Fast robust detection of edges in noisy depth images

Wei Liu,[a,b] Xiaogang Chen,[c] Qiang Wu,[d] and Jie Yang[a,b,*]
[a]Shanghai Jiao Tong University, Institute of Image Processing and Pattern Recognition, 800 Dongchuan Road, Minhang District, Shanghai 200240, China
[b]Key Laboratory of System Control and Information Processing, Ministry of Education, 800 Dongchuan Road, Minhang District, Shanghai 200240 China
[c]University of Shanghai for Science and Technology, College of Communication and Art Design, 516 Jun Gong Road, Shanghai 200093, China
[d]University of Technology Sydney, School of Computing and Communications, P.O. Box 123, Broadway, New South Wales 2007, Australia

Abstract. Depth edges play an important role in depth image upsampling. Many recent upsampling methods rely on the prior images of depth edges to preserve sharp depth edges in restored depth images. However, recent depth edge detection methods are not robust against the noise in depth images. Some methods are also too time-consuming. We propose a method to efficiently detect edges in depth images. The proposed method is very simple but very robust against the noise in depth images. It is also fast and has near $\mathcal{O}(1)$ implementation. We apply the proposed method to the existing edge guided depth image upsampling. Experimental results on both simulated and real data show the effectiveness of the proposed method. *© 2016 SPIE and IS&T* [DOI: 10.1117/1.JEI.25.5.053003]

Keywords: robust edge detection; depth image; depth upsampling.
Paper 16301 received Apr. 10, 2016; accepted for publication Aug. 10, 2016; published online Sep. 6, 2016.

1 Introduction

Acquisition of depth information of three-dimensional (3-D) scenes is essential for many applications in computer vision and graphics. Applications range from 3-D modeling to 3-DTV and augmented reality. A number of applications require accurate and high-resolution depth images, for instance, object reconstruction, robot navigation, and automotive driver assistance. Recently, modern time-of-flight (ToF) depth cameras such as SwissRanger SR4000 have shown impressive results and become increasingly affordable. They can obtain dense depth measurements at a high frame rate. However, their depth images are usually quite noisy and suffer from low resolution.

To facilitate the use of depth data, tremendous efforts have been spent on upsampling depth images obtained by modern depth cameras.[1–5] One of the most challenging issues in the upsampling is to preserve depth edges in depth images. To this end, many methods rely on the prior of depth edges.[6–11] These methods mainly focus on how to get a high-quality depth edge map given a noisy low-resolution depth image and an aligned color image (if the color image is available). Here, depth edges mean abrupt depth changes in depth images (also known as depth discontinuities in related papers[6,12]). To get the depth edges, these methods can be mainly categorized into three kinds: (1) detecting edges on both the depth image and the aligned color image using traditional edge detectors such as Canny detector.[13] The obtained edge maps are then combined using certain rules.[6,9,10] (2) Detecting depth edges only using depth images.[8] (3) Learning edge maps with an external database.[7,11] The first kind of method is only suitable to the situation where aligned color images are available. The last two kinds of methods are suitable to the situation where there are no color images. However, most of these methods are not robust against the noise in depth images. Some of them[6,9] can even introduce false depth edges. The learning-based methods are also quite time-consuming. How to get high quality depth edge maps in a robust and fast way still remains a challenging issue in edge guided depth upsampling.

In this paper, we propose a method for robust detection of edges in noisy depth images which can be used to guide the edge guided depth upsampling. While many previous methods are only tested on simulated depth images without any noise which is not true for real depth images, we show our method is robust enough and can work well on real data that contain heavy noise. Moreover, we propose the near $\mathcal{O}(1)$ implementation of the proposed method. This makes the proposed method more practical for real applications. We apply the proposed method to the existing edge guided depth upsampling. Experimental results on both simulated and real data show the promising performance of the proposed method.

The rest of this paper is organized as follows: in Sec. 2, we briefly present some recent methods that detect depth edges. In Sec. 3, we show the proposed robust detection of depth edges and its near $\mathcal{O}(1)$ implementation. Experimental results of both simulated and real data are shown in Sec. 4. We also discuss the parameters in the proposed method. We draw the conclusion of this paper in Sec. 5.

2 Related Work

To detect edges in depth images, one way is to use a traditional edge detector such as the Canny detector.[13] Schwarz et al.[10] proposed to first detect edges in a depth image and the corresponding aligned color image using the Canny detector. Second, they combined these two edge maps to get the final edge map. A similar method was also adopted by Camplani et al.[6] and Liu et al.[9] These methods have two problems: (1) when the noise is too heavy in depth images, which is

*Address all correspondence to: Jie Yang, E-mail: jieyang@sjtu.edu.cn

true for ToF depth images, the Canny detector may result in a noisy depth edge map. (2) The final depth edge map has many "false depth edges" that are in fact color edges.

Another way is detecting depth edges only in depth images. Nguyen et al.[14] proposed a depth dependent threshold to detect depth edges in depth images. Hua et al.[8] proposed to detect depth edges by thresholding the absolute difference between the minimal value and the maximal value within depth patches. These methods are only suitable for noise free depth images or depth images containing little outliers (e.g., Kinect depth images). When depth images contain large outliers (e.g., ToF depth images), these methods do not work well.

There are learning-based methods that detect depth edges. Ferstl et al.[7] proposed to learn a dictionary of edge priors from an external database of high- and low-resolution examples. They used a sparse coding approach to precalculate edge priors out of low resolution examples. Xie et al.[11] proposed to learn the high-resolution edge map from the edge map of the upsampled low-resolution depth image and an external database by solving a Markov random field (MRF) framework. These methods need quite a long time to get the final depth edge map. The method proposed by Xie et al.[11] is also not robust against the noise because they used the Canny detector[13] to detect edges in the upsampled low-resolution depth image to initialize their framework.

3 Proposed Method

3.1 *Depth Edges Based on the Relative Smoothness*

A better depth edge detector should be robust against the noise and should be efficient to implement. It should also not rely on color images that can introduce irrelevant false depth edges (i.e., color edges). First, we should know the unique property of depth images: Most regions in depth images are quite smooth. Only object boundaries have depth edges in depth images. These can make the detection of edges in depth images in a different manner from that of natural images which contain many textures. Another point is that real depth images also contain large outliers.[15] To make use of the unique property of depth images and be robust against the noise, we propose to measure the smoothness of depth images as follows:

$$\lambda_i = \frac{\sum_{s \in N_\lambda(x_{\min})} x_s}{\sum_{t \in N_\lambda(x_{\max})} x_t}, i \in \Omega, \tag{1}$$

where $x_{\min}$ and $x_{\max}$ are the minimal and maximal depth values within the given depth patch $N(i)$, respectively, $N(i)$ is a square patch that has the radius of r, $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$ denote a small square patch of radius r_λ centered at the pixel which has the value of $x_{\min}$ and $x_{\max}$, and Ω represents the coordinate set of the depth image. λ_i is denoted as the "relative smoothness" of $N(i)$ in this paper. If $N(i)$ is located in a homogeneous depth region, its relative smoothness λ_i will be close to 1. If $N(i)$ contains a depth edge, its relative smoothness λ_i will be much smaller than 1. Note that Eq. (1) does not depend on the depth range of the depth image since we always have $\lambda_i \in (0,1]$.

There are two aspects behind the intuition of Eq. (1): (1) Since depth images are quite smooth except for depth edges, depth values inside a small patch are always very close. (2) For a noisy depth image, the depth value of a single pixel may be an outlier. However, if we use the sum of a set of pixels that have the same "ground truth" depth value, then the noise will be suppressed. This is approximated by using the sum inside $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$ based on the first intuition.

In fact, both $x_{\min}$ and $x_{\max}$ will be noisy points since depth images are noisy. We analyze the correctness of Eq. (1) given $x_{\min}$ and $x_{\max}$ are noisy points. First, if the patch $N(i)$ in Eq. (1) locates on homogeneous regions, then no matter whether $x_{\min}$ and $x_{\max}$ are noisy or not, $\sum_{s \in N_\lambda(x_{\min})} x_s$ will always be close to $\sum_{t \in N_\lambda(x_{\max})} x_t$ based on the intuition above. Then λ_i will be close to 1. This is what we suppose it to be. Second, if the patch $N(i)$ contains a depth edge, then on average, depth values on one side of the edge are smaller than that on the other side. In this case, $x_{\min}$ is more likely to be located on the side with smaller depth values while $x_{\max}$ is located on the other side. This is also what we suppose it to be. There are also cases where both $x_{\min}$ and $x_{\max}$ are located on one side or $x_{\min}$ is located on the side with larger depth values while $x_{\max}$ is located on the other side. However, the probability is quite small. Our experimental results in the experimental part also validate this where depth edges are properly detected.

In real applications, there may be the case where there is more than one pixel that correspond to the minimal/maximal value. In this case, we only choose one of them. For the pixel located on the border area where only part of its neighbor pixels are valid pixels in a depth image, we only use the valid ones to compute the minimal/maximal value.

After we have made all these clear, we show how to measure the relative smoothness of the whole depth image. To this end, we measure the relative smoothness of every patch in the depth image. Then we get a relative smoothness map Λ that has the same size as the depth image. Each element in Λ is λ_i computed through Eq. (1). Then only a small fraction of elements in Λ that correspond to depth edges will have small values and the rest will be close to 1 and will correspond to homogeneous regions in depth images.

To classify depth edges and homogeneous depth regions, we simply threshold every element λ_i in Λ. If the edge map is denoted as E, then for each element $E_i (i \in \Omega)$, it can be obtained as

$$E_i = \begin{cases} 0, & \text{if } \lambda_i \geq \theta \\ 1, & \text{if } \lambda_i < \theta \end{cases}. \tag{2}$$

In fact, Hua et al.[8] and Chan et al.[16] proposed similar methods to measure the smoothness of depth images. They proposed to measure the smoothness of a local depth patch by using the absolute difference between the minimal value and the maximal value within the patch. That is $\Delta = |x_{\min} - x_{\max}|$ where $x_{\min}$ and $x_{\max}$ are the same as those in Eq. (1). Their work computes the depth difference based on a single pixel. On the contrary, we use the sum of pixel values within the patch. This patch is centered at the pixel which has the minimal/maximal depth value. Our method is more robust against noise. Usually, real depth images are quite noisy, such as the ones captured by SwissRanger SR4000. As a result, the difference between

the minimum and maximum can be quite large even for a smooth depth region. The first row of Fig. 1 shows a comparison between the smoothness map of a noisy depth image obtained by the method of Hua et al.[8] and our method. For convenience, both of the smoothness maps are normalized into interval [0, 1]. The corresponding edge maps are shown in the second row of Fig. 1. The edge map in Fig. 1(a2) is obtained by thresholding the smoothness map in Fig. 1(a1) with 5% of the depth range, which is proposed by Hua et al.[8] As shown in the figure, both the smoothness map in Fig. 1(a1) and the edge map in Fig. 1(a2) are much more noisy than ours in Figs. 1(b1) and 1(b2).

3.2 *From $\mathcal{O}(r^2)$ Computational Cost to Near $\mathcal{O}(1)$ Computational Cost*

The main computational cost of our method is Eq. (1). The bruteforce implementation of Eq. (1) has $\mathcal{O}(r^2)$ computing complexity, where r is the radius of $N(i)$ in Eq. (1). In this section, we show that Eq. (1) can be efficiently implemented in near $\mathcal{O}(1)$ time. The computational cost of Eq. (1) has two parts: the minimal/maximal value search inside $N(i)$ and the sum inside $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$. We first explain the near $\mathcal{O}(1)$ implementation of the minimal/maximal value search inside $N(i)$.

First, we decompose the min/max operation inside the square support $N(i)$ into a vertical min/max operation followed by a horizontal min/max operation. These two operations can both be implemented within a one-dimensional (1-D) data array as shown in Fig. 2(a). If the radius of $N(i)$ is r, then $N(i)$ can be decomposed into $2r+1$ column vectors which have the length of $2r+1$. We use $r=3$ for the illustration in Fig. 2(a). We then compute the minimal/maximal value of these column vectors separately. We denote this step as the vertical min/max operation. The outputs are denoted as $x_{\min}^{(-r)}/x_{\max}^{(-r)} \cdots x_{\min}^{(r)}/x_{\max}^{(r)}$. Then these $2r+1$ values can again form a row vector that has a length of $2r+1$: $[x_{\min}^{(-r)}/x_{\max}^{(-r)}, \ldots, x_{\min}^{(r)}/x_{\max}^{(r)}]$ as shown in Fig. 2(a). We then compute the minimal/maximal value of this row vector. The result are denoted as $x_{\min}$ and $x_{\max}$ which are the minimal and maximal value within $N(i)$. We denote this step as the "horizontal min/max operation".

For a given depth image, we first perform the vertical min/max operation along each column of the depth image. After this step, we can obtain a column minimal/maximal map which has the same size as the original depth image. Each value on this map is the minimum/maximum along the column direction. Then we perform the horizontal min/max operation along each row of this newly obtained column minimal/maximal map. After this step, we obtain the minimal/maximal value map which also has the same size as the original depth image. Each value at pixel i on this map is the minimal/maximal value within $N(i)$. As both the vertical and

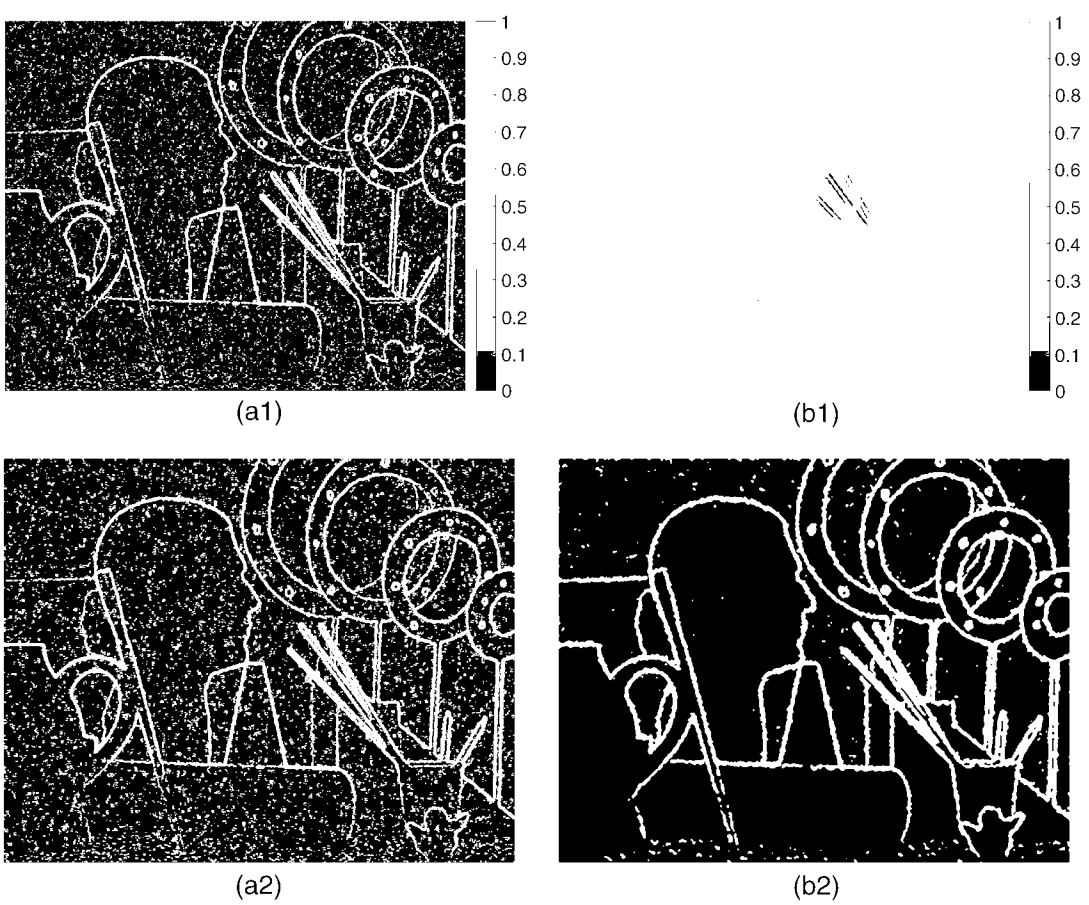

Fig. 1 Smoothness map of (a1) the method of Hua et al.[8] and (b1) our method. (a2) The corresponding edge map of (a1). (b2) The corresponding map of (b1).

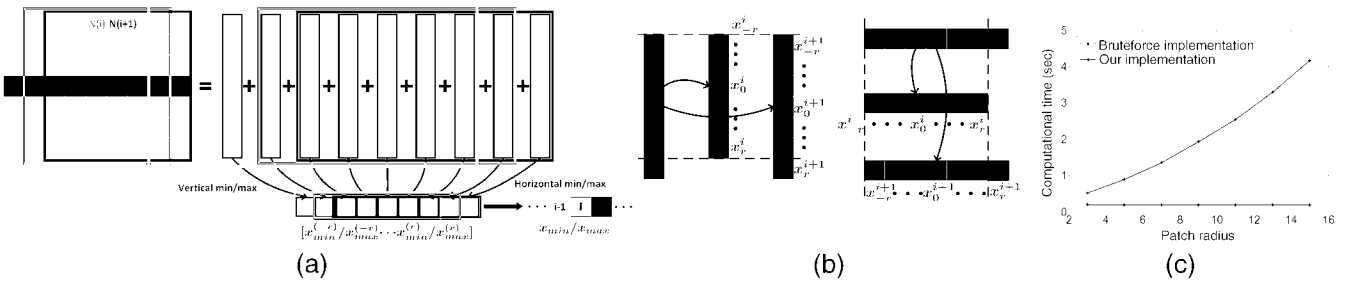

Fig. 2 (a) The min/max operation inside a square patch can be decomposed into a vertical min/max operation followed by a horizontal min/max operation. (b) When the central pixels of two 1-D supports are neighbors, most elements of these two supports are the same (the red ones). Only the green one and the blue one on two sides are different. (c) The computational time of the bruteforce min/max implementation versus the proposed near $\mathcal{O}(1)$ implementation.

horizontal min/max operation have $\mathcal{O}(r)$ computing complexity, the computational cost of the minimal/maximal value search in Eq. (1) decreases from $\mathcal{O}(r^2)$ to $\mathcal{O}(r)$.

Second, due to the reason that the vertical or horizontal min/max operation slides from one pixel to the next pixel, there is a strong correlation between the minimal/maximal values of different supports whose central pixels are neighbors. As shown in Fig. 2(b), the 1-D support of radius r centered at i is denoted as $\tilde{N}(i)$. The pixels in $\tilde{N}(i)$ are denoted as $x^i_{-r},\ldots,x^i_0,\ldots,x^i_r$. Then $\tilde{N}(i)$ and $\tilde{N}(i+1)$ have many common pixels: $x^i_{-r+1}=x^{(i+1)}_{-r},\ldots,x^i_r=x^{(i+1)}_{r-1}$ as shown in Fig. 2(b). The only differences are x^i_{-r} in $\tilde{N}(i)$ and x^{i+1}_r in $\tilde{N}(i+1)$. If the minimal value of $\tilde{N}(i)$ is $x^i_{\min}$, then the minimal value $x^{i+1}_{\min}$ of $\tilde{N}(i+1)$ can be computed as

$$x^{i+1}_{\min}=\begin{cases} x^i_{\min}, & \text{if } x^i_{\min}\le x^{i+1}_r \text{ and } x^i_{\min}\neq x^i_{-r} \\ x^{i+1}_r, & \text{if } x^{i+1}_r < x^i_{\min} \\ \min\{x^{i+1}_{-r},\ldots,x^{i+1}_r\}, & \text{otherwise} \end{cases}. \tag{3}$$

The maximal value of $\tilde{N}(i+1)$ can be computed in a similar way with the direction of inequality signs changed. Equation (3) takes full advantage of the neighbor support. For the first two cases in Eq. (3), the computational complexity is $\mathcal{O}(1)$ which is independent of the patch size. According to our experimental results, there are quite a few opportunities to encounter the third case in Eq. (3). In this way, we further reduce the computational cost of the minimal/maximal value search in Eq. (1) from $\mathcal{O}(r)$ time to near $\mathcal{O}(1)$. Figure 2(c) shows the computational cost of our proposed method versus the bruteforce implementation. The experiment is performed on noisy depth images of size 1088×1376 pixels. It is clear that the computational time of the proposed near $\mathcal{O}(1)$ min/max implementation seldom increases as the radius of the patch increases.

For the sum inside $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$, we can use the summed-area tables technique[17] which has exact $\mathcal{O}(1)$ implementation complexity. The summed-area tables technique has inspired many variants for exact $\mathcal{O}(1)$ implementation in the field of computer vision.[18–20] In our method, we can first compute the sum inside every patch that has a radius of r_λ using the summed-area tables technique. Then we obtain a map that has the same size as the depth image. Each element on this map corresponds to the sum inside the patch of radius r_λ that is centered at the corresponding coordinate in the depth image. This map is denoted as S. Then after the minimal/maximal value search is finished, we can get the corresponding coordinates of the minimum and maximum. According to the coordinates, we can find that $S(x_{\min})$ and $S(x_{\max})$ in S correspond to the sum inside $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$. At last, we compute the relative smoothness according to Eq. (1). We summarize our method in Algorithm 1.

As the minimal/maximal value search inside $N(i)$ has near $\mathcal{O}(1)$ implementation and the sum inside $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$ has exact $\mathcal{O}(1)$ implementation, Eq. (1) thus has near $\mathcal{O}(1)$ implementation.

4 Experiments and Parameter Discussion

In this section, the proposed method is tested on noise-free and noisy simulated ToF data and real ToF data that contain heavy noise. We compare our method with the Canny

Algorithm 1 Robust near $\mathcal{O}(1)$ depth edge map computing.

Input: D (depth map), r [radius of $N(i)$], r_λ [radius of $N_\lambda(x_{\min})$ and $N_\lambda(x_{\min})$], θ (threshold).

Output: E (depth edge map)

1. Initialize minimal/maximal values of the first $2r+1$ pixels in each column of D.
2. Use the initialization in step 1 and Eq. (3) and perform vertical min/max operation along each column of D. Output M_C.
3. Initialize minimal/maximal values of the first $2r+1$ pixels in each row of M_C.
4. Use the initialization in step 3 and Eq. (3) and perform horizontal min/max operation along each row of D. Output M. M is the map of minimal/maximal value inside each $N(i)$.
5. Compute the sum in each square patch of radius r_λ in D using the summed-area tables technique. Output S.
6. Compute relative smoothness map Λ using M and S based on Eq. (1).
7. Compute edge map E using Λ and θ based on Eq. (2).

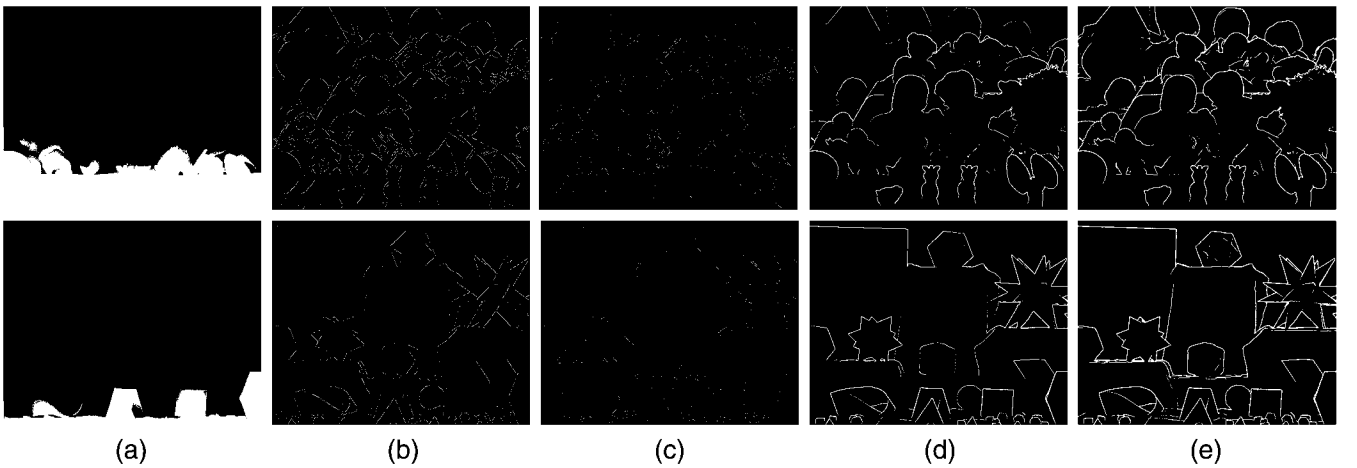

Fig. 3 Edge detection results of 2× upsampling. (a) The upsampled noise-free depth images. Edge maps obtained by (b) Canny detector,[13] (c) learning-based method proposed by Xie et al.,[11] (d) the method proposed by Hua et al.,[8] and (e) ours.

detector[13] that only detects depth edges on the depth map, the method proposed by Hua et al.,[8] and the learning-based method proposed by Xie et al.[11] We show both visual comparisons and computational time comparisons. We first show the comparison between depth edge maps obtained by different methods. Then we use the obtained depth edge maps to guide the upsampling process proposed by Xie et al.[11] and compare the upsampling results both qualitatively and quantitatively. Since the edge maps that guide the upsampling process need to be of the same size as the upsampled depth images, we thus first upsample input depth images to the target size through bicubic interpolation and then perform edge detection. In this way, we can get depth edge maps which have the same size as the upsampled depth images. This strategy is also adopted by Xie et al.[11] and Hua et al.[8]

Parameters of our method are set as follows: the radius r of $N(i)$ to measure the relative smoothness in Eq. (1) is set as 1 for noise-free simulated data and 4 for both noisy simulated data and real data. The radius r_λ of $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$ in Eq. (1) are set as 1 for noise-free simulated data and 5 for both noisy simulated data and real data. The threshold θ in Eq. (2) is set as 0.98 for noise-free simulated data and 0.97 for both noisy simulated data and real data.

4.1 *Experiments on Simulated and Real Data*

We use the Middleburry dataset[21] for noise-free simulated ToF data test. Two upsampling factors are tested: 2× and 4×. Figures 3 and 4 show the results. As shown in the figures, all the compared methods have similar results. Depth edges are well detected by these methods. Depth edge maps obtained by using the ground truth are shown in Fig. 8.

To further test the proposed method, we perform experiments on the noisy simulated ToF data that are used by Yang et al.[22] Two upsampling factors are tested: 2× and 4×. Results are shown in Figs. 5 and 6. We set the threshold of the method proposed by Xie et al.[11] a little higher than their default threshold to make the results less noisy. As shown in the figures, their results are quite noisy and

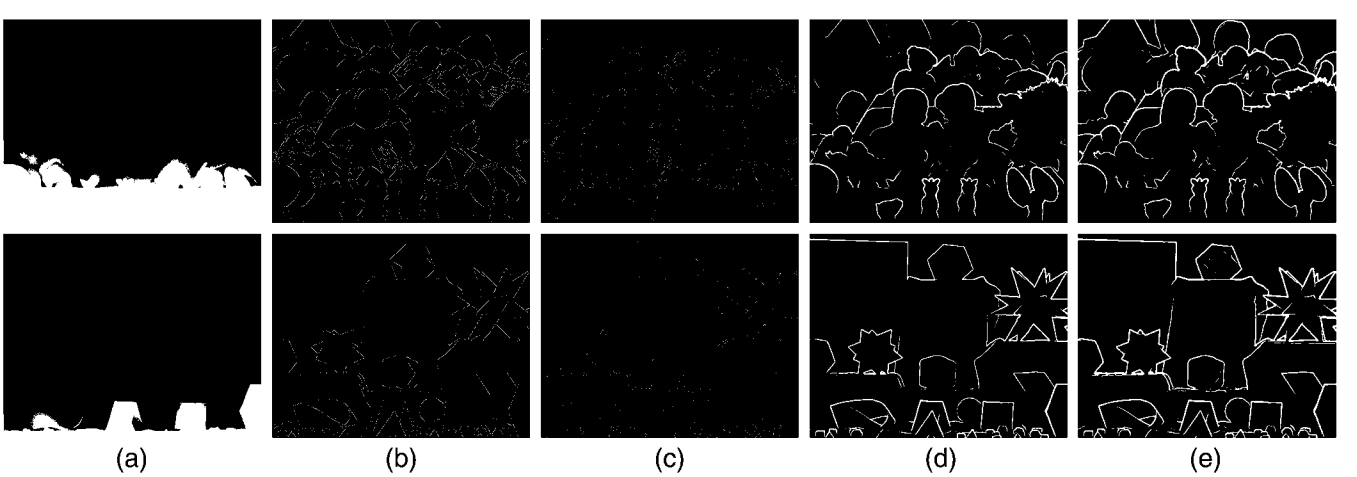

Fig. 4 Edge detection results of 4× upsampling. (a) The upsampled noise-free depth images. Edge maps obtained by (b) Canny detector,[13] (c) learning-based method proposed by Xie et al.,[11] (d) the method proposed by Hua et al.,[8] and (e) ours.

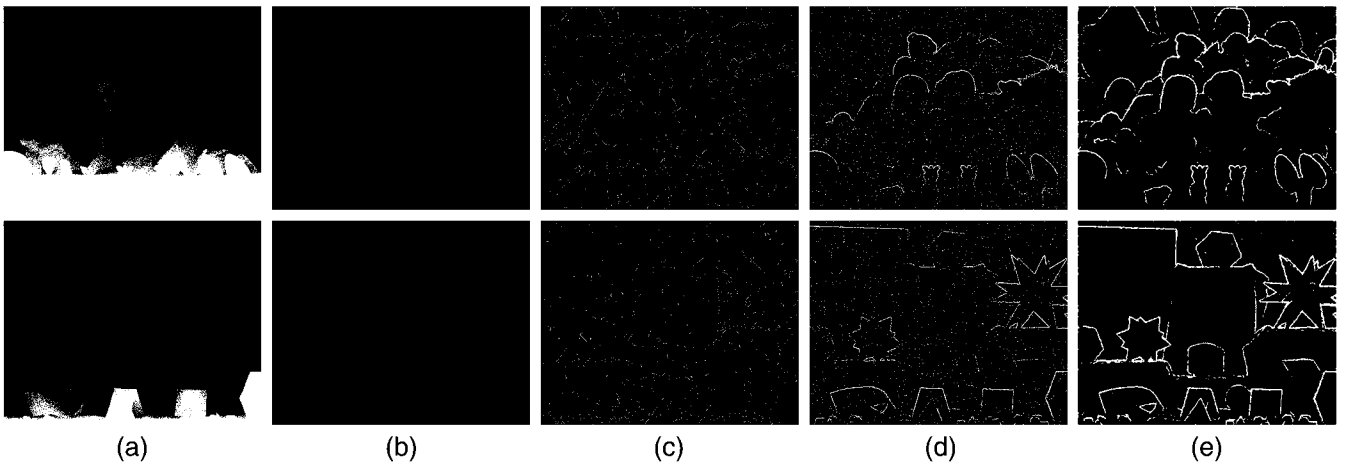

Fig. 5 Edge detection results of 2× upsampling. (a) The upsampled noisy depth images. Edge maps obtained by (b) Canny detector,[13] (c) learning-based method proposed by Xie et al.,[11] (d) the method proposed by Hua et al.,[8] and (e) ours.

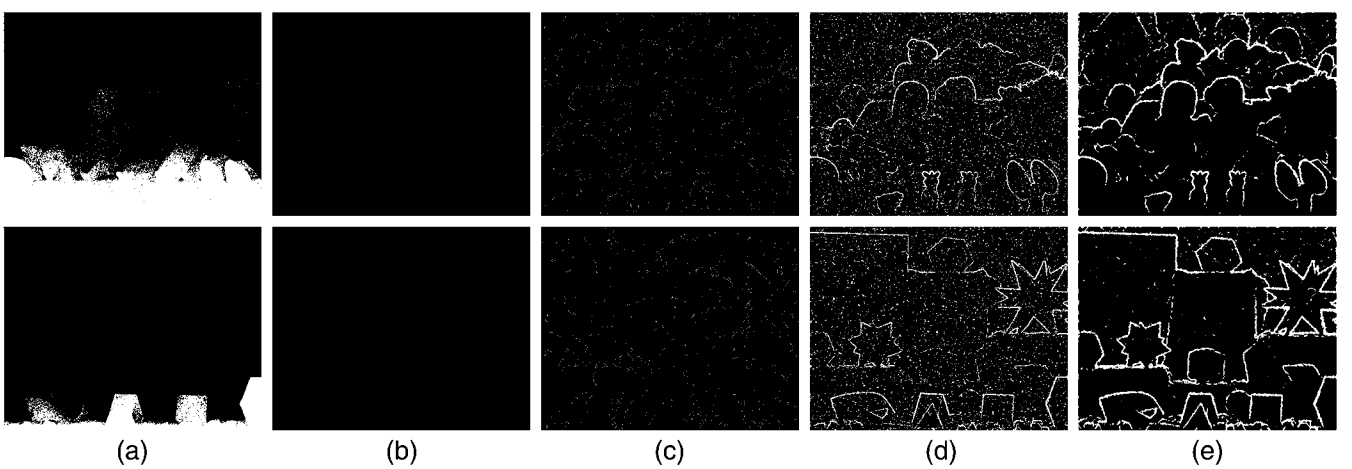

Fig. 6 Edge detection results of 4× upsampling. (a) The upsampled noisy depth images. Edge maps obtained by (b) Canny detector,[13] (c) learning-based method proposed by Xie et al.,[11] (d) the method proposed by Hua et al.,[8] and (e) ours.

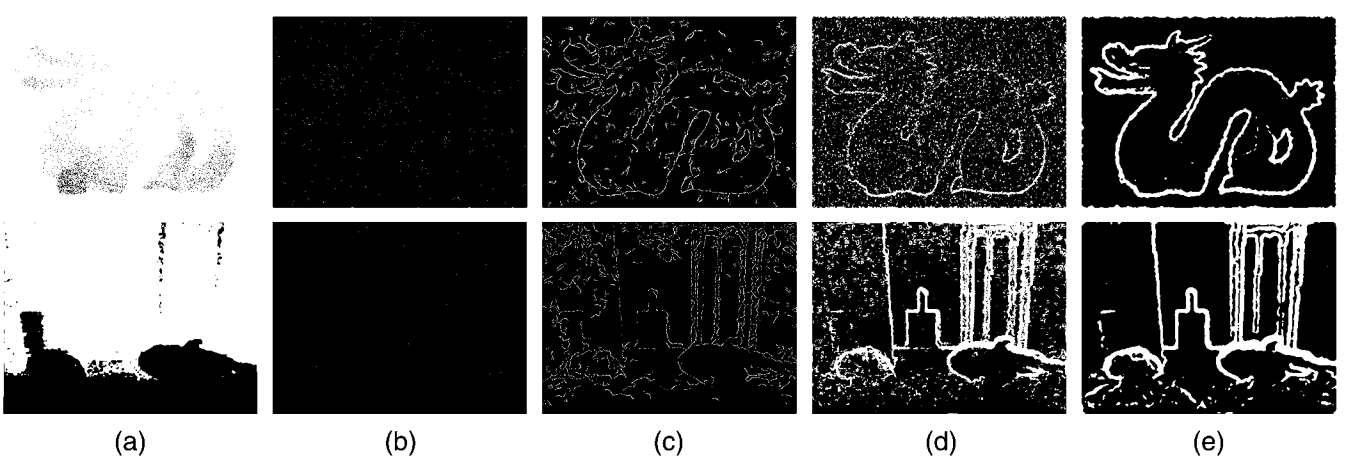

Fig. 7 Edge detection results of real data. (a) The upsampled noisy depth images. Edge maps obtained by (b) Canny detector,[13] (c) learning-based method proposed by Xie et al.,[11] (d) the method proposed by Hua et al.,[8] and (e) ours.

Fig. 8 Edge detection results on the ground truth of depth images in Figs. 3–7.

Table 1 Average computational time (in seconds) comparison of different methods on noise-free and noisy simulated data and real data.

	Canny[13]	Xie et al.[11]	Hua et al.[8]	Ours
Noise-free data (s)	0.187	576.3	0.69	0.93
Noisy data (s)	0.187	2645.5	0.69	0.93
Real data (s)	0.057	311.9	0.096	0.12

some depth edges are still not detected. On the contrary, our method can well detect depth edges and the results are also quite clean. Depth edge maps obtained by using the ground truth are shown in Fig. 8.

We also test our method on real data as shown in Fig. 7. The depth image in the first row is from the Human 3.6 M dataset.[23] The depth image in the second row is from the dataset published by Ferstl et al.[15] As shown in the figure, both these two real depth images contain heavy noise. As a result, the results of the compared methods are quite noisy while our results are much less noisy and depth edges are also properly detected. Depth edge maps obtained by using the ground truth are shown in Fig. 8.

To further validate the effectiveness of our method, we show the computational time comparison in Table 1. The simulated depth images have the size of 1088×1376 pixels. The real depth image in the first row of Fig. 7 has the size of 480×640 pixels. The real depth image in the second row of Fig. 7 has the size of 610×810 pixels. Computational time is evaluated on the platform of Intel i5-4200M CPU with 8 GB memory. For fair comparison, we also use the near $\mathcal{O}(1)$ implementation proposed in Sec. 3.2 to accelerate the minimal/maximal value search in the method of Hua et al.[8] As shown in the table, our method needs a little more time than the Canny detector[13] and the method of Hua et al.[8] However, it is much faster than the method of Xie et al.[11] As the method of Xie et al.[11] needs MRF inference using loop belief propagation,[24] their method is quite time-consuming. Moreover, their computational time is sensitive to the noise level which is different from the other methods. This is because they used the results of the Canny detector to initialize their MRF framework. High level noise makes the Canny detector result in more noisy edge maps which make the MRF inference more time-consuming.

We also use the obtained depth edge maps to perform the edge guided depth image upsampling. We use the code published by Xie et al.[11,25] We perform the comparison by using different depth edge maps to guide the depth upsampling process proposed by Xie et al.[11] Tables 2 and 3 show the upsampling results of both simulated and real ToF data. Results of simulated data are evaluated in terms of root of mean square errors (RMSE) and percentage of error pixels between the result and the ground truth. Results of real data are evaluated in terms of mean absolute errors (MAE) which is adopted by Ferstl et al.[15] who published the data. As shown in tables, our method outperforms the other two methods in most cases. Though results of Xie et al.[11] are comparable with ours, our method uses much less time than theirs as shown in Table 1. Figures 9 and 10 also show visual comparisons. As illustrated in the highlighted region, our result is less noisy than the others especially for the real data. Our result is especially better than the result of Hua.[8] This is due to the fact that our edge maps are much less noisy than theirs.

4.2 *Parameters Discussion*

The key parameters in the proposed method are the r_λ in Eq. (1) and the θ in Eq. (2). r_λ is the radius of $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$. A larger r_λ makes the relative smoothness

Table 2 Experimental results on simulated data. Results are evaluated in terms of RMSE (left) and percentage of error pixels (right) between the result and the ground truth. The best results are in bold.

	Art				Book				Dolls				Laundry				Moebius				Reindeer			
	2×		4×		2×		4×		2×		4×		2×		4×		2×		4×		2×		4×	
Xie et al.[11]	1.58	**16.37**	1.08	**5.06**	0.89	14.57	0.56	2.6	0.88	15.6	0.54	**2.16**	1.05	13.12	0.72	5.55	0.87	14.39	0.55	**3.17**	1.26	**15.49**	0.86	**2.94**
Hua et al.[8]	1.55	17.19	1.08	8.37	0.91	15.3	0.58	4.21	0.91	16.36	0.56	4.9	1.04	13.6	0.71	6.01	0.89	15.07	0.55	4.61	1.27	16.28	0.83	5.23
Ours	**1.54**	16.7	**1.04**	5.94	**0.88**	**14.5**	**0.53**	**2.49**	**0.87**	**15.53**	**0.52**	3.24	**1.02**	**12.62**	**0.68**	**4.64**	**0.86**	**14.33**	**0.51**	3.24	**1.25**	15.64	**0.79**	3.35

Table 3 Experimental results on real data. Results are evaluated in terms of MAE between the result and the ground truth. The best results are in bold.

	Books	Devil	Shark	Dragon
Xie et al.[11]	16.39	16.79	17.93	0.36
Hua et al.[8]	17.72	18.08	19.27	0.46
Ours	**16.26**	**16.75**	**17.8**	**0.33**

λ_i in Eq. (1) more robust against the noise. θ in Eq. (2) is the threshold for the relative smoothness to decide whether there is a depth edge in $N(i)(i \in \Omega)$ or not. In our experiments, we find the size of the support $N(i)(i \in \Omega)$ can also affect the final depth edge map. Thus, we also discuss the relationship between its radius r and the final depth edge map.

Figure 11 shows visual comparison of depth edge maps obtained with different r_λ and r when θ is fixed ($\theta = 0.97$). There are two observations: (1) As r_λ becomes larger, the final depth edge map is more robust against the noise. However, the final depth edge map is more likely to lose weak depth edges. When $r_\lambda > r$, the tendency to lose weak depth edges becomes even more severe. This is because when r_λ becomes larger, $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$ are more likely to overlap each other. Thus, λ will be close to 1 and will be larger than the preset threshold θ. This results in the corresponding element $E_i = 0$ on the depth edge map. For the case of the pixels around weak depth edges, even when there is no overlap between $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$, λ is close to 1 because the depth values on the two sides of the weak depth edges are close to each other. When there is overlap between $N_\lambda(x_{\min})$ and $N_\lambda(x_{\max})$, λ will be even closer to 1 and be larger than θ. Then the corresponding element E_i on the depth edge map will be incorrectly assigned to 0. Based on the analysis above, the final depth edge map more easily loses weak depth edges. (2) As r becomes larger, more pixels around depth edges are classified as depth edges. Theoretically, all the pixels whose distances to the depth edge are within r should be classified as elements of the depth edge. Thus, the number of pixels classified as depth edges is proportional to r. The larger r is, the more pixels are classified as depth edges. Based on the observation above, the choice of r and r_λ depend on the noise in real application. The greater the noise is, the larger r and r_λ should be.

Figure 12 shows the visual comparison of depth edges maps obtained with different θ when both r_λ and r are fixed

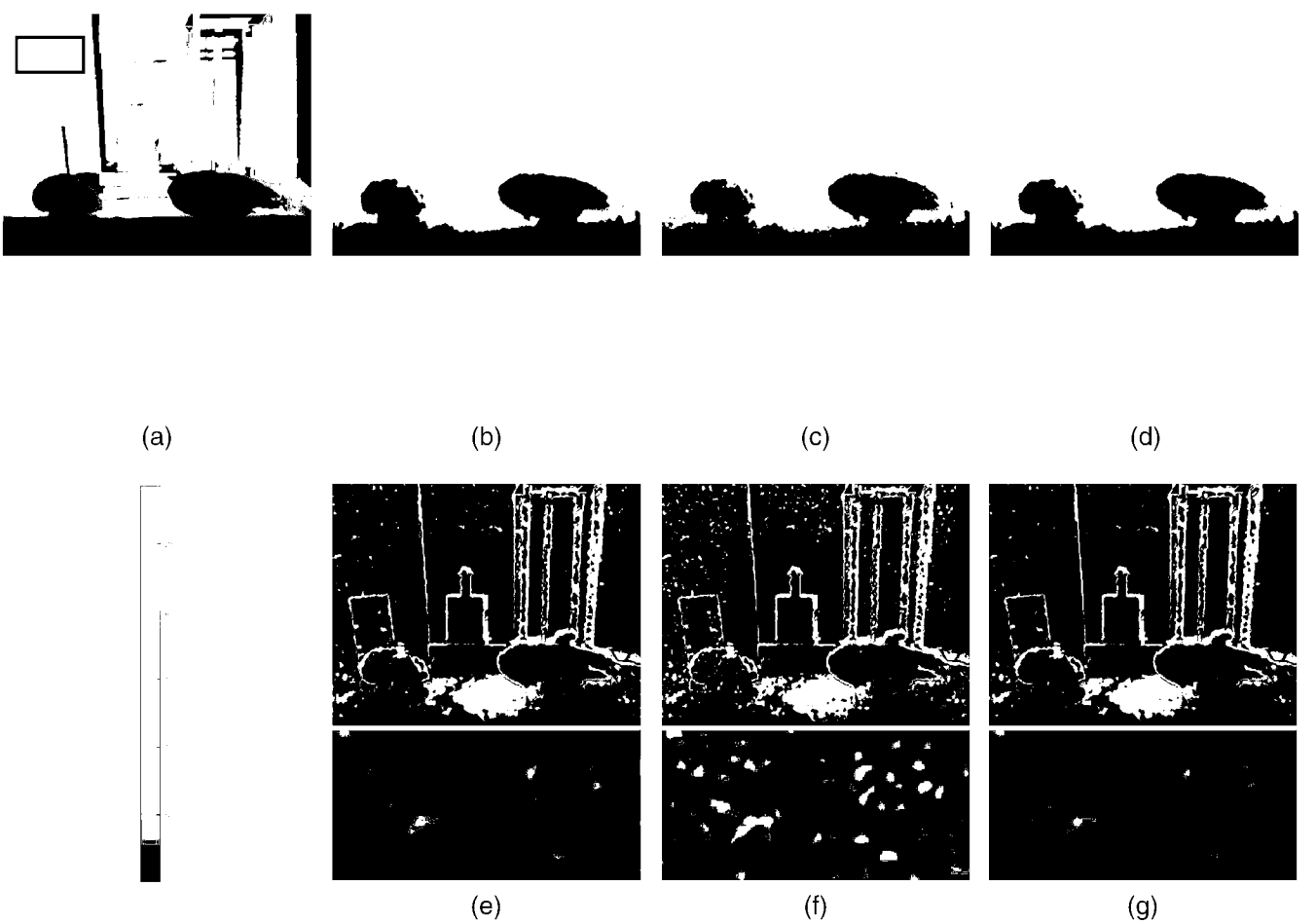

Fig. 9 Upsampling results on simulated ToF data with 4× upsampling. (a) Ground truth depth image. The result by (b) Xie,[11] (c) Hua,[8] and (d) ours. The corresponding error maps are shown at the bottom from (e)–(g). Region in red box is highlighted.

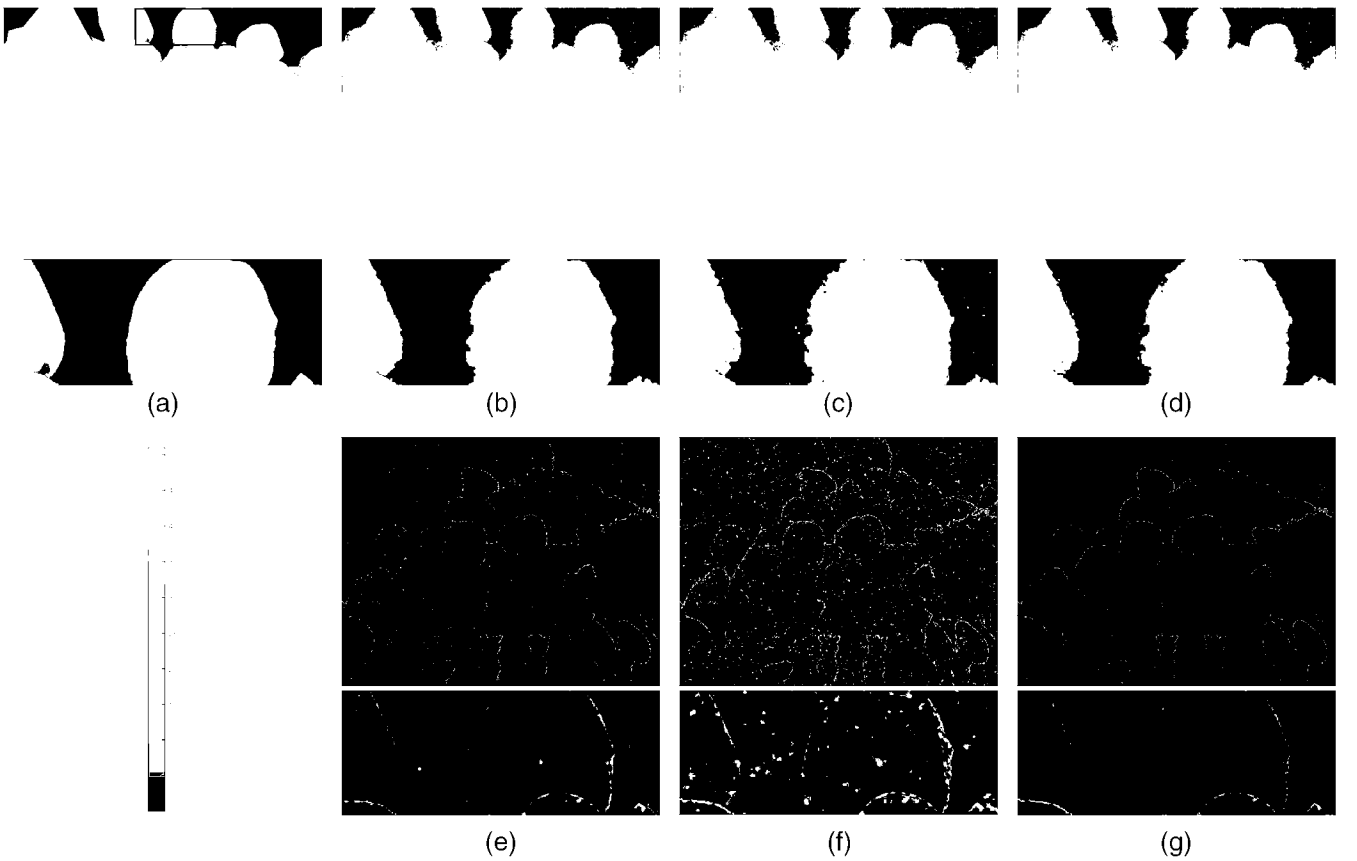

Fig. 10 Upsampling results on real ToF data. (a) Ground truth depth image. The result by (b) Xie,[11] (c) Hua[8] and (d) ours. The corresponding error maps are shown at the bottom from (e)–(g). Region in red box is highlighted.

($r_\lambda = 5, r = 4$). As shown in the figure, on the one hand, too small an θ may cause loss of weak depth edges, but the bandwidth map is quite clean. On the other hand, too large θ may make the depth edge map noisy, but weak depth edges are well preserved. Based on our observations, we find that when $\theta = 0.95 \sim 0.98$, it can produce satisfying results in most situations. In real applications, when the noise level is high, a smaller θ is recommended.

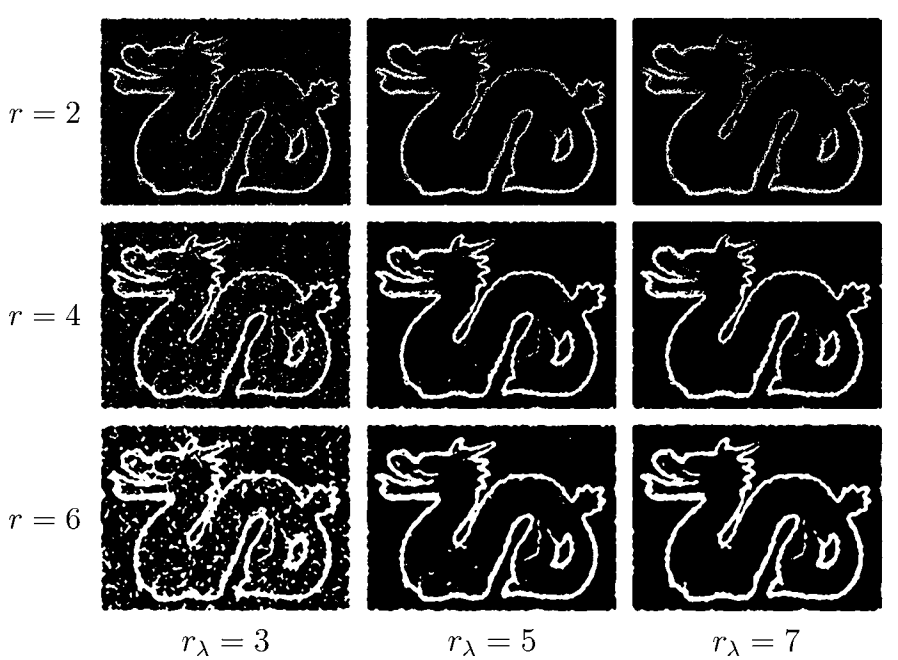

Fig. 11 Examples of the depth edge map obtained with different r_λ and r.

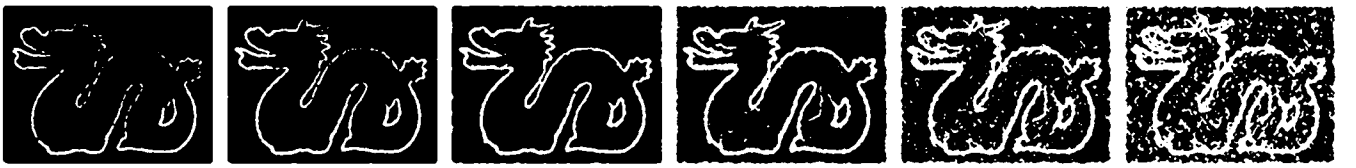

Fig. 12 Examples of the depth edge map obtained with threshold (a) $\theta = 0.94$, (b) $\theta = 0.95$, (c) $\theta = 0.96$, (d) $\theta = 0.97$, (e) $\theta = 0.98$, and (f) $\theta = 0.99$.

5 Conclusion

In this paper, we propose a method to detect edges in noisy depth images. It is very simple but very robust against the noise in depth images, which is useful for real applications. Moreover, we propose the near $\mathcal{O}(1)$ implementation of the proposed method. Experimental results on both simulated and real data show that the proposed method can yield satisfying results within very short time.

Acknowledgments

This research was partly supported by NSFC, China (Nos. 61273258, 61572315, and 61503250) and 863 Plan, China (No. 2015AA042308).

References

1. W. Liu et al., "RGB-D depth-map restoration using smooth depth neighborhood supports," *J. Electron. Imaging* **24**(3), 033015 (2015).
2. M. Hornacek et al., "Depth super resolution by rigid body self-similarity in 3D," in *Computer Vision and Pattern Recognition (CVPR)* (2013).
3. J. Park et al., "High quality depth map upsampling for 3D-ToF cameras," in *Int. Conf. on Computer Vision (ICCV)* (2011).
4. M.-Y. Liu, O. Tuzel, and Y. Taguchi, "Joint geodesic upsampling of depth images," in *Computer Vision and Pattern Recognition (CVPR)* (2013).
5. O. Mac Aodha et al., "Patch based synthesis for single depth image super-resolution," in *European Conf. on Computer Vision (ECCV)* (2012).
6. M. Camplani, T. Mantecon, and L. Salgado, "Depth-color fusion strategy for 3-D scene modeling with Kinect," *IEEE Trans. Cybern.* **43**(6), 1560–1571 (2013).
7. D. Ferstl, M. Ruther, and H. Bischof, "Variational depth superresolution using example-based edge representations," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 513–521 (2015).
8. K.-L. Hua, K.-H. Lo, and Y.-C. F. F. Wang, "Extended guided filtering for depth map upsampling," *IEEE Multimedia* **23**(2), 72–83 (2016).
9. W. Liu et al., "Shape preserving RGB-D depth map restoration," in *Int. Conf. Neural Information Processing*, pp. 150–158, Springer International Publishing (2014).
10. S. Schwarz, M. Sjostrom, and R. Olsson, "A weighted optimization approach to time-of-flight sensor fusion," *IEEE Trans. Image Process.* **23**(1), 214–225 (2014).
11. J. Xie, R. S. Feris, and M.-T. Sun, "Edge-guided single depth image super resolution," *IEEE Trans. Image Process.* **25**(1), 428–438 (2016).
12. W. Liu et al., "An MRF-based depth upsampling: upsample the depth map with its own property," *IEEE Signal Process Lett.* **22**(10), 1708–1712 (2015).
13. J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*(6), 679–698 (1986).
14. C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *2012 Second Int. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pp. 524–530, IEEE (2012).
15. D. Ferstl et al., "Image guided depth upsampling using anisotropic total generalized variation," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 993–1000 (2013).
16. D. Chan et al., "A noise-aware filter for real-time depth upsampling," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008* (2008).
17. F. C. Crow, "Summed-area tables for texture mapping," *SIGGRAPH Comput. Graph.* **18**(3), 207–212 (1984).
18. K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(6), 1397–1409 (2013).
19. X. Tan, C. Sun, and T. D. Pham, "Multipoint filtering with local polynomial approximation and range guidance," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2941–2948, IEEE (2014).
20. J. Lu et al., "Cross-based local multipoint filtering," in *2012 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 430–437, IEEE (2012).
21. D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2007)*, pp. 1–8, IEEE (2007).
22. J. Yang et al., "Color-guided depth recovery from RGB-D data using an adaptive autoregressive model," *IEEE Trans. Image Process.* **23**(8), 3443–3458 (2014).
23. C. Ionescu et al., "Human 3.6 m: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Trans. Pattern Anal. Mach. Intell.* **36**, 1325–1339 (2014).
24. J. S. Yedidia et al., "Generalized belief propagation," in *Neural Information Processing Systems (NIPS)*, Vol. 13, pp. 689–695 (2000).
25. https://github.com/ClaireXie/edgeGuidedSDSP

Wei Liu received his BS degree from the Department of Automation of Xi`an Jiaotong University in 2012. Currently, he is a PhD candidate of Shanghai Jiao Tong University at the Institute of Image Processing and Pattern Recognition under the supervision of professor Jie Yang. His current research interests mainly include image/video denoise, image deblur, and depth image restoration.

Xiaogang Chen received his PhD from Shanghai Jiao Tong University, Shanghai, China, in 2013. Then, he joined the University of Shanghai for Science and Technology, Shanghai, as an assistant professor. His current research interests mainly focus on image and video processing.

Qiang Wu received his BEng and MEng degrees from the Harbin Institute of Technology, Harbin, China, in 1996 and 1998, respectively, and his PhD from the University of Technology Sydney, Australia, in 2004. He is an associate professor and a core member of Global Big Data Technologies Centre at the University of Technology Sydney. His research interests include computer vision, image processing, pattern recognition, machine learning, and multimedia processing.

Jie Yang received his PhD from the Department of Computer Science, Hamburg University, Germany, in 1994. Currently, he is a professor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China. His major research interests are object detection and recognition, data fusion and data mining, and medical image processing.