HIGH-QUALITY KINECT DEPTH FILTERING FOR REAL-TIME 3D TELEPRESENCE

Mengyao Zhao[1,2], Fuwen Tan[2], Chi-Wing Fu[1], Chi-Keung Tang[3], Jianfei Cai[1] and Tat Jen Cham[1]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
[2] BeingThere Centre, Institute for Media Innovation, Nanyang Technological University, Singapore
[3] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

ABSTRACT

3D telepresence is a next-generation multimedia application, offering remote users an immersive and natural video-conferencing environment with real-time 3D graphics. Kinect sensor, a consumer-grade range camera, facilitates the implementation of some recent 3D telepresence systems. However, conventional data filtering methods are insufficient to handle Kinect depth error because such error is quantized rather than just randomly-distributed. Hence, one could often observe large irregularly-shaped patches of pixels that receive the same depth values from Kinect. To enhance visual quality in 3D telepresence, we propose a novel depth data filtering method for Kinect by means of multi-scale and direction-aware support windows. In addition, we develop a GPU-based CUDA implementation that can perform real-time depth filtering. Results from the experiments show that our method can reconstruct hole-free surfaces that are smoother and less bumpy compared to existing methods like bilateral filtering.

Index Terms—Kinect, 3D telepresence, multi-scale filtering, direction-aware filtering

1. INTRODUCTION

3D telepresence is a next-generation multimedia application, allowing remote collaboration through a natural and immersive video-conferencing environment supported with real-time remote 3D graphics. By this, participants can have a perception of being co-located with others who are remote. The pioneering works, office of the future [1] and the blue-c system [2], are milestone projects toward this vision.

The launch of consumer-grade range cameras, such as Microsoft's Kinect [3], enables convenient and low-cost acquisition of 3D depth in real-time, thus not only facilitating numerous applications, such as 3D gaming, but also accelerating the progress in 3D telepresence research. One recent example is a system by Maimone and Fuchs [4].

However, when employing Kinect in 3D telepresence applications, we have to consider the following vital issues:

- First, raw depth data from Kinect is noise-prone and unstable. We need proper filtering to improve the visual quality of 3D data in the 3D telepresence applications;
- Existing data filtering methods on Kinect depth data mainly address uniformly-distributed random noise, but ignore depth quantization problem in the data;
- In 3D telepresence applications, e.g., tele-conferencing, the hardware depth sensors have to be mounted on a fixed location in the physical space, rather than being freely movable as in KinectFusion [5];
- The filtering process should run in real-time, so we cannot afford tedious computation with global data optimization, e.g., solving with Poisson equations.

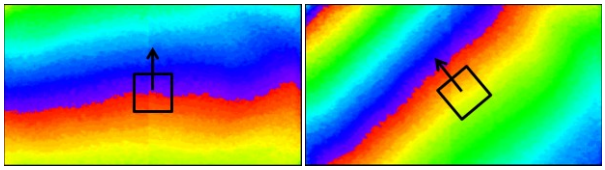
Considering these requirements, we propose a multi-scale direction-aware filtering method, aiming at making use of Kinect in 3D telepresence applications. Our method not only can effectively filter Kinect depth data, in particular to resolve the depth quantization issue, but also can run in real-time. Our specific contributions are as follows:

1. The analysis of quantization errors in raw Kinect data and the resulting holes between irregularly-shaped surface patches in the 3D reconstruction;
2. A new filtering technique that smoothes large surface areas while preserving small-scaled surface details; in particular, it can effectively reduce quantization error and filter holes between surface patches;
3. We also devise an efficient CUDA implementation on the GPU; it can perform our filtering method in real-time, thus facilitating 3D telepresence applications.

2. RELATED WORK

2.1 Telepresence Applications with Kinect

Among the various researches that use Kinect, KinectFusion by Newcombe et al. [5] is a state-of-the-art method, which offers efficient high-quality room-sized 3D reconstruction. Since this method employs Kinect as a handheld 3D scanner to acquire depth, it can capture different views of the same object to improve the reconstruction quality. However, this approach is not feasible for our targeted application - *3D telepresence*, because the hardware sensor in our application has to be mounted and fixed in the physical space rather than freely movable. More recently, Maimone and Fuchs [4], [6] proposed an enhanced 3D telepresence system with multiple Kinects. Kuster et al. [7] developed a hybrid camera system for telepresence and 3D video-conferencing using multiple high-quality digital cameras.



(a)Kinect's view ⊥ wall (b) Kinect's view slanted

Figure 1. Color-coded Kinect depth data on a flat wall.

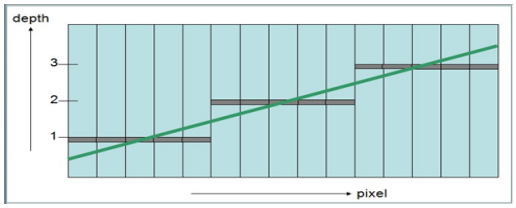


Figure 2. 1D quantized depth with ground truth in green.

2.2 Kinect Depth Filtering

Based on general image denoising approach, Maimone and Fuchs [4] developed a modified two-pass median filter for hole-filling of Kinect depth data. Fu et al. [8] proposed a bilateral filtering framework that integrates both spatial and temporal information in filtering Kinect depth data. Nguyen and Izadi [9] derived an empirical noise model for Kinect sensor, and applied it to improve the filtering quality in KinectFusion [5]. Other than local neighborhood filtering methods, Reisner-Kollmann et al. [10] proposed a view-dependent locally optimal projection method to optimize the point cloud geometry while Liu et al. [11] employed colors to guide the hole-filling and filtering of depth. However, these two methods require huge computational load, making them unsuitable for real-time depth filtering.

In contrast, this work proposes a real-time GPU-based multi-scale filtering method for filtering Kinect depth data. Compared to local methods, our method produces higher-quality 3D reconstruction results, in particular, on large depth-quantized surface patches. Compared to global methods, which could produce higher-quality results by sacrificing the computational performance, our method is local, and so, can run in parallel with real-time performance.

2.3 Scale-space and Multi-scale Analysis

Scale-space theory, or multi-scale analysis, is a popular tool in computer vision and image processing. It dated back to the early work of Witkin [12] who suggested that real-world objects have structures of different scales, thereby motivating the development of computational methods to perform image denoising, e.g., Tai et al. [13]. We approach the Kinect data filtering problem with this multi-scale concept, but consider also the depth quantization orientation (affected by the relative orientation between Kinect viewing direction and surface's normal, see Sections 3.1 and 3.3 for details), which inspires us to devise a novel direction-aware filtering mechanism to further suppress the depth error.

3. OUR APPROACH

3.1 Kinect Raw Depth Data

Kinect sensor is a low-cost depth acquisition device that comes with various kinds of errors, e.g., system error, random noise, and depth quantization. To better understand them, we start with a simple preliminary experiment. First, we use a Kinect sensor to capture a flat white wall with uniform color and illumination at various distances: from 0.5m to 5m, which is the operation range of Kinect. Figure 1 shows color-coded visualizations of two raw depth maps from Kinect; each colored strip in the visualization shows a group of pixels that have the same depth value. When the orientation of Kinect changes with respect to the wall's normal, e.g., from a frontal view as in Figure 1(a) to a slanted view as in Figure 1(b), the orientation of the strips changes relatively (see the black arrows in Figure 1).

In detail, these stripes are caused by Kinect's low precision, or quantization, in acquiring the depth data. Such precision decreases with increases in distances from Kinect: The point spacing along the Kinect's optical axis can be as large as 7cm up to a maximum of 5m. Hence, we often observe large irregularly-shaped patches/stripes (see again Figure 1) in the acquired depth data, where the strip width and orientation could change according to the angle between the surface's normal and the Kinect's viewing direction.

3.2 Multi-scale Filtering

Figure 2 depicts the problem of quantization with a 2D example: the vertical axis is depth while the horizontal axis is the screen. The green line shows the actual 1D surface. After quantizing it into three depth levels (grey), the surface continuity is lost, and we see only three groups of pixels with the same depth similar to the stripes shown in Figure 1.

Traditional filtering methods, in particular local methods, aim at high computational performance. They often use a fixed-size support window to filter all the pixels. The larger the window size, the smoother the surface, but the larger the computational demand and the less the amount of small details that can be preserved. Since image features usually occur at multiple different scales as suggested by the scale-space theory, the optimal scale to analyze different pixels can be very different subject to the local image context. Hence, we employ the multi-scale method to detect an optimal scale for each pixel in the depth map. Note that multi-scale analysis is particularly important for Kinect data since we need to overcome the depth quantization problem.

Figure 3 illustrates depth filtering of two different regions (see 1st row: left and right) on the same 1D depth map that is quantized (grey pixels). For the region on the left, it has three large quantized patches. Here, if we use a small three-pixel-sized filtering window (see 2nd row (left)), the gaps between quantized depth levels are smoothed a bit but it creates certain unnatural bumpiness (red pixels) since the ground truth is actually a straight line (green). This suggests why surfaces reconstructed from Kinect are usually bumpy

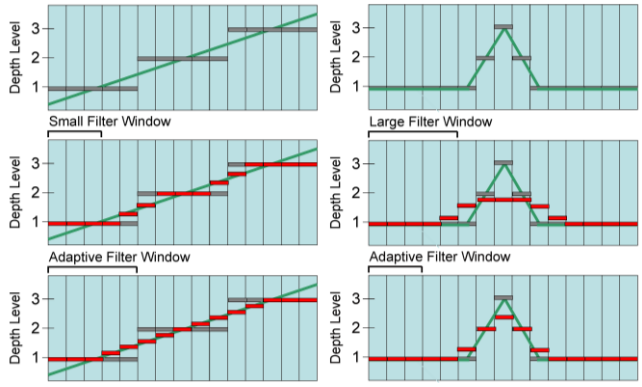


Figure 3. First row: two different regions in the same raw depth data: large patches (left) and a small feature (right). Second Row: applying small (left) and large (right) bilateral filtering windows. Third row: our multi-scale filtering method avoids producing bumpy patches and can better preserve small surface details. Green lines show the ground truth while red pixels are the filtered result.

(see Section 6). These large patches require a larger filtering window. However, for the region shown on the right, it has a small feature; if we use a large window, the detail will be undesirably filtered out (see 2nd row (right)). Our method determines appropriate filtering window sizes for different regions adaptively (see 3rd row), and thus can produce higher-quality depth filtering results, see Section 6. In particular, we suggest that for quantized strips (see Figure 1) the filter window size should be comparable to the strip width; otherwise, undesirable bumpiness would be resulted.

3.3 Direction-aware Filtering

As shown in Figure 1, the orientation of depth quantization (strips) is usually not aligned with screen axes because it depends on the angle between the object surface and Kinect viewing direction. Hence, different surface regions could receive different quantization orientations. Since filtering is fundamentally weighed averaging, the choice and weights of neighbor pixels used in the filtering can greatly affect the filtering quality. Therefore, we propose a direction-aware filtering method (see Section 5), attempting to improve the filtering quality by adjusting the pixel weight contributions based on the orientation of local depth gradient.

4. PROCESSING PIPELINE

Our data processing pipeline has three stages (see Figure 4):

1. **Multi-scale Analysis**: We take a Kinect depth data as input, compute the optimal filter window size (scale) for each pixel based on the statistic analysis around the pixel neighborhood, and output an optimal scale map.
2. **Direction-aware Analysis**: In this stage, we estimate the local predominant orientation using the structure tensor for each pixel at the optimal scale (from the previous step) and output an eigenvector map.

Figure 4. Our data processing pipeline.

3. **Data Filtering**: The two maps are then employed to reconstruct and hole-fill the raw depth map.

5. ALGORITHM

Given a raw depth map from Kinect, say D, we first compute the *optimal scale*, i.e., the filtering window size, of each pixel by [13], and infer the pixel's *local predominant orientation* by analyzing the depth gradient. Then, we use these information to construct the weights in the filtering.

5.1 Multi-Scale Analysis

1. We define a set of discrete scales, i.e., candidate filter window sizes. For each scale, we compute the local depth (Gaussian) distribution of every pixel, say (μ_i, Σ_i), where μ_i and Σ_i are the mean and covariance matrix of the depth distribution at scale *i*.
2. We compute the conformity of each pixel (per scale) by checking how well its depth value is modeled by (μ_i, Σ_i). Here, we use a fixed conformity threshold, say t_{con}; if the conformity is smaller than t_{con} for all scales, the pixel is labeled as *noise*; otherwise, it is labeled as a *salient* pixel.
3. For each salient pixel, we determine the optimal scale as the largest filter size that gives a $|\Sigma_i|$ that is smaller than a fixed covariance threshold, say t_{cov}. If no such scale exists, we label the pixel as *boundary*; else we label it as *region*. The results are aggregated over all pixels and output as an optimal scale map, say S.

5.2 Direction-aware Analysis

Using the optimal scale, we compute for each pixel a structure tensor, which is a matrix representation; its major eigenvector indicates the direction of the largest depth gradient in the local pixel neighborhood. In detail, we have the following steps:

1. For each pixel, we perform the *Difference of Gaussians* method to calculate the partial derivative, or gradient, $\nabla I = (I_x, I_y)$, over its optimal window I, where I_x and I_y are the local partial derivatives along x and y of the image domain, respectively.
2. Then, we construct the structure tensor matrix

$$M = \begin{bmatrix} (I_x)^2 & I_x I_y \\ I_x I_y & (I_y)^2 \end{bmatrix}, \quad (1)$$

and apply eigen-decomposition to M to calculate eigenvectors (e_1, e_2).
3. For each pixel, e_1 is the unit vector that indicates the maximum gradient while e_2 is the local tangent. The output of this step is an eigenvector map E.

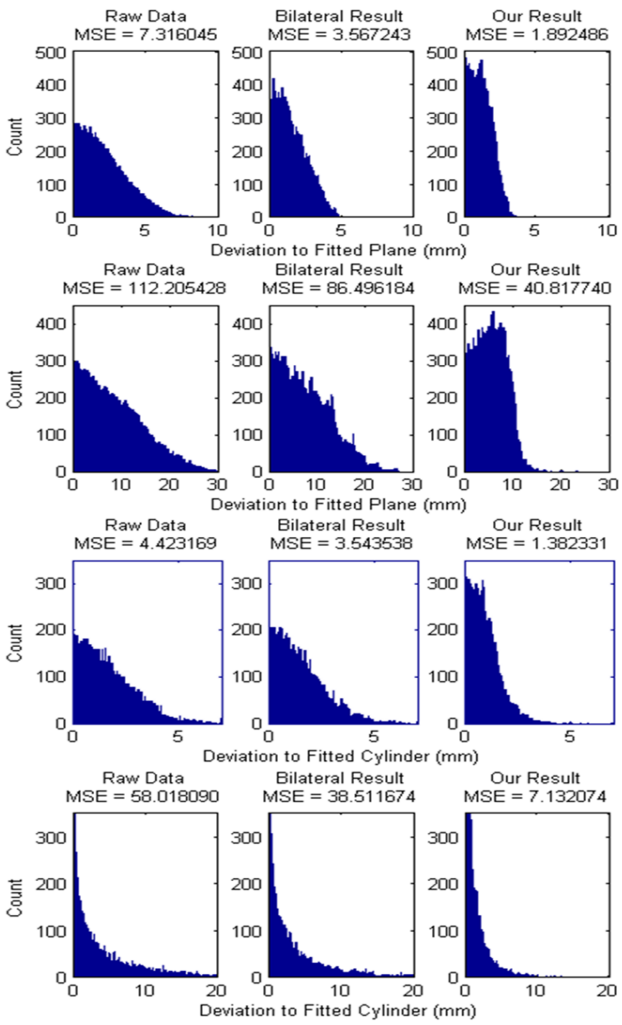


Figure 5. Histograms of depth deviations between point clouds and the fitted surface (ground truth). The window size used in bilateral filtering is 11x11.

5.3 Data Filtering

In this stage, we construct the weights by using S and E, and perform the filtering with the following formulation:

1. For each pixel, say p, we compute its depth similarity, say f_d, against its neighbor pixel q by

$$f_d(p,q) = e^{-\frac{1}{2}\left(\frac{|D(q)-D(p)|}{\sigma_d}\right)^2}, \quad (2)$$

where σ_d is a threshold, which is 9 in practice.

2. Instead of rotating the filter windows to align them with the local depth gradient, we devise an efficient method for direction-aware filtering by giving higher weight to pixels along the eigenvector directions. We define function f_e to measure the directional closeness between pixel p and its neighbor pixel q:

$$f_e(p,q) = e^{-\frac{1}{2}\left(\frac{min\,(v\cdot e_1, v\cdot e_2)}{\sigma_e}\right)^2}, \quad (3)$$

where v is the unit vector from p to q, e_1, e_2 are the eigenvectors of p (from E), and σ_e is a threshold, which is 30 in practice.

3. Lastly, we filter depth map D by

$$D'(p) = \frac{1}{W(p)} \sum_{q \in \Omega(p)} D(q)\, f_d(p,q)\, f_e(p,q), \quad (4)$$

where $W(p)$ is the total sum of weights ($\sum f_d f_e$), and $\Omega(p)$ is the local filter window (from S).

5.4 CUDA Implementation on GPU

Since our filtering method is local, we can develop GPU-based parallel computation methods with CUDA to enable real-time depth filtering. In detail, we assign one CUDA thread to process each pixel in the raw depth data, and group 16x16 threads into one CUDA block. We allocate shared memory per block to store the depth data local to each pixel block (rather than using global memory), so that threads within the same block can access the depth data more efficiently. Since the resolution of Kinect depth data is 640x480, there are altogether 1200 blocks. Moreover, we divide our method into three CUDA kernels since these three kernel programs are interdependent, and have to be executed sequentially one after the other:

1. **Initialization**: First, we copy the raw depth map from host memory to GPU's global memory, and distribute local depth data to the shared memory of each block;
2. **Multi-scale kernel**: Then, we execute the 1st kernel over all threads to compute the optimal scale map; this map is stored in the GPU's global memory;
3. **Direction-aware filtering kernel**: Next, we compute the difference of Gaussian (DoG) using the constant GPU memory (which is cached), and generate the eigenvector map by forming the structure tensor and performing the eigen-analysis [14].
4. **Filtering kernel**: Lastly, we apply the 3rd kernel to perform per-pixel filtering and hole-filing.

6. EXPERIMENTS AND RESULTS

Implementation Our test system runs on 64-bit Windows and employs the OpenNI APIs along with a Kinect driver.

6.1 Quantitative Comparison

To quantitatively evaluate our method, we perform a surface fitting experiment. First, we use a Kinect sensor to capture depth maps of a white flat wall and a white cylindrical column (diameter ~0.8m and height ~2.5m), each at two different distances: 1m and 1.5m. For each of the four captured depth maps, we collect all the 3D raw data points and fit a plane or cylinder over them respectively. Hence, we can consider the fitted plane or cylinder as the estimated ground truth for quantitative comparison, and project each raw data point onto its related fitted surface to compute the deviation (in millimeter) as the acquisition error (per pixel).

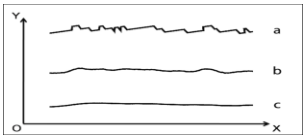


Figure 6. Depth profile on the walls shown on 1st column of Figure 7. (a), (b), and (c) in this figure correspond to raw data, bilateral filtering and our result, respectively.

Figure 5 (left column) shows the histograms of such error for the four raw depth maps. More specifically, the first two rows correspond to the cases of the wall at 1m and 1.5m while the last two rows are for the cases of the cylinder, respectively. In each histogram, the horizontal axis denotes the amount of error in millimeter while the vertical axis presents the related pixel counts. Furthermore, we perform bilateral filtering and our filtering method on the four raw depth maps, and compute also the deviation error of the filtered results against the estimated ground truth, i.e., the fitted planes/cylinders, see the middle and right columns in Figure 5 for the results. From the histograms, we can see that errors of our filtered results are consistently smaller than those of the bilateral filtering and the raw data. The MSE results also confirm the superiority of our method since it produces much smaller MSE values.

6.2 Visual Comparison

To visually assess the quality of a depth map (raw or filtered), we connect the data points into a 3D mesh, and render it with Phong shading. Figure 7 presents the visual comparison results with four different data sets, each in one column. From top to bottom, the first three rows in the figure correspond to raw depth data, bilateral filtered results (related code is extracted from the PCL (Point Cloud Library) implementation of KinectFusion [5]), and results from our method, while the next three rows present corresponding zoom-in views of the results.

Due to depth quantization, we can see from the 1st row of the figure that raw data typically produce large amount of bumpy surfaces, e.g., the sofa (1st column) and the plane (2nd column). Bilateral filtering can smooth out certain amount of bumpiness (see 2nd row), but since it uses only a small filter window (11x11) over the entire data, quantization effect still affects the filtered results. Our approach produces the best result by adaptively picking an appropriate filter size for different regions in the depth data. Hence, we can produce smoother surfaces (3rd row) as compared to bilateral filtering, and at the same time, can also better preserve small-scale details, see and compare the results on the Human and Cloth data sets in 3rd and 4th columns.

To further evaluate the filtering quality of the wall surface shown in the 1st column of Figure 7, we do a depth profiling by intersecting the wall with a horizontal plane. See the red lines in the related sub-figures. The resulting intersection lines help indicate the surface smoothness (see Figure 6). Comparing the results, it is clear that our method can produce the smoothest surface with the least bumpiness.

Table 1. Performance Evaluation Results.

a. Compare CPU and GPU Performance

Stage	CPU (millisec.)	GPU (millisec.)
Multi-Scale Analysis	26968	3.3209
Direction-Aware Analysis	16825	0.5780
Data Filtering	4121	5.1491

b. Compare Bilateral Filter & Our Method

Data Set	Performance on GPU (milliseconds)	
	Bilateral Filter	Our Method
sofa	2.9961	9.0480
ball and box	3.0492	9.7417
human	3.0338	10.1719
cloth	3.0634	8.6158

6.3 Performance Evaluation

We implemented two versions of our method: a CPU version with C++ and a GPU version with CUDA 5.0, and employed a desktop computer with Intel Six Core 3.2GHz CPU and an NVIDIA GeForce GTX 580 1.5GB graphics card. Table 1a presents the time performance statistics and shows that our GPU implementation achieves real-time performance. Table 1b compares the performance of bilateral filter and our method on the four different data sets shown in Figure 7. It shows that our filtered results have higher quality as compared to that from bilateral filtering. Since our method allows real-time filtering, it can be used as a pre-processing component in 3D telepresence applications.

7. CONCLUSION

This paper presents a novel filtering method tailored for filtering raw 3D depth data acquired from Kinect. Since Kinect depth data is heavily quantized, we propose a multi-scale direction-aware filtering method, which is capable of effectively addressing the depth quantization problem with Kinect depth data. Our experimental results show that surfaces reconstructed from our method are closer to the estimated ground truths and are also of higher visual quality as compared to the common filtering method, i.e., bilateral filtering. Moreover, we develop and implement our method with CUDA, and show that it can efficiently run in parallel on the GPU to filter Kinect depth data in real-time. Hence, it is possible to incorporate it to support our targeted 3D telepresence application, which is our future work.

Acknowledgment. This research, which is carried out at BeingThere Centre, is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

8. REFERENCES

[1] R. Raskar, G. Welch, *et al.*, "The office of the future: A unified approach to image-based modeling and spatially immersive displays," in SIGGRAPH, pp. 179-188, 1998.

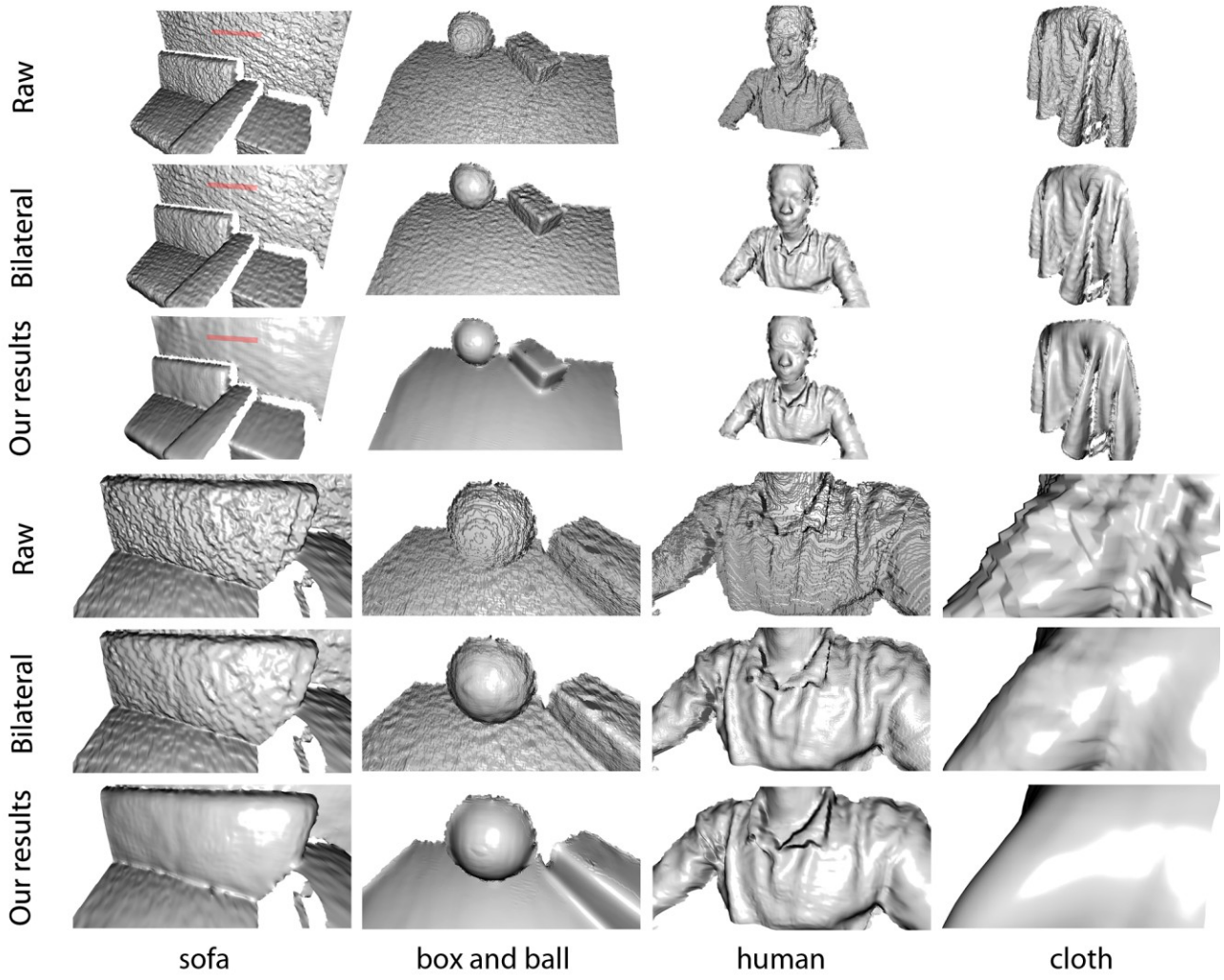


Figure 7. 1st and 4th rows: raw data. 2nd and 5th rows: bilateral-filtered results. 3rd and 6th rows: our results. 4th, 5th, and 6th rows are zoom-in view of 1st, 2nd, and 3rd rows respectively. Red lines shown in the 1st column are the intersection lines used in depth profiling (see Figure 6). Size of neighborhood used in bilateral filtering is 11x11.

[2] M. Gross, S. Würmlin, *et al.*, "blue-c: A spatially immersive display and 3D video portal for telepresence," *ACM Tran. On Graphics (SIGGRAPH),* vol. 22(3), pp. 819-827, 2003.

[3] Microsoft Kinect. http://www.xbox.com/Kinect, 2010

[4] A. Maimone and H. Fuchs, "Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras," in *ISMAR*, pp. 137-146, 2011.

[5] R. A. Newcombe, S. Izadi, et al., "KinectFusion: real-time dense surface mapping and tracking," in *ISMAR*, pp. 127-136, 2011.

[6] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *IEEE VR*, pp. 51-54, 2012.

[7] C. Kuster, T. Popa, *et al.*, "FreeCam: A hybrid camera system for interactive free-viewpoint video," in *VMV*, pp. 17-24, 2011.

[8] J. Fu, S. Wang, *et al.*, "Kinect-like depth denoising," in *ISCAS*, pp. 512-515, 2012.

[9] C.V. Nguyen, S. Izadi, *et al.*, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *3DIMPVT,* pp. 524-530, 2012

[10] I. Reisner-Kollmann and S. Maierhofer, "Consolidation of multiple depth maps," in *ICCV Workshops*, pp.1120-1126, 2011.

[11] J. Liu, X. Gong, *et al.*, "Guided inpainting and filtering for Kinect depth maps," in *ICPR*, pp. 2055-2058, 2012.

[12] A. Witkin, "Scale-space filtering: a new approach to multi-scale description," in *ICASSP*, pp. 150-153, 1984.

[13] Y.W. Tai, W.S. Tong, *et al.*, "Simultaneous image denoising and compression by multiscale 2D tensor voting," in *ICPR,* pp. 818-821, 2006.

[14] B. Jahne, "*Spatio-Temporal Image Processing: Theory and Scientific Applications*," Springer-Verlag, 1993.