Laboration 2: ASUS Xtion Pro: Calibration, noise characterization and filtering

Sensors and Sensing

Michael Floßmann, Anders Wilkström

2015–12–07

1 Introduction: Structured light cameras

Structured light cameras are a low-cost option for depth measuring in three dimensional space. The cameras project a known light pattern to a scene and record the reflection of that light pattern. This recorded data is then used for triangulation.

For this lab, the ASUS Xtion Pro sensor was used as a structured light camera.

2 Task and implementation

The task at hand was to set up and calibrating the sensor, as well as to characterize the noise in the depth measurement and to set up filtering routines.

2.1 Basic setup

To set up the camera, the package `openni2` for ros-indigo was used. When starting the openni ROS drivers for the camera it publishes a wide range of topics from the camera.

For this laboration, only the topics which publish a viewable image were of interest. This included two main topics:

- `/camera/rgb/`
  This topic publishes data from the RGB camera on the ASUS Xtion Pro. The topic `/camera/rgb/raw` shows the unprocessed RGB image like a regular camera. A sample image from this topic is shown in figure 1.
- `/camera/depth/`
  This topic publishes the depth data as a 2D-array of float variables containing the depth values in meters. A sample image from this topic is shown in figure 2.
- `/camera/depth_registered`
  This topic combines the RGB and the depth image into a coloured point cloud. A visualization of this topic through the tool `rviz` is shown in figure 3.

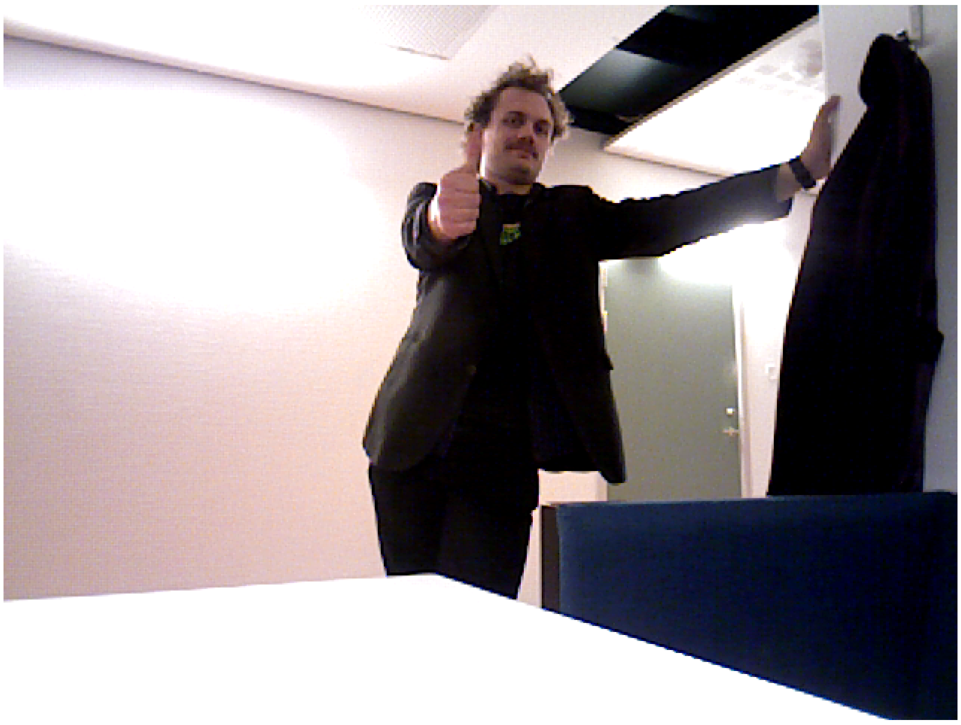


Figure 1: Output image of `/camera/rgb/raw`

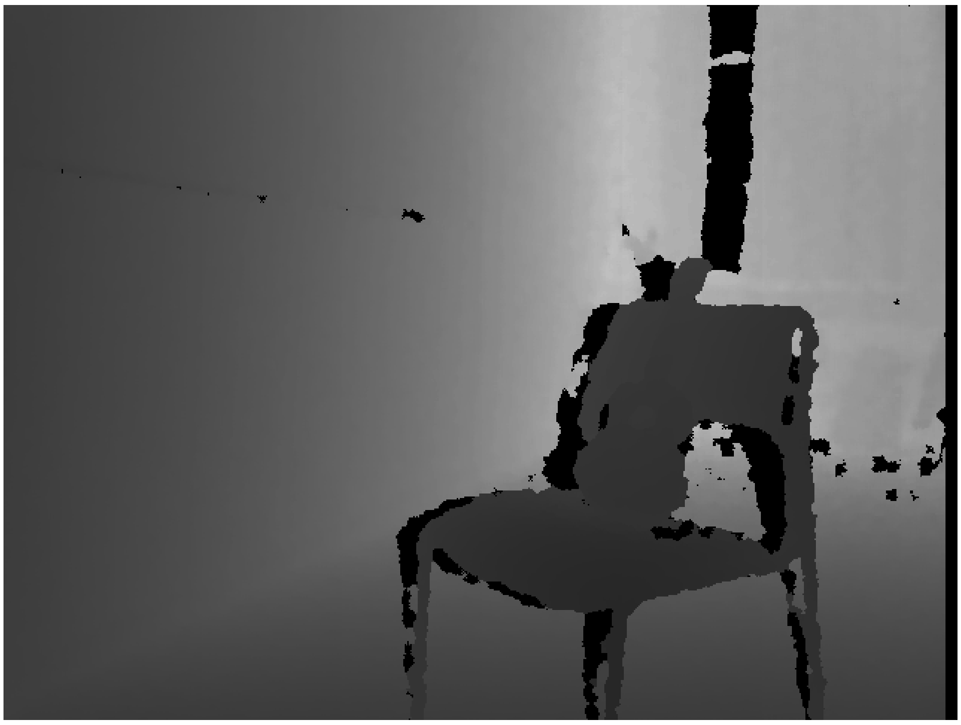


Figure 2: Original depth image

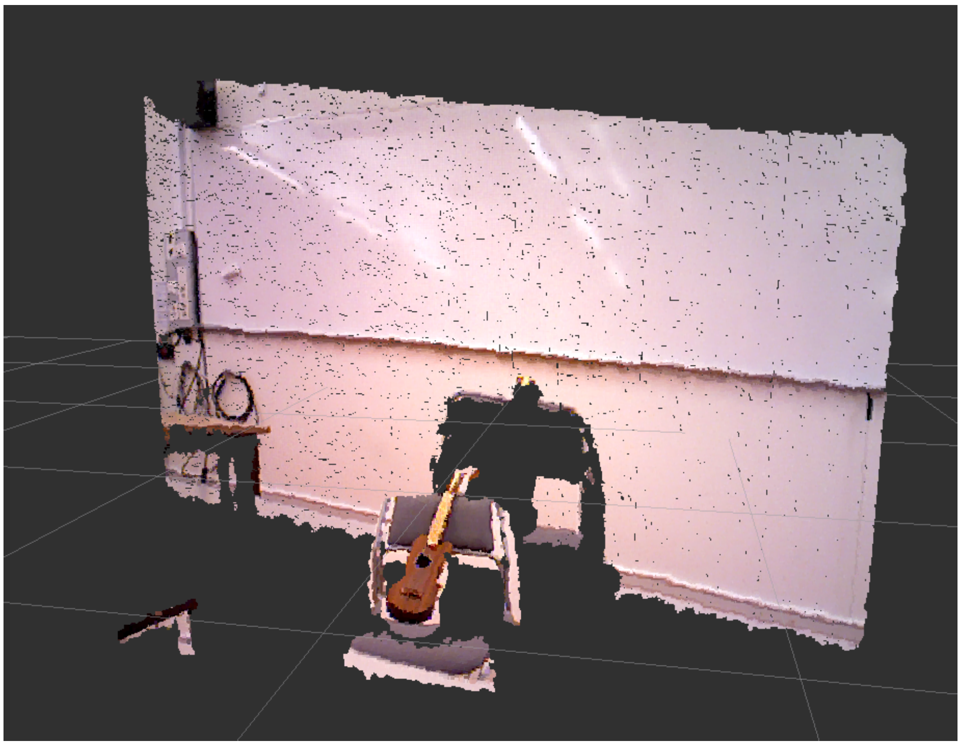


Figure 3: Screenshot of the vizualized pointcloud of `/camera/depth_registered`

2.2 Basic ROS node

After the basic setup, a ROS node template was used as a base to process the images and point clouds published. The received images are shown in figure 1 and figure 2.

2.3 Color camera calibration

In this part the color camera was calibrated using a checkers board print. The camera calibration is used to remove distortion in the image caused by imperfection in the camera such as the lens not being perfectly aligned to the center. The ROS package camera_calibration was used to obtain the calibration values for the camera. To do the calibration we moved the board around within the cameras field of vision make it gather data points. The output is a set of parameters that tries to minimize the distortions when applied to images of real world objects such as the checkers board. More detailed information on the calibration process can be accessed through the following url: `http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration`

The improvement of the calibrated picture isn't quantitatively measurable with the tools at hand, so the verification has to be carried out with judgment by the naked eye. It is possible to see a small difference between the images, but it seems like the calibration added distortions. In the original image figure 4 no errors can be seen but in figure 5 a distortion at the bottom of the image is observible, as the line at the very bottom clearly isn't straight with its' ends vanishing at the left and right sides.

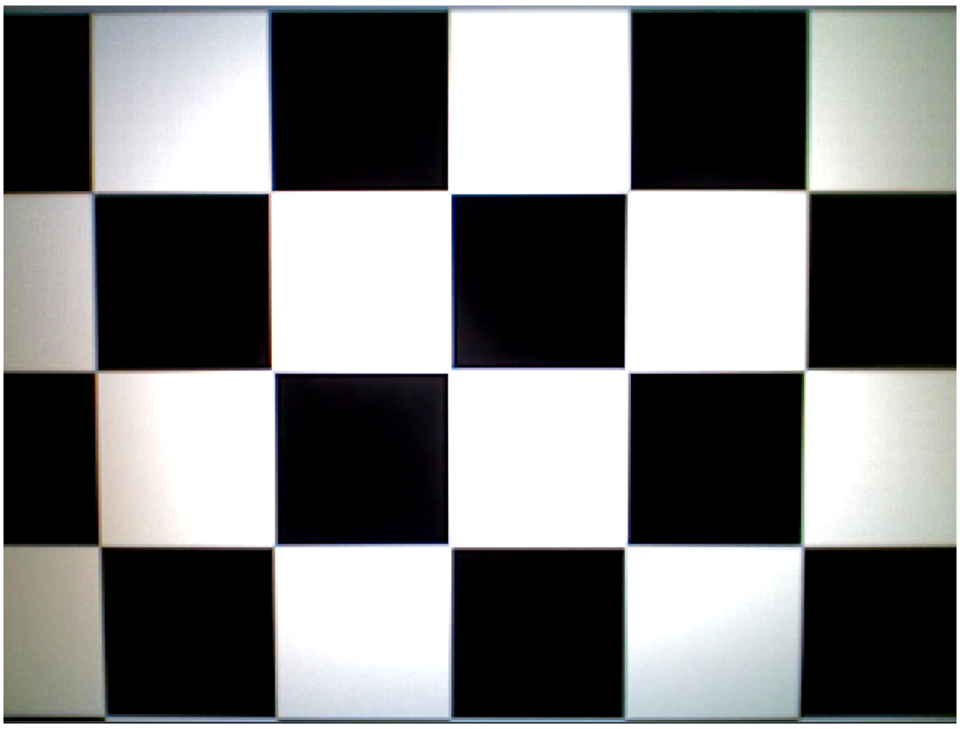


Figure 4: Uncalibrated checkers board image

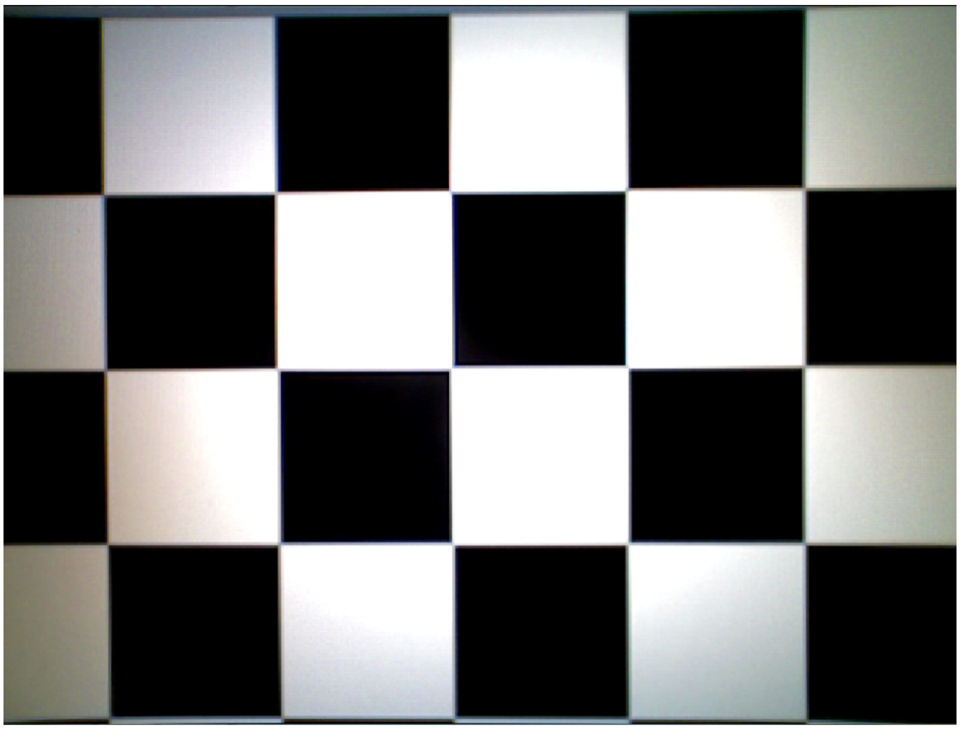


Figure 5: Calibrated checkers board image

2.4 Noise characterization

In this part we cropped small windows of varying sizes in the center of the depth image. The camera was then placed at distances varying from 0.6 to 1.8 meters as measured by measuring tape. The mean and variance of the distance values was then calculated from the depth values for the different cropped windows.

The mean error of 10 seconds of measurement in relation to the distance is shown in figure 6. The variances are shown in figure 7.

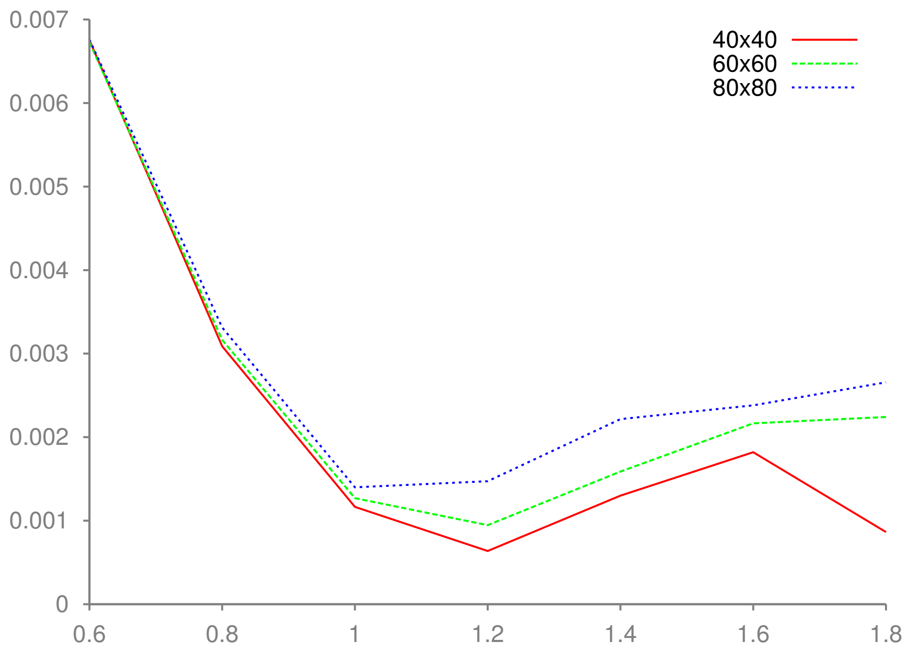


Figure 6: Mean absolute error in meters of the mean depth in relation to the ground truth distance

Analysing the mean error (fig. 6), one can see that the error values diverge with the error size and the distance to the wall. Since both the ground truth distance value and in the distances measured by the camera contain errors, the accumulative error can grow substantially without further knowledge of the specific error source causing the discrepancy. But if we model the ground truth error as a relative error (e.g. $\pm 5\%$), the absolute ground truth error would grow with increased distance. But since our error decreases with distance the error below one meter is possibly a systematic error in the depth measurement. More precise ground truth measurements are required to determine if this is the case. The diverging error values and increasing variances can be explained by the geometric characteristics of the measurement (see fig.8).

Figure 8 shows a 2D schematic view of the measurement. The camera records the depth values of an area around a maximum value of θ. The depth value at C doesn't contain the actually desired distance to the wall d, but rather

$$h = \frac{d}{\cos\theta}.$$

As the distance to the wall d increases, so does the absolute error between h and d. Increasing

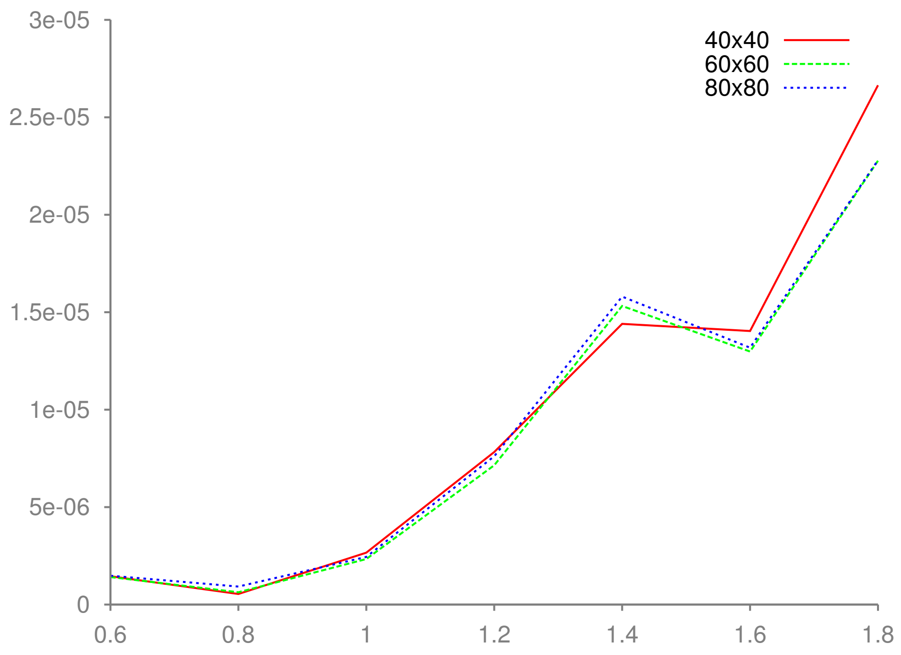


Figure 7: Mean of the variance of the depth in relation to the ground truth distance

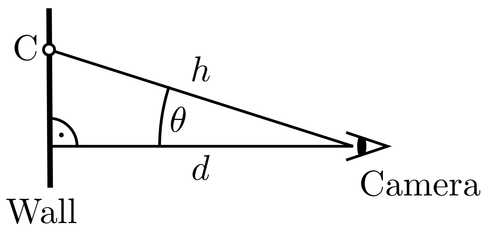


Figure 8: Schematic of the distance measurement

the observed window size additionally increases θ, which has a similar effect.

2.5 Noise filtering

In this part we applied several filters to the depth images in order to remove the noise from the measurement data. These filters are:

- Gaussian blur
- Median filter
- Bilateral filter
- Median over several image samples
- Average over several image samples

The effect on the image data after the application of the filters will be discussed in the following.

2.5.1 Filter effects

The original image is seen in figure 2.

Gaussian blur The gaussian blur filter (fig. 9) blurs the contours of the observed objects. Additionally, the pixels with `NaN` values reproduce and strongly disturb the algorithm. This can be explained as anytime one of the pixels in the kernel is a `NaN` value, the resulting output pixel also becomes `NaN`, regardless of any other pixel's value. This also explains the "blocky" artifacts in areas which originally contained small `NaN` speckles.

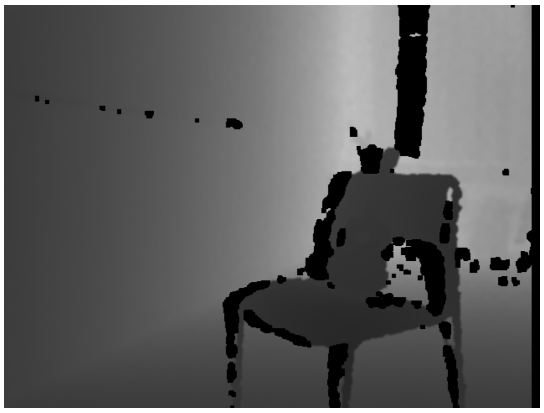


Figure 9: Gaussian blur

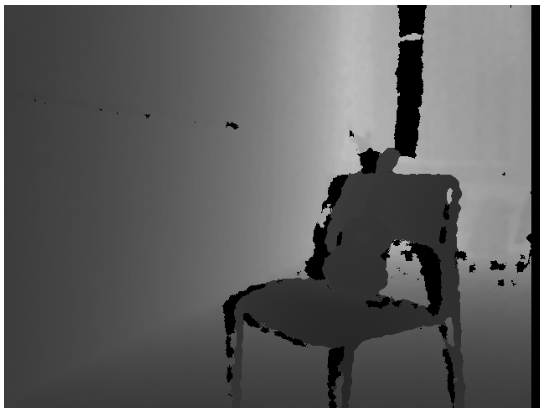


Figure 10: Median filter

Median filter The median filter (fig. 11) doesn't show a big difference to the original image.

Bilateral filter The bilateral filter (fig. 13) blurs the contours of the objects. Otherwise, no remarkable differences to the original image are discernible.

In order for this filter not to produce very artifact-prone images, the `NaN`-values had to be adjusted. Before applying the filter all `NaN` values were replaced with values either from the left or from above of the `NaN` value. After running the filter on the image the `NaN` values from the

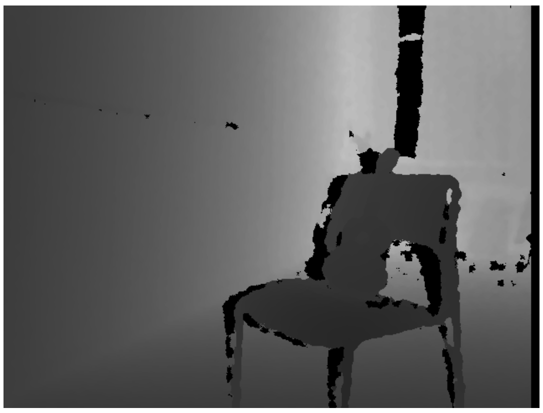


Figure 11: Median filter

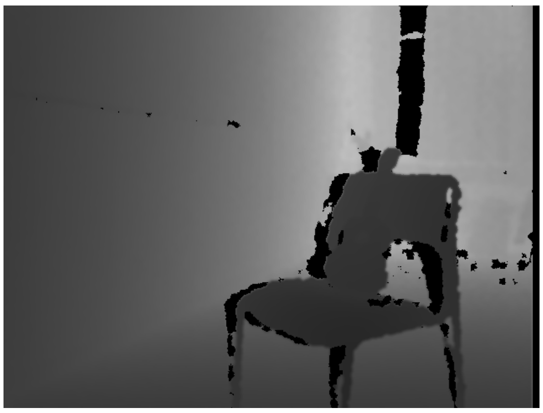


Figure 12: Bilateral filter

original where reapplied to the filtered image. This technique is not ideal since it can produce strange values new edges of large blocks of `NaN` values. It would have been also possible to modify the bilateral filter to return `NaN` if it encounters any `NaN` value in the kernel.

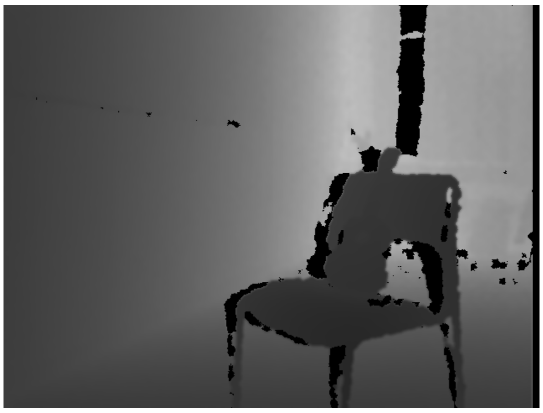


Figure 13: Bilateral filter

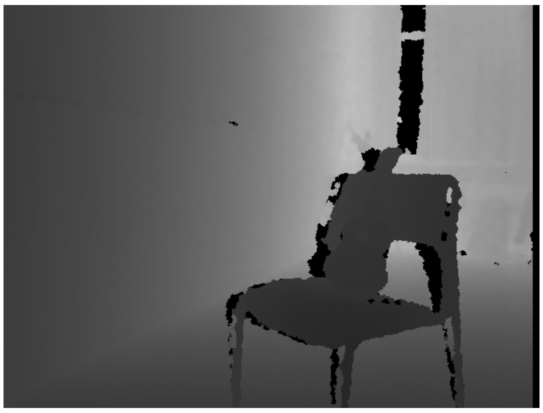


Figure 14: Moving median image

Median over several image samples The moving median image (fig. 14) leaves the contours visible and removes smaller spots with `NaN` values. When the image moves too quickly, it has a ghosting effect, which extremely complicates any data interpretation.

Mean over several image samples The moving mean image (fig. 15) blurs the contours of the objects (although not as strongly as previous contour blurring algorithms). It also removes small speckles of `NaN` values. This filtering technique also suffers from too quick movement.

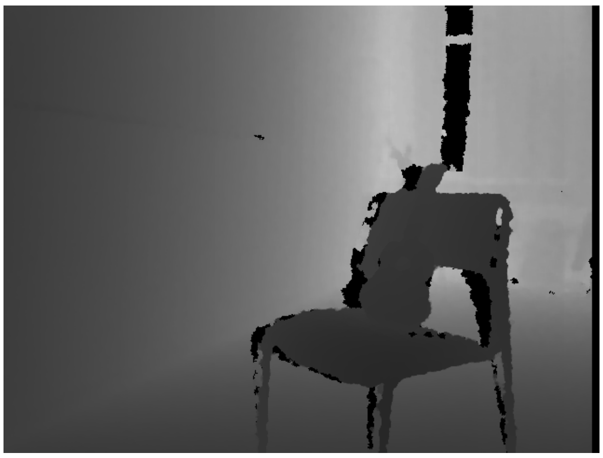


Figure 15: Moving mean image

2.5.2 Filters and distance error

In this part, the effects of the used filters is analyzed on the distance windows recorded in subsection 2.4.

All of the filters show only little change on the absolute error values. The variance values differ on the other hand, while roughly keeping the general form of the curves.

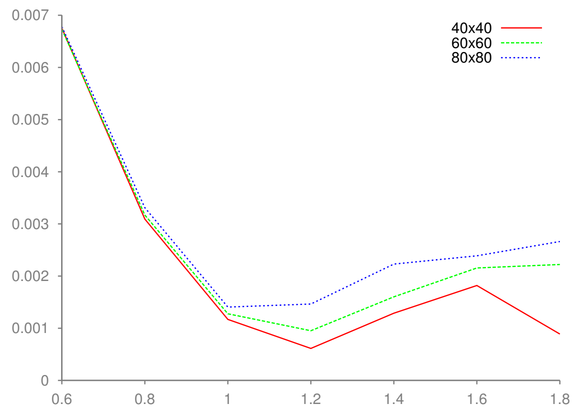
Gaussian filter The gaussian filter (fig. 16) decreases the variance values by a factor of roughly 0.66.

Median filter The median filter (fig. 17) decreases the variance values by a factor of roughly 0.8.

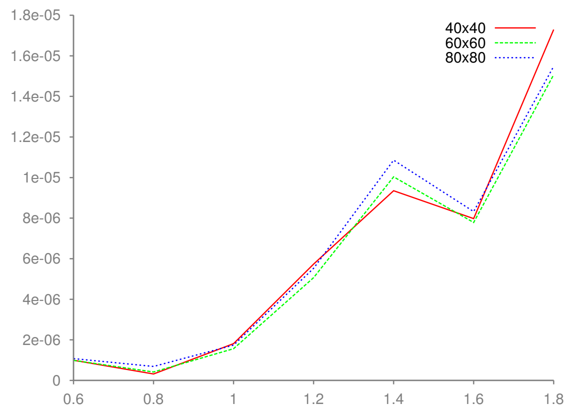
Bilateral filter The bilateral filter (fig. 18) decreases the variance values by a factor of roughly 0.5

Mean over several image samples The running median (fig. 19) image shows only little change on the variance curve.

Median over several image samples The running mean image (fig. 20) decreases the variance values by a factor of roughly 0.8.

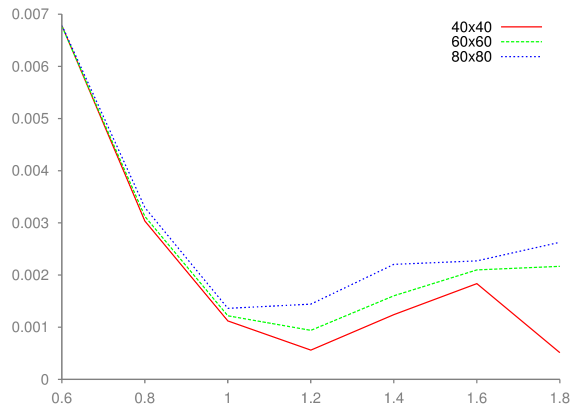


(a) Absolute Error

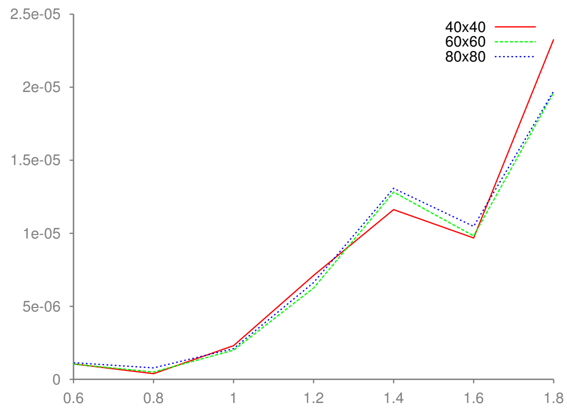


(b) Variance

Figure 16: Gaussian filter on distance measurement

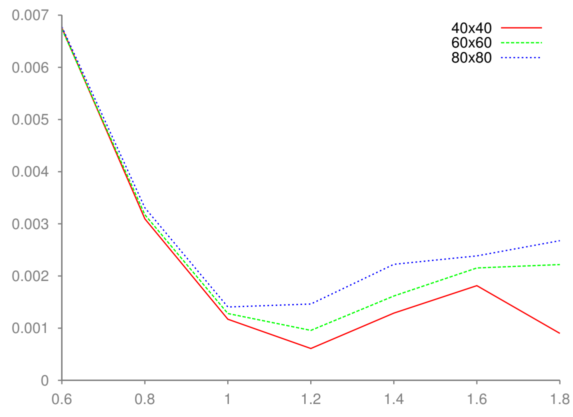


(a) Absolute Error

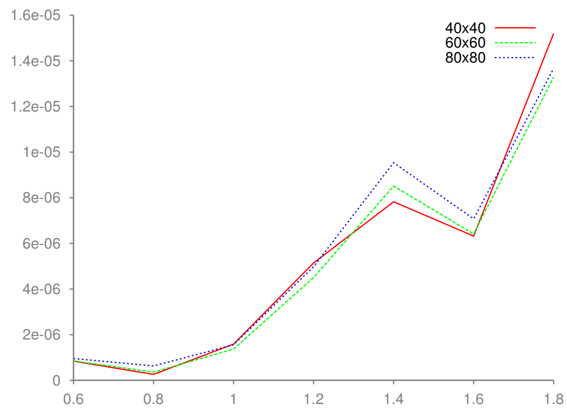


(b) Variance

Figure 17: Median filter on distance measurement

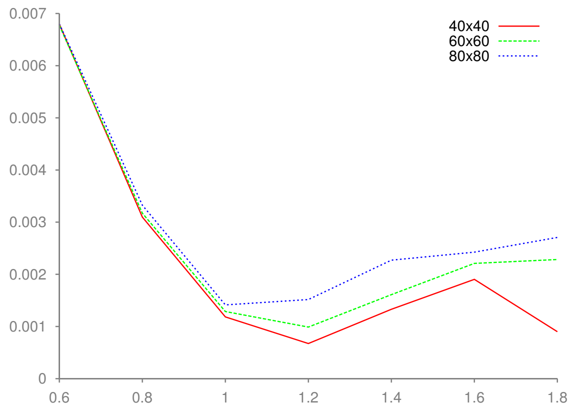


(a) Absolute Error

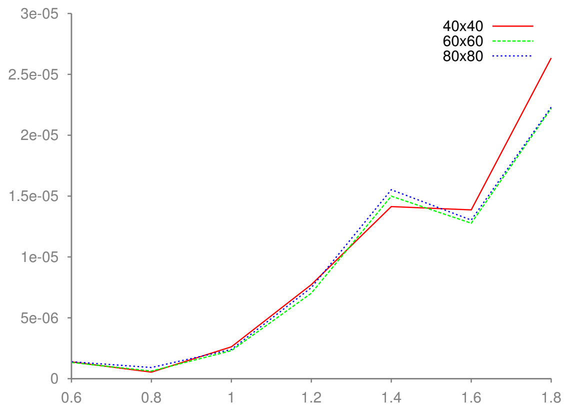


(b) Variance

Figure 18: Bilateral filter on distance measurement

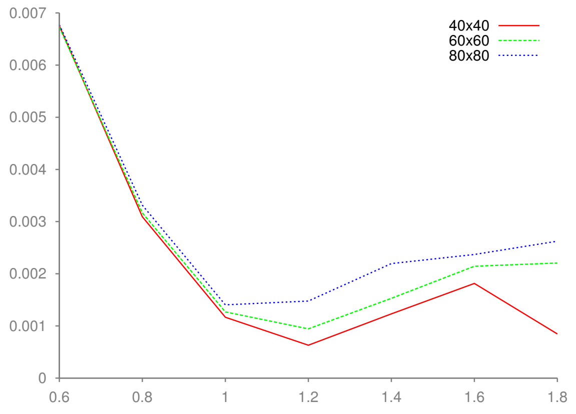


(a) Absolute Error

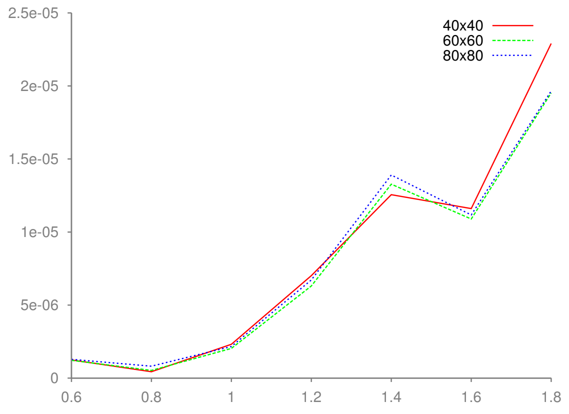


(b) Variance

Figure 19: Running median of the distance measurement



(a) Absolute Error



(b) Variance

Figure 20: Running mean of the distance measurement

2.5.3 Salt-and-pepper removal properties

The depth images didn't suffer from noise clearly visible to the eye. In order to test the noise cancelling properties of the filters, artificial salt-and-pepper noise was added to the image file. This was achieved by changing random pixels in the image to black or white in each received depth-image.

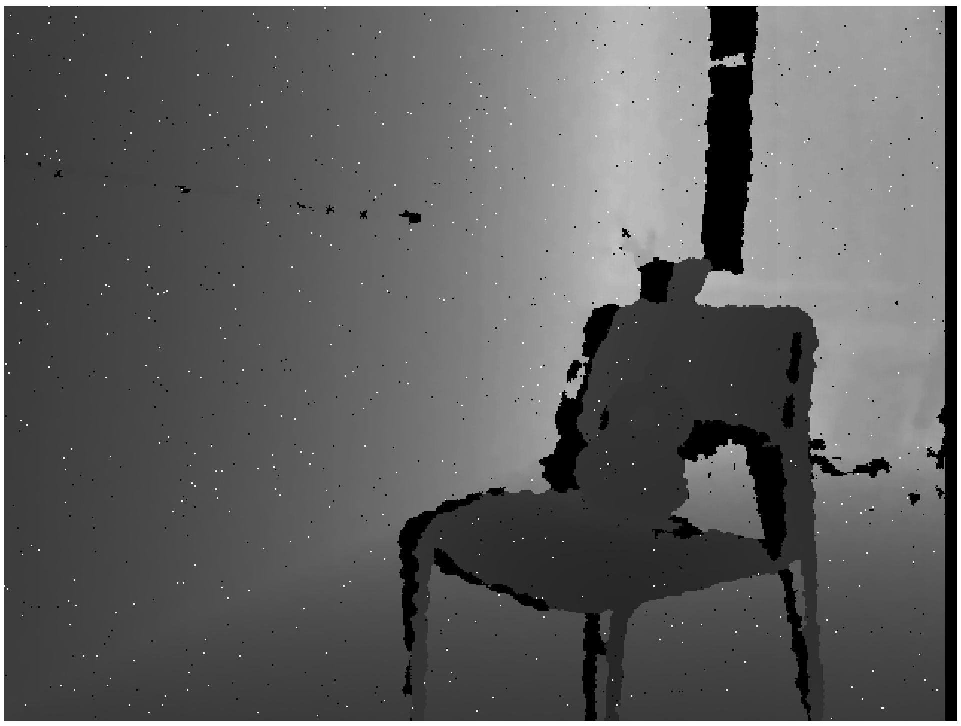


Figure 21: Original depth image with salt and pepper

Gaussian blur The gaussian blur (fig. 22) doesn't filter out the salt and pepper appropriately. It rather spreads the noisy data, negatively influencing the surrounding, proper data.

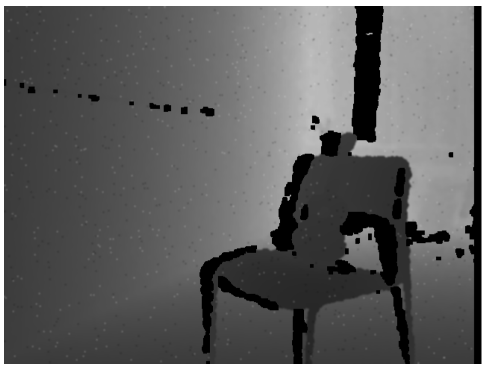


Figure 22: Gaussian blur with salt and pepper

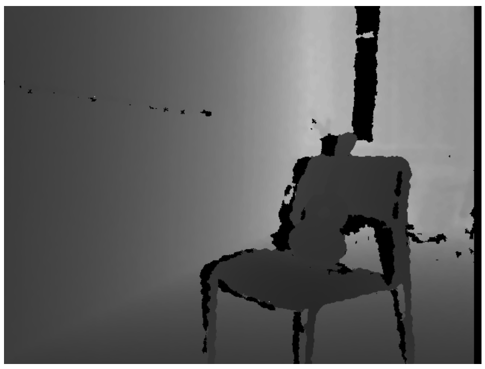


Figure 23: Median filter with salt and pepper

Median filter The median filter (fig. 23) removes the salt and pepper speckles reliably.

Bilateral filter The bilateral filter (fig. 24) decreases the influence of the salt and pepper speckles in the image, although they are still visible.

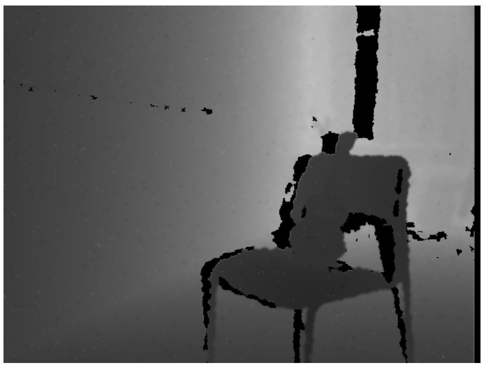


Figure 24: Bilateral filter with salt and pepper

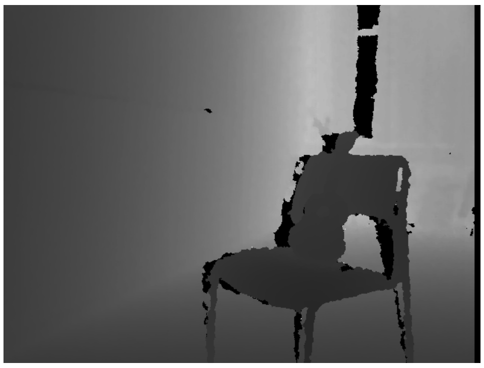


Figure 25: Moving median image with salt and pepper

Median over several image samples The bilateral filter (fig. 25) removes the salt and pepper speckles reliably.

Mean over several image samples The moving mean filter (fig. 26) accumulates the salt and pepper speckles of all recorded images. Although the speckles overall are more dull, their amount increased, still showing a considerable amount of noise.

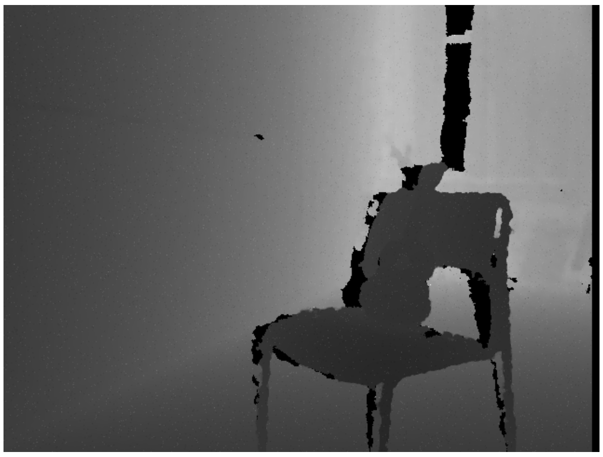


Figure 26: Moving mean image with salt and pepper