Real-time Preprocessing for Dense 3-D Range Imaging on the GPU: Defect Interpolation, Bilateral Temporal Averaging and Guided Filtering

Jakob Wasza[1], Sebastian Bauer[1], Joachim Hornegger[1,2]

[1]Pattern Recognition Lab, Department of Computer Science
[2]Erlangen Graduate School in Advanced Optical Technologies (SAOT)
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
jakob.wasza@cs.fau.de

Abstract

Recent advances in range imaging (RI) have enabled dense 3-D scene acquisition in real-time. However, due to physical limitations and the underlying range sampling principles, range data are subject to noise and may contain invalid measurements. Hence, data preprocessing is a prerequisite for practical applications but poses a challenge with respect to real-time constraints. In this paper, we propose a generic and modality-independent pipeline for efficient RI data preprocessing on the graphics processing unit (GPU). The contributions of this work are efficient GPU implementations of normalized convolution for the restoration of invalid measurements, bilateral temporal averaging for dynamic scenes, and guided filtering for edge-preserving denoising. Furthermore, we show that the transformation from range measurements to 3-D world coordinates can be computed efficiently on the GPU. The pipeline has been evaluated on real data from a Time-of-Flight sensor and Microsoft's Kinect. In a run-time performance study, we show that for VGA-resolution data, our preprocessing pipeline runs at ∼100 fps on an off-the-shelf consumer GPU.

1. Introduction

Recently, several range imaging (RI) sensor technologies have been introduced that allow for metric 3-D surface acquisition at high resolutions (up to 307k points) and real-time frame rates (up to 60 Hz). In particular, Time-of-Flight (ToF) based devices [13] and Microsoft's low-cost Kinect [7] are of special interest and popularity. Among others, these real-time capable RI sensors are currently deployed in controller-free gaming in consumer electronics and hold potential for biometric face recognition [3] or pedestrian detection in automotive industry [18]. Furthermore, the deployment of RI technologies in medical engineering is subject to current research with a broad range of applications such as fractionated radiotherapy [19] or image guided liver surgery [15]. However, for the majority of these applications, direct usage of the raw data obtained from RI sensors is not feasible as the underlying sampling principles and physical limitations lead to sensor noise and may involve unreliable measurements. With ToF imaging, erroneous range values typically result from overexposed and saturated sensor elements caused by specular reflections or so-called flying pixels at sharp object boundaries [13]. For the Kinect sensor principle, difficulties arise due to illumination issues, when capturing reflective or transparent objects, and in regions that are not covered by the infra-red projector pattern. These effects are depicted in Fig. 1 for Kinect data, for ToF issues see Fig. 4. Consequently, as a

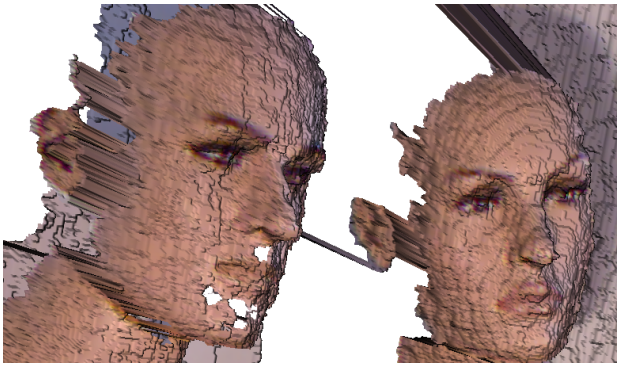

Figure 1. Close-up view of a 3-D scene showing triangulated raw 3-D Kinect data. Note the ubiquitous staircase artifacts and the missing regions around the male's chin and nose that result from illumination issues. For the large white areas no depth measurements exist as the corresponding regions were not covered by the infrared projector pattern. Here, this was the case as the sensor was positioned in front of the mannequins.

2011 IEEE International Conference on Computer Vision Workshops
978-1-4673-0063-6/11/$26.00

prerequisite for subsequent data analysis, denoising and enhancement of RI data is a key component. However, the immense amount of data in the scale of 500 Mbit/s poses a challenge for preprocessing and analysis algorithms.

In this paper, we show that real-time constraints for the task of RI data preprocessing can be faced by exploiting the inherent degree of parallelism in the employed algorithms for implementation on the graphics processing unit (GPU) using NVIDIA's CUDA architecture. GPUs, which were originally used exclusively for visualization purposes, have evolved into powerful co-processors for general purpose computing. Compared to a CPU, a much larger portion of GPU resources is devoted to data processing than to caching or flow control, increasing throughput and reducing computation time. Today's massively parallel GPU hardware architectures yield the best performance for algorithms that exhibit high data parallelism and high arithmetic intensity, respectively. Therefore, we propose efficient GPU implementations of normalized convolution for restoring invalid depth measurements, bilateral temporal averaging for dynamic scenes, and guided filtering for edge-preserving denoising. As another element, we demonstrate that the transformation from range measurements to 3-D world coordinates can be performed efficiently on the GPU and seamlessly integrates into the preprocessing pipeline.

2. Related Work

In the field of image processing and computer vision, a variety of denoising approaches have been introduced in recent years. Here, edge-preserving filters for smoothing homogeneous regions while preserving manifest discontinuities are of special interest and importance. One of the most popular and established methods is the bilateral filter [2, 20]. Beyond its application for a multitude of conventional imaging modalities, it is a common choice for RI data denoising [14]. The filter is straightforward to implement, but exhibits a poor run-time performance due to its non-linear nature. Recent algorithmic acceleration concepts have attempted to overcome the inherent computational complexity by quantization and approximation techniques, however with the drawback of impairing accuracy [16, 17, 23, 24]. In contrast, the concept of guided filtering [9] is based on a non-approximative algorithm with a computational complexity that is independent of the filter kernel size. At the same time, it exhibits a comparable degree of edge-preserving smoothing and does not suffer from gradient reversal artifacts.

Besides algorithmic acceleration concepts, GPUs assume a prominent role for high-performance data processing. Chen et al. demonstrated real-time framerates for high-definition video processing by accelerating the bilateral grid on the GPU [4]. Furthermore, GPUs were successfully deployed for accelerating denoising and resolution enhancement of RI data, however, without explicitly addressing real-time constraints [10].

Prior to denoising of RI data, the restoration of invalid measurements has to be taken into consideration. In contrast to static defect pixels, these invalid measurements occur unpredictable and can affect both an isolated pixel or connected local regions. In the literature, a plurality of methods for defect pixel correction have been proposed. Given the tradeoff between effectiveness and complexity, conventional approaches like normalized convolution [6, 12] provide satisfying results. For applications with more challenging demands, defect pixel interpolation in the spectral domain may be employed [1].

3. Range Image Preprocessing Pipeline

We propose a generic and modality-independent preprocessing pipeline for range data streams obtained from RI devices (Fig. 2). Range data preprocessing is performed in the 2-D sensor domain. The pipeline setup is motivated by the observation that independent of the underlying physical causes and characteristics, erroneous range measurements can be roughly grouped into three categories: (i) missing or invalid information, (ii) temporal noise and (iii) sensor noise or quantization issues. Consequently, we restore invalid depth measurements (Sec. 3.1) as a first step. This renders an extra conditioning of invalid data unnecessary for subsequent algorithms. Second, we propose a bilateral temporal averaging scheme for dynamic scenes (Sec. 3.2). Third, we perform edge-preserving denoising using the recently introduced guided filtering technique (Sec. 3.3). In contrast to a joint approach combining the abovementioned steps, this modular setup has the advantage that individual modules can be disabled or replaced by algorithms tailored to a specific RI device. Furthermore, the integration of additional processing modules is enabled by design [11, 22].

Note that we do not incorporate additional photometric information such as RGB data into the preprocessing algorithms. Even though this might enhance results for certain scenes and illumination conditions, such additional information is not granted to be available for RI sensors and thus contradicts a modality-independent pipeline.

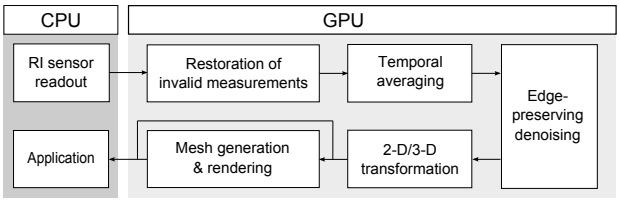

Figure 2. Flowchart of our proposed modality-independent RI data preprocessing pipeline. Note that solely the sensor readout is performed on the CPU.

Nomenclature Below, $g(\boldsymbol{x})$ denotes the corrupted input range image, $f(\boldsymbol{x})$ the restored and denoised output. The variable $\boldsymbol{x} = (x, y)$ represents a position within the sensor image matrix, N being the total number of pixels. We further introduce $\omega_{\boldsymbol{x}}$ as the set of coordinates in a local quadratic subimage window (neighborhood) of size $|\omega_{\boldsymbol{x}}| = (2r+1)^2$, centered at position $\boldsymbol{x}$. In terms of filters, the window radius r denotes the kernel size.

3.1. Restoration of Invalid Measurements

In this work, we adopt the concept of normalized convolution (NC) proposed by Knutsson and Westing [12]. We use the method for restoring missing or corrupted depth measurements. As a prerequisite, in addition to the range data $g(\boldsymbol{x})$, we assume the availability of a binary mask image $m(\boldsymbol{x})$ holding information whether a pixel is valid $m(\boldsymbol{x}) = 1$ or invalid $m(\boldsymbol{x}) = 0$. The availability of such a mask image can be taken for granted for the sensors employed in this work. ToF sensors directly provide a scalar reliability indicator regarding flying pixels or saturated and overexposed sensor elements. The mask image for the Kinect sensor can be obtained by checking the range measures for zero entries. For different sensors, software based methods have to be applied. Consequently, the restoration of invalid depth measurements can be formulated as:

$$f_{\text{NC}}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}' \in \omega_{\boldsymbol{x}}} g(\boldsymbol{x}')\, a(\boldsymbol{x}, \boldsymbol{x}')\, m(\boldsymbol{x}')}{\sum_{\boldsymbol{x}' \in \omega_{\boldsymbol{x}}} a(\boldsymbol{x}, \boldsymbol{x}')\, m(\boldsymbol{x}')}, \tag{1}$$

where the applicability function $a(\boldsymbol{x}, \boldsymbol{x}')$ is given as a Gaussian:

$$a(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{\sigma_s^2}\right). \tag{2}$$

Here, the parameter σ_s controls the spatial similarity.
As valid pixels are supposed to remain unchanged we rewrite equation (1) and express the actual filter output as:

$$\widetilde{f}_{\text{NC}}(\boldsymbol{x}) = g(\boldsymbol{x})\, m(\boldsymbol{x}) + f_{\text{NC}}(\boldsymbol{x})\, (1 - m(\boldsymbol{x}))\,. \tag{3}$$

3.2. Temporal Averaging for Dynamic Scenes

The averaging of successive frames is a common denoising technique in range imaging. However, this inherently introduces blurring artifacts in dynamic scenes. We therefore propose a constrained temporal averaging (TA) technique based on the concept of bilateral filtering [20]. The basic idea is that range gradients along the temporal dimension are typical for dynamic scenes and that averaging across theses edges falsifies the output. For a series of T temporally successive frames $\{g_0(\boldsymbol{x}), \ldots, g_{T-1}(\boldsymbol{x})\}$ with the subscript denoting the relative temporal shift, the filter output is given as:

$$f_{\text{TA}}(\boldsymbol{x}) = \frac{\sum_{i=0}^{T-1} g_i(\boldsymbol{x})\, c(i)\, s(g_0(\boldsymbol{x}) - g_i(\boldsymbol{x}))}{\sum_{i=0}^{T-1} c(i)\, s(g_0(\boldsymbol{x}) - g_i(\boldsymbol{x}))}. \tag{4}$$

Here, the temporal closeness $c(i)$ and range similarity $s(g_0(\boldsymbol{x}) - g_i(\boldsymbol{x}))$ are in analogy to the Gaussian in Eq. 2 with parameters σ_c and σ_s to control the temporal extent and level of dynamic scene preservation, respectively.

3.3. Edge-Preserving Denoising

For the purpose of high performance edge-preserving denoising, we employ the non-approximative guided filtering (GF) technique as proposed by He et al. [9]. The guided filter output $f_{\text{GF}}(\boldsymbol{x})$ can be deduced as the following linear model:

$$f_{\text{GF}}(\boldsymbol{x}) = \text{E}_{\omega_{\boldsymbol{x}}}(A(\boldsymbol{x})) \cdot i(\boldsymbol{x}) + \text{E}_{\omega_{\boldsymbol{x}}}(B(\boldsymbol{x}))\,. \tag{5}$$

Here, $i(\boldsymbol{x})$ is the guidance image that, without loss of generality, can be the input image $g(\boldsymbol{x})$ itself. $\text{E}_{\omega_{\boldsymbol{x}}}$ denotes the expectation value of a uniformly distributed random variable in a local quadratic region $\omega_{\boldsymbol{x}}$ centered at $\boldsymbol{x}$. $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$ can be found as:

$$A(\boldsymbol{x}) = \frac{\text{Cov}_{\omega_{\boldsymbol{x}}}(i(\boldsymbol{x}), g(\boldsymbol{x}))}{\text{Var}_{\omega_{\boldsymbol{x}}}(i(\boldsymbol{x})) + \epsilon}, \tag{6}$$

$$B(\boldsymbol{x}) = \text{E}_{\omega_{\boldsymbol{x}}}(g(\boldsymbol{x})) - A(\boldsymbol{x}) \cdot \text{E}_{\omega_{\boldsymbol{x}}}(i(\boldsymbol{x}))\,, \tag{7}$$

where ϵ is a regularization parameter that controls the edge-preserving functionality. $\text{Cov}_{\omega_{\boldsymbol{x}}}$ and $\text{Var}_{\omega_{\boldsymbol{x}}}$ denote the covariance and variance, respectively. Again, the subscript $\omega_{\boldsymbol{x}}$ indicates that only a local region is considered. For two random variables X and Y the covariance is given as $\text{Cov}(X, Y) = \text{E}(XY) - \text{E}(X)\,\text{E}(Y)$. Thus, the guided image filter can be formulated solely in terms of expectation values which can be computed by using mean filters. Mean filtering techniques that exhibit an algorithmic complexity of $\mathcal{O}(N)$ such as integral images [5, 21] eventually provide the basis for a filter run-time that is decoupled from the kernel size.

4. Implementation Details

In this section we outline important implementation details for porting the presented algorithms to the graphics card using the CUDA architecture. We aim to exploit the GPU's single instruction multiple data (SIMD) architecture by one single kernel and the assignment of one thread to each pixel in the input data $g(\boldsymbol{x})$. This scheme naturally arises from the mathematical definitions of the normalized convolution (Eq. 1) and bilateral temporal averaging (Eq. 4). In contrast, implementing the linear-time algorithm of the guided filter in one single kernel is not feasible, as a complex infrastructure in terms of intermediate image representations and sequential workflows is required (Eqs. 5,6,7). However, each of these subtasks can again be computed efficiently using the SIMD concept.

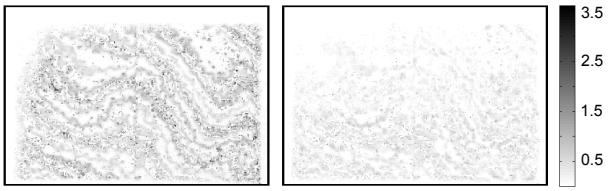

Figure 3. Erroneous filter output (error in mm) for guided filtering using standard integral images (left) and our modified approach using separated mean filter (right) compared to double-precision CPU results. The test scene shows a plane at a distance of 1 m in front of the camera.

4.1. Performance Aspects

The summation over a local neighborhood $\omega_{\boldsymbol{x}}$ in the definition of the normalized convolution (Eq. 1) entails non-coalesced memory access patterns. Non-coalesced memory access is a crucial performance bottleneck when porting algorithms to the CUDA architecture. Therefore, we bind the input data as a texture which is a common strategy to cope with read-only and non-coalesced memory access. We further note that the applicability function $a(\boldsymbol{x}, \boldsymbol{x}')$ (Eq. 2) and the temporal closeness $c\,(i)$ (Eq. 4) rely on a Gaussian that is independent of the actual input data. Therefore, we precompute these values and hold them in a look-up-table.

4.2. Numerical Issues

As the majority of CUDA capable GPUs do not support double-precision floating-point numbers, special care has to be taken with respect to numerical issues. In particular, we observed a strong impact for efficient mean filtering using integral images as proposed by He et al. for linear-time guided filtering [9]. Due to the accumulative nature of integral images and the range of values of single-precision floating-point numbers this may lead to erroneous results as depicted in Fig. 3. To lessen this problem, we exploit the separability of 2-D mean filtering and perform two successive 1-D mean filtering operations along both image axes. 1-D mean filtering can again be performed independent from the kernel size by using cumulative sums that can be computed efficiently on the GPU using the parallel prefix sum technique [8].

5. Experiments and Results

In this section we evaluate the preprocessing modules described in Sec. 3 on real data acquired with a PMD CamCube 3.0 ToF sensor and a Microsoft Kinect. First, we analyze the run-time performance of our GPU filter implementations. Afterwards, we present qualitative results for the two RI modalities.

5.1. Run-time Performance Study

The run-time study in this paper is conducted on an off-the-shelf consumer desktop computer running an NVIDIA GeForce GTX 285 GPU and an AMD Phenom II X4 920 CPU. Table 1 depicts the run-times for the individual preprocessing modules. We have investigated the overall mean run-times (including software or CPU overhead) and the actual GPU times neglecting such overhead. Please note that GPU times are considerably smaller than the overall run-times, however being hard to achieve in practice. We thus focus on the overall run-time in the benchmarks below.

As the sensor readout is performed on the CPU, we first evaluate the transfer time of range data from the CPU (host) to the GPU (device). For the employed system, we measured transfer times of 1.09 ms (1.13 GB/s) and 0.23 (0.70 GB/s) for Kinect ($640 \cdot 480 \cdot 4$ bytes) and ToF ($200 \cdot 200 \cdot 4$ bytes) data, respectively.

The Run-times for the restoration of invalid measurements using the proposed normalized convolution concept depend on the actual scene as the filter is applied for invalid pixels only (Eq. 3). For comparison purposes and in order to establish an upper run-time boundary for a worst-case scenario, we apply the filter for all pixels, however, solely replacing invalid measurements. Using this approach, a maximum run-time for the restoration of invalid measurements is 4 ms for Kinect and 2 ms for ToF data, respectively.

The run-times of 1.5 ms (0.5 ms) for Kinect (ToF) data for our bilateral temporal averaging method ($T = 40$ frames) demonstrate that algorithms featuring a high computational complexity can be implemented very efficiently on the GPU. Its runtime increases approximately linearly when incrementing the number of frames to be considered. However, experimental tests show that even for $T = 200$ frames a real-time satisfying run-time of 7 ms for Kinect data and 2 ms for ToF data is feasible.

Despite its complex infrastructure in terms of intermediate image representations and sequential workflows, our GPU implementation of the linear-time guided filtering algorithm achieved run-times of approximately 2 ms for Kinect data and 1 ms for ToF data, respectively. Again, we emphasize that the run-time is independent from the kernel size, thus allowing arbitrary large kernels without performance losses. This is of particular importance for high-resolution range data as obtained from the Kinect device.

We also investigated the performance for transforming range measurements to 3-D world coordinates on the GPU. The standard pinhole camera model fits perfectly to the SIMD paradigm. Our implementation achieved run-times of 0.3 ms for Kinect and 0.2 ms for ToF data.

For the final transfer of the filtered range data and the corresponding 3-D world coordinates from the device to the host, we measured a transfer-time of 1.4 ms and 0.5 ms for Kinect and ToF data, respectively. However, note that

Step	Kinect (GPU)	Kinect (overall)	Kinect (percentage)	CamCube (GPU)	CamCube (overall)	CamCube (percentage)
Host/Device Transfer	0.66	1.09 ± 0.07	10.7%	0.11	0.23 ± 0.01	05.4%
Normalized Convolution	2.92	4.00 ± 0.06	39.2%	0.80	1.88 ± 0.05	44.2%
Bilateral Temporal Averaging	1.30	1.49 ± 0.04	14.6%	0.27	0.48 ± 0.04	11.3%
Guided Filtering	1.31	1.93 ± 0.21	18.9%	0.46	0.97 ± 0.01	22.8%
2-D/3-D Transformation	0.17	0.34 ± 0.01	03.3%	0.04	0.20 ± 0.03	04.7%
Device/Host Transfer	1.20	1.39 ± 0.02	13.6%	0.33	0.49 ± 0.07	11.5%
Total	**7.56**	**10.2**	**100%**	**2.01**	**4.25**	**100%**

Table 1. Run-time performance evaluation for the Kinect (640×480 px) and the CamCube 3.0 (200×200 px). Run-times are given in [ms] and are averaged over 100 frames. The supplement 'GPU' denotes pure GPU times neglecting software and CPU overhead. Normalized convolution employs a 15×15 kernel and bilateral temporal averaging is performed with $T = 40$ frames.

this step can be omitted if subsequent post-processing algorithms run on the GPU, too.

Given a total run-time of ∼10 ms for Kinect data, the pipeline runs at ∼100 fps. Comparing this achieved runtime to the maximum frame rate of the Kinect sensor (30 fps), we explicitly note that residuary GPU resources can be devoted to postprocessing tasks, for instance.

5.2. Experiments on Real Data

Results of our preprocessing pipeline for the Kinect sensor and the CamCube are given in Fig. 4. Note that for facilitating the visual interpretation of filter effects, the studies are conducted on a triangulated 3-D mesh computed from the preprocessed and transformed range data. Furthermore, for the same reason, the restoration of invalid measurements is illustrated after bilateral temporal averaging and guided filtering. However, recall that the restoration is actually performed as the first step in the preprocessing pipeline (see Fig. 2).

As expected, our bilateral temporal averaging method and edge-preserving denoising in the 2-D sensor domain significantly increases the smoothness of the mesh surface while retaining distinctive surface contours (Fig. 4c,d). Here, we note that ToF data still exhibits minor fluctuations after bilateral temporal averaging of $T = 40$ frames, whereas for Kinect data the effect of temporal noise is considerably smaller and a steady surface could be established by averaging less than $T = 10$ frames.

In contrast to ToF data, edge-preserving denoising is a delicate problem with Kinect range data due to staircase quantization artifacts present in the raw data. Thus, the filter parameters have to be chosen properly such that these artifacts are not classified erroneously as representative topological features being subject to preservation. For our experiments, we have chosen the parameters in a way that gives priority to the smoothness of the surface.

The restoration of invalid measurements using normalized convolution effectively interpolates missing regions in Kinect data (Fig. 4e) and removes spikes due to specular reflections in the ToF data (Fig. 4f). However, for ToF data, minor artifacts remain as the sensor's built-in validity mechanism fails to recognize unreliable measurements at the boundaries of invalid regions.

6. Discussion and Conclusion

In this work, we have proposed and evaluated a GPU based and modality-independent preprocessing pipeline for dense 3-D range image streams. The pipeline is built in a modular fashion and is explicitly designed to enable real-time preprocessing of high resolution range data. For the Kinect device, featuring the highest RI resolution at real-time frame rates available to date, we have shown that preprocessing can be performed at ∼100 fps on an off-the-shelf GeForce GTX 285 consumer GPU. This is a promising result with regard to next-generation RI sensors that will feature even higher resolutions. Furthermore, these low run-times enable the usage of residuary GPU resources for postprocessing and visualization tasks. To our knowledge, this work is the first study covering a real-time preprocessing pipeline for RI sensors that entirely runs on the GPU. Though the experiments conducted on real data demonstrated the fitness of the proposed algorithms, further investigations concerning alternative real-time capable preprocessing modules will be subject of our upcoming research.

Acknowledgments

J. Wasza and S. Bauer gratefully acknowledge the support by the European Regional Development Fund and the Bayerisches Staatsministerium für Wirtschaft, Infrastruktur, Verkehr und Technologie, in the context of the R&D program IuK Bayern under Grant No. IUK338.

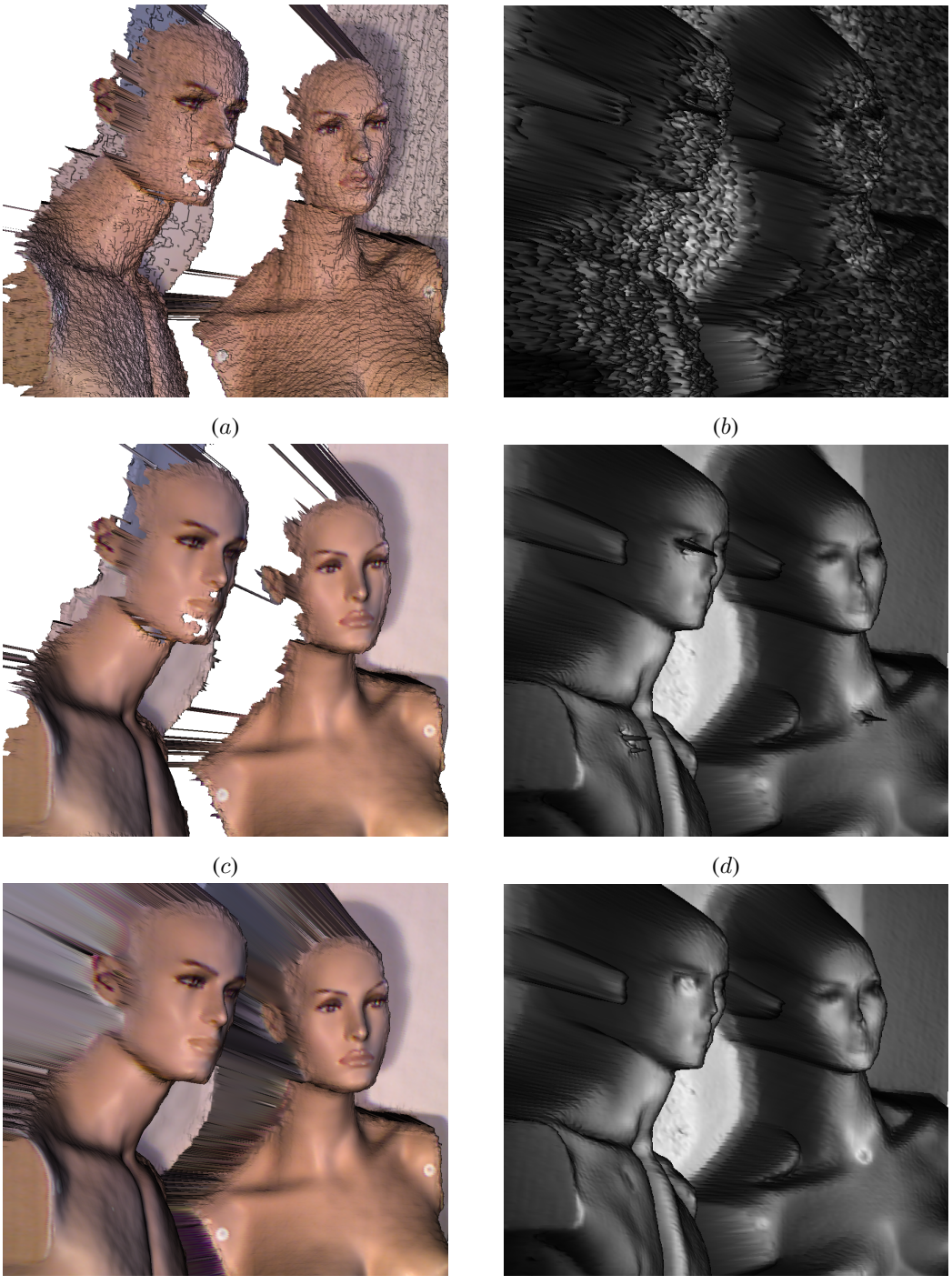

Figure 4. Triangulated 3-D Kinect data with RGB texture overlay (left column) and ToF data with intensity texture overlay (right column). The raw kinect data (a) shows staircase artifacts and holes due to missing range information. Note that the raw ToF data (b) exhibits an exceptional low signal-to-noise ratio (SNR). Bilateral temporal averaging and guided filtering effectively produce a steady and smooth surface for Kinect data (c) and significantly increases the SNR for ToF data (d). Performing normalized convolution for the restoration of invalid measurements as the first step in the preprocessing pipeline interpolates missing regions for Kinect data (e) and removes spikes being characteristic for specular reflections in ToF imaging (f).

References

[1] T. Aach and V. Metzler. Defect Interpolation in Digital Radiography - How Object-Oriented Transform Coding Helps. In *Proc. SPIE Medical Imaging*, volume 4322, pages 824–835, Feb 2001.

[2] V. Aurich and J. Weule. Non-Linear Gaussian Filters Performing Edge Preserving Diffusion. In *Proc. DAGM*, pages 538–545. Springer, 1995.

[3] S. Bauer, J. Wasza, K. Müller, and J. Hornegger. 4D Photogeometric Face Recognition with Time-of-Flight Sensors. In *Proc. IEEE Workshop on Applications of Computer Vision*, 2011.

[4] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26, 2007.

[5] F. C. Crow. Summed-area tables for texture mapping. In *Proc. SIGGRAPH*, pages 207–212. ACM, 1984.

[6] M. Frank, M. Plaue, and F. A. Hamprecht. Denoising of Continuous-Wave Time-Of-Flight Depth Images using Confidence Measures. *Optical Engineering*, 48(7), Jul 2009.

[7] J. Garcia and Z. Zalevsky. Range mapping using speckle decorrelation. US patent No.7433024, 2008.

[8] M. Harris, S. Sengupta, and J. D. Owens. Parallel Prefix Sum (Scan) with CUDA. In *GPU Gems 3*. Addison Wesley, Aug 2007.

[9] K. He, J. Sun, and X. Tang. Guided Image Filtering. In *Proc. European Conference on Computer Vision: Part I*, volume 6311, pages 1–14. Springer, 2010.

[10] B. Huhle, T. Schairer, P. Jenke, and W. Straßer. Fusion of range and color images for denoising and resolution enhancement with a non-local filter. *Computer Vision and Image Understanding*, 114:1336–1345, Dec 2010.

[11] F. Jargstorff. A framework for image processing. In *GPU Gems*. Addison Wesley, 2004.

[12] H. Knutsson and C.-F. Westin. Normalized and Differential Convolution: Methods for Interpolation and Filtering of Incomplete and Uncertain Data. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 515–523, Jun 1993.

[13] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-Flight Sensors in Computer Graphics. In *Eurographics 2009 - State of the Art Reports*, pages 119–134. Eurographics, Mar 2009.

[14] M. Lindner, I. Schiller, A. Kolb, and R. Koch. Time-of-Flight sensor calibration for accurate range sensing. *Computer Vision and Image Understanding*, 114(12):1318 – 1328, 2010. Special issue on Time-of-Flight Camera Based Computer Vision.

[15] K. Müller, S. Bauer, J. Wasza, and J. Hornegger. Automatic Multi-modal ToF/CT Organ Surface Registration. In *Proceedings of Bildverarbeitung für die Medizin*, pages 154–158, 2011.

[16] S. Paris and F. Durand. A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach. *International Journal of Computer Vision*, 81:24–52, Jan 2009.

[17] F. Porikli. Constant time O(1) bilateral filtering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –8, Jun 2008.

[18] M. Rapus, S. Munder, G. Baratoff, and J. Denzler. Pedestrian recognition using combined low-resolution depth and intensity images. In *IEEE Intelligent Vehicles Symposium*, pages 632–636, Jun 2008.

[19] P. J. Schöffel, W. Harms, G. Sroka-Perez, W. Schlegel, and C. P. Karger. Accuracy of a commercial optical 3D surface imaging system for realignment of patients for radiotherapy of the thorax. *Physics in Medicine and Biology*, 52(13):3949–3963, Jul 2007.

[20] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. IEEE International Conference on Computer Vision*, pages 839–846, 1998.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518, 2001.

[22] J. Wasza, S. Bauer, S. Haase, M. Schmid, S. Reichert, and J. Hornegger. RITK: The range imaging toolkit - a framework for 3-D range image stream processing. In *Proceedings of International Workshop on Vision, Modeling, and Visualization*, Oct 2011, accepted for publication.

[23] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time o(1) bilateral filtering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 557 –564, Jun 2009.

[24] S. Yoshizawa, A. Belyaev, and H. Yokota. Fast gauss bilateral filtering. *Computer Graphics Forum*, 29:60–74(15), 2010.