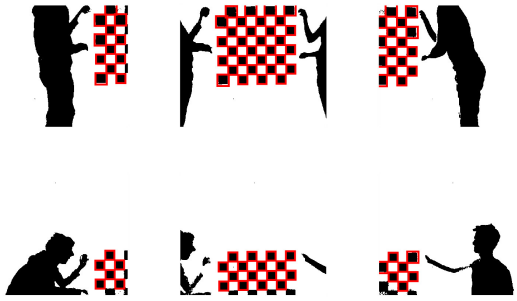

Figure 8: Detected Squares after the Laplacian Operator at a distance of 1100 mm

in width to minimize overflowing to other parts of the image. Along the edges in the blue and green rectangles, a standard deviation was calculated and noise extracted.

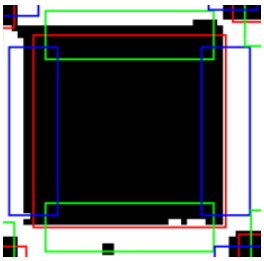

Figure 9: Detected Square with Rectangles for X and Y Lateral Noise Calculation

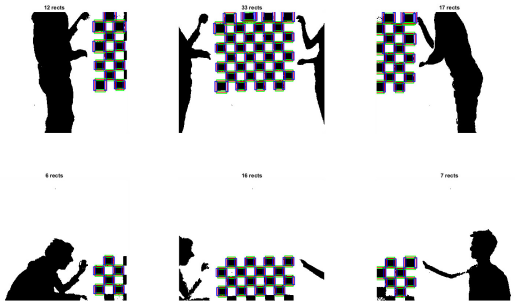

Figure 10: Detected Squares with Rectangles for X and Y Lateral Noise Calculation

6.4 Step 4 - Noise Calculation for X and Y Noise

Noise could be calculated in the before mentioned rectangles along the edges of the detected areas. The basic idea was to reduce the selected area to a white line with a width of one pixel. Using this row, the standard deviation was calculated and the noise in X and Y direction extracted.

Before the actual reduction, the rectangles needed to be transformed to an equal layout. The basic layout followed the structure of the green rectangle at the top of the squares. The matrix of the green rectangle on the bottom

simply needed to be flipped, the matrix of the blue rectangle on the right needed to be transposed and the blue rectangle on the left needed to be flipped and transposed.

The next step was the above-mentioned reduction to a 1 px line of white pixels (value = 1) for the calculation of the standard deviation. This algorithm works after following scheme:

Data: Binary values of a rectangular area
Result: Reduced 1 px line
start at first column;
while *currentColumn != lastColumn* **do**
  traverse from top to bottom;
  if *Value from current position == 1* **then**
    if *Value[row + 1] == 1* **then**
      set value at current position = 0;
    end
    if *Value[row + 1] == 0 and Value[row + 2] == 1* **then**
      set value at current position = 0;
    end
  end
  switch to next column;
end

Algorithm 1: Reduction to a 1 px row

The function traverses each column of the detected rectangles around the edges. If the value in the next row is 1 and the current value is 1, then the current value gets assigned 0. This removes the white areas above the black ones and leaves a 1 px – white line. The second `if` is necessary for eliminating black interfering pixels. Figure 11 shows an example of the reduction algorithm.

Figure 11: Reduction of the binary values to a 1 px line

After the reduction, the standard deviation was calculated using the Matlab function `std` over the positions of the white pixels. The returned standard deviation got multiplied by the distance and a constant, sensor specific factor.

$$\sigma_{[mm]} = \sigma_{[pixel]} * \text{distance} * 0.0028...\text{KinectV2} \quad (2)$$

$$\sigma_{[mm]} = \sigma_{[pixel]} * \text{distance} * 0.0057...\text{Phab2Pro} \quad (3)$$

The returned noise is the actual lateral noise in either X or Y direction, depending on the selected rectangle. The noise corresponds to the real-world noise in mm. This noise got saved with the current X and Y coordinates of the centre of the selected rectangle.

7 Noise Model Hypothesis

We used a splitting into regions. The noises of the X, Y and Z direction were handled independently. The previously calculated noise over the whole frustum was used as input data. This area got split up into 8 rows and 8 columns (like Choo et al. [1] suggested). In each of these cells, a mean value of the deviations was calculated. This average noise was then taken as a representative value for the respective cell. Figure 12 shows an example of the region splitting for the KinectV2 sensor in Z direction (axial noise), starting at a distance of 900 mm at the top left and ending at 3100 mm at the bottom right. The colouring emphasizes the average noise in the respective area. In the dark blue cells, the average error is near 0 mm and at the yellow areas about 12 mm. The farther away the measurements were conducted, the higher the average error got in these areas. Especially at the values measured above 2500 mm, the high noise produced by the depth sensor in the corners is visible. These average noise values for each region/cell are input values for RANSAC [3]. The noise model is fitted in the first step. RANSAC [3] was then executed 1000 times to remove gross outliers. Considering the KinectV2 in Z direction, 3.77% of the data values got classified as gross outliers, leading to an average estimation error of 0.6131 mm – meaning that the noise model in average deviates with 0.6131 mm. To compare, simply using a quadratic function for the fitting process without the splitting into regions and RANSAC would have led to an estimation error of about 3 mm.

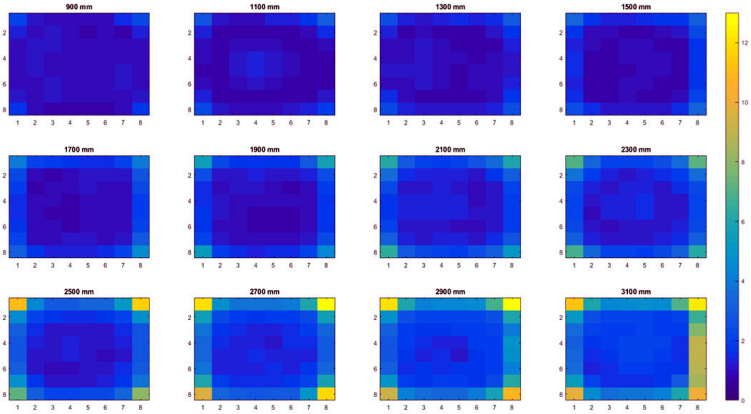

Figure 12: 8 x 8 regions of Z Noise in mm

While modelling the **axial noise**, we decided to further improve the results by using a cubic function. The KinectV2 Z noise model using a cubic function achieved an RMSE of 0.4852 mm and the Phab2Pro 1.0668 mm, enhancing the results. Using a cubic function also led to the best results at our evaluation. Table 2 shows the 30 coefficients for the functions for both depth sensors for the Z noise (see equation 4).

The **lateral noise** was modelled by using a 0.9 percentile overestimation. We considered each slice of mea-

	KinectV2 X	KinectV2 Y
$\alpha[1]$	6.6987e-09	3.9018e-09
$\alpha[2]$	-3.1781e-05	-1.3349e-05
$\alpha[3]$	0.0518	0.013148
$\alpha[4]$	-23.4839	1.8268
$RMSE(mm)$	0.1465	0.0959
	Phab2Pro X	Phab2Pro Y
$\alpha[1]$	$4.3031 * 10^{-9}$	$1.9045 * 10^{-9}$
$\alpha[2]$	$-1.8169 * 10^{-5}$	$-5.0756 * 10^{-6}$
$\alpha[3]$	0.0284	0.0066
$\alpha[4]$	-11.4981	-0.6622
$RMSE(mm)$	0.3031	0.5689

Table 1: Results of the X and Y Noise Models (Equation 5) using the Regions Approach with Quantiles

surements – like seen in Figure 12 – and computed the 0.9 percentile. Equation 5 was used during the fitting leading to the coefficients seen in Table 1.

$$\begin{aligned}\sigma(x,y,z) =& \alpha_{[1]} + \alpha_{[2]}z^1 + \alpha_{[3]}z^2 + \alpha_{[4]}y^1 + \alpha_{[5]}y^1z^1 + \\ & \alpha_{[6]}y^1z^2 + \alpha_{[7]}y^2 + \alpha_{[8]}y^2z^1 + \alpha_{[9]}y^2z^2 + \alpha_{[10]}x^1 + \\ & \alpha_{[11]}x^1z^1 + \alpha_{[12]}x^1z^2 + \alpha_{[13]}x^1y^1 + \alpha_{[14]}x^1y^1z^1 + \\ & \alpha_{[15]}x^1y^1z^2 + \alpha_{[16]}x^1y^2 + \alpha_{[17]}x^1y^2z^1 + \\ & \alpha_{[18]}x^1y^2z^2 + \alpha_{[19]}x^2 + \alpha_{[20]}x^2z^1 + \alpha_{[21]}x^2z^2 + \\ & \alpha_{[22]}x^2y^1 + \alpha_{[23]}x^2y^1z^1 + \alpha_{[24]}x^2y^1z^2 + \alpha_{[25]}x^2y^2 + \\ & \alpha_{[26]}x^2y^2z^1 + \alpha_{[27]}x^2y^2z^2 + \\ & \alpha_{[28]}x^3 + \alpha_{[29]}y^3 + \alpha_{[30]}z^3\end{aligned} \tag{4}$$

$$\sigma(z) = \alpha_{[1]}z^3 + \alpha_{[2]}z^2 + \alpha_{[3]}z + \alpha_{[4]} \tag{5}$$

The noise model described in this work and the noise model of Grossmann et al. [4] were combined into a single noise model describing the axial and lateral noise for both the KinectV2 and the Phab2Pro, taking the X and Y pixel coordinates, the distance and the rotation into account. For the axial noise calculation, we decided to use a weight function for the combination of our two noise models. The model described in this work does not take rotation into account, therefore, the more rotated the surface is, the less influence does the noise model of this work have. The model of Grossmann et al. [4] is entitled Model 1 and the model of this work Model 2 in the following section.

Due to consistency, we used the 0.9 percentile for both depth sensors for the two noise models, overestimating the lateral noise so that 90% of the measured noise should be below the estimated lateral noise for the X and Y direction from our combined noise model. This procedure of taking the 0.9 percentile is oriented at the work by Fankhauser et al. [2].

	KinectV2 Z	Phab2Pro Z
$\alpha[1]$	6.569	-20.2165
$\alpha[2]$	-0.0063664	0.025415
$\alpha[3]$	4.5605e-06	-1.7253e-06
$\alpha[4]$	-3.2304	11.28
$\alpha[5]$	0.0029717	-0.01511
$\alpha[6]$	-1.6241e-06	3.0272e-06
$\alpha[7]$	0.46318	-1.5463
$\alpha[8]$	-0.00042755	0.0019859
$\alpha[9]$	2.088e-07	-4.2756e-07
$\alpha[10]$	-2.9137	11.0018
$\alpha[11]$	0.0027723	-0.01431
$\alpha[12]$	-1.6192e-06	2.6006e-06
$\alpha[13]$	2.0513	-4.9918
$\alpha[14]$	-0.0021142	0.0070606
$\alpha[15]$	8.908e-07	-1.5391e-06
$\alpha[16]$	-0.27216	0.6874
$\alpha[17]$	0.00028939	-0.00097463
$\alpha[18]$	-1.1505e-07	2.2922e-07
$\alpha[19]$	0.30466	-1.2972
$\alpha[20]$	-0.00025628	0.0015642
$\alpha[21]$	1.6642e-07	-2.8225e-07
$\alpha[22]$	-0.20004	0.54637
$\alpha[23]$	0.00020537	-0.00078213
$\alpha[24]$	-9.2035e-08	1.7218e-07
$\alpha[25]$	0.026884	-0.076423
$\alpha[26]$	-2.8628e-05	0.00010955
$\alpha[27]$	1.1963e-08	-2.6047e-08
$\alpha[28]$	-0.0037752	0.013358
$\alpha[29]$	-0.0029981	0.0101
$\alpha[30]$	-1.9996e-10	-5.6216e-10
$RMSE(mm)$	0.4852	1.0668

Table 2: Resulting coefficients for the Noise Model (Equation 4) using the Regions/Ransac Approach – Cubic Function

8 Evaluation

The combined noise model was evaluated through a real-world scenario using a solid pressboard cube measured with the KinectV2 and the Phab2Pro sensors. The cube size is 300 x 300 x 300 mm. The cube got placed at different positions covering the whole frustum in a 3 x 3 grid. At each of these positions, the distance and the rotation of the cube was altered. At a distance of approximately 1 m and 2 m, measurements were conducted, representing a near and a far distance. A 0 and 45 degree rotation was used in order to measure the cube, showing one and two faces. Through tilting the depth sensors, depth images highlighting 3 faces were measured. The noise calculation of the X, Y and Z direction is similar to the described method, but only 1 image was used per distance/rotation/position. These measured noises got compared to the combined noise model afterwards. The rotation and distance could be extracted manually due to the simplicity of the evaluation setup.

The **axial noise** was measured at the captured surfaces of the cube. At different orientations either 1, 2 or 3 faces were visible that were selected manually during the evaluation. In the next step, a plane was fitted through the captured data points and the noise was extracted – similar to the test setup – by calculating the standard deviation from the plane to the data points. The average distance value of each plane was taken as the Z parameter for the evaluation. Considering the rotation, the normal vectors between the plane and the camera were used, allowing the calculation of the angle.

The **lateral noise** was measured at the borders of the cube similar to the extraction process in the test setup described in this work. Due to the evaluation setup using a rotated cube, the Y noise had to be extracted with an addition because of the tilted borders in the captures. Figure 13 shows an example. The red square in the left image shows an explanatory area for the Y noise extraction of the cube. In this area the reduction algorithm – explained in 1 – was used, leading to a line with a width of 1 px, as seen on the right side of Figure 13. In contrast to the test setup, this line is tilted. Simply calculating the standard deviation of the white pixels did not lead to the correct result. Therefore, we fitted a line (red) through the diagonal line and calculated the standard deviation from that line. The borders of the square were selected manually during the evaluation process. The X and Y pixel coordinates of the centres of the red squares and the distance of the cube were used for the evaluation.

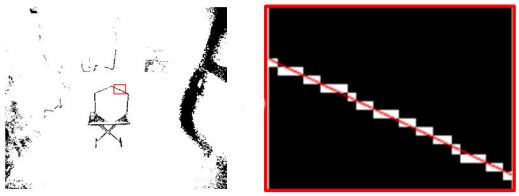

Figure 13: Y Noise Extraction during the Evaluation

8.1 Results

We decided to use the RMSE as a quality measure showing the deviation from the measured noise of the evaluation setup compared to the estimation of our combined noise model. Because we used the 0.9 percentile, overestimating the lateral noise component, we checked, how many of the measured lateral noises of the evaluation setup were under the specified threshold of the combined noise model.

Figure 14 shows the results of the axial evaluation on the top of the KinectV2 and on the bottom of the Phab2Pro. Each point in this scatter-plot represents a measured surface of the cube at the rotation given by the X axis and the distance given by the Y axis. Each point was coloured in a divergent colour scheme showing colours from red to blue. The bars on the right side of the scatter-plots show the different deviations from the estimated model, highlighted by a colour. If a point in the scatter-plot is white, the RMSE is between -0.7 mm and 0.7 mm. The red colour shows a higher positive deviation and the blue colour shows a higher negative deviation.

	KinectV2	Phab2Pro
Model 1	0.8947	2.5146
Model 2	1.5197	4.1739
Combined (averaged)	1.1780	2.0438
Combined (weighted)	0.8926	1.7856

Table 3: RMSE for the Axial Evaluation [4]

	KinectV2	Phab2Pro
X – Combined (averaged)	94.4	93.4
Y – Combined (averaged)	94.4	94.8

Table 4: Resulting percentages of our estimated 0.9 percentiles

Model 2 generally underestimated the axial noise when the object got rotated because rotation was not measured during this test setup. Table 3 shows the RMSE values of the different models. The general underestimation of Model 2 is visible in the scatter-plots of Figure 14. The RMSE values of Model 2 are, therefore, generally higher than Model 1. Simply averaging the two models led to worse results than taking only Model 2. Therefore, we decided to introduce a weight function to our combined noise model. The more rotated the object is, the less influence does Model 2 have. In case of the KinectV2, the resulting combined noise model shows a slightly smaller RMSE, while the RMSE of the combined model for the Phab2Pro shows a significantly smaller error.

Figure 15 shows the evaluation of the lateral components X and Y, while Table 4 shows the percentage values of how many values at the borders of the cube were under our estimated threshold. As mentioned above, we used the 0.9 percentile for the lateral noises. The circles in the scatter-plots 15 show the X-deviations while the rhombi show the Y-deviations of the KinectV2 on the top and the Phab2Pro on the bottom. The X and Y axes describe the pixel coordinates of the measured border of the cube. The distances of the measurements are not shown in the scatter-plots, but they all lie between 1 m and 2 m. The percentage values of 4 show a sophisticating result, therefore, simply averaging the two noise models was sufficient for the lateral case.

9 Conclusion

We have created a combined noise model consisting of two different test setups, estimating axial and lateral noise for the KinectV2 and the Phab2Pro. Others have already developed noise models for the KinectV2, but we investigated the noise of the Phab2Pro for the first time.

The first setup was described by the work of Grossmann

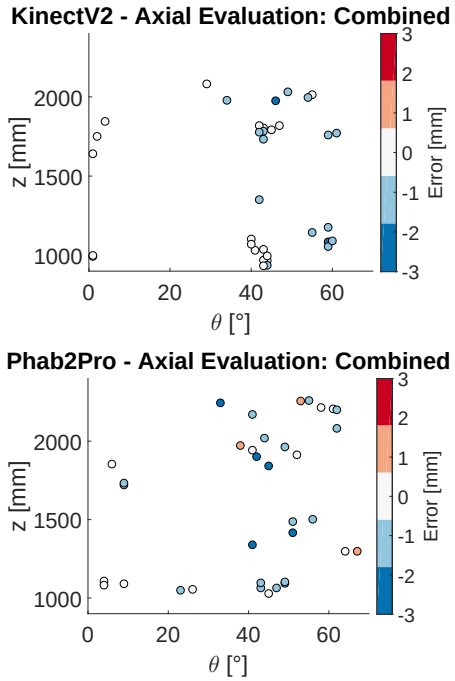

Figure 14: Results of the final Axial Model for both sensors [4]

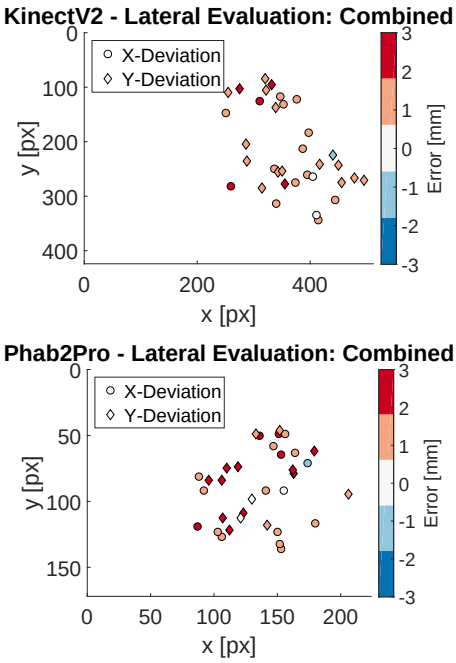

Figure 15: Results of the Lateral Evaluation

et al. [4] – measuring noise, using a plane placed at different distances, rotated to different orientations. The axial noise was measured at the surface of the plane, while the sensor was aligned horizontally and rotated by 90 degrees, in order to extract both the lateral noises in X and Y direction.

The second setup, described in this work, covers the whole frustum, taking the position in pixel coordinates and the distance into account. A 3D chequerboard was used to extract both lateral noises in X and Y direction at the edges of each field of the chequerboard.

Using the measurements of the two test setups, two empirically derived noise models were created and combined into a single one by using a weight function. The combined model was evaluated using a solid cube placed at different locations and orientations, leading to sophisticating results underlining the quality of our resulting noise model.

The model could further be improved by experimenting with different fitting functions and developing other improvements and techniques specifically aimed at a better coverage of the data, resulting in a smaller RMSE.

References

[1] Benjamin Choo, Michael J. Landau, Michael D. DeVore, and Peter A. Beling. Statistical analysis-based error models for the microsoft kinect depth sensor. In *Sensors*, 2014.

[2] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 388–394. IEEE, 2015.

[3] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[4] Nicolas Grossmann. Extracting Sensor Specific Noise Models. Bachelor's thesis, TU Wien, Austria, 2017.

[5] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[6] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[7] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 524–530. IEEE, 2012.