J Electr Eng Technol Vol. 9, No. 3: 1016-1022, 2014
http://dx.doi.org/10.5370/JEET.2014.9.3.1016

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

Real-time Virtual-viewpoint Image Synthesis Algorithm Using Kinect Camera

Gyu-cheol Lee* and Jisang Yoo†

Abstract – Kinect is a motion sensing camera released by Microsoft in November 2010 for the Xbox360 that is used to produce depth and color images. Because Kinect uses an infrared pattern, it generates holes and noises around an object's boundaries in the obtained images. The flickering phenomenon and unmatched edges also occur. In this paper, we propose a real time virtual-view video synthesis algorithm which results in a high quality virtual view by solving these problems stated above. The experimental results show that the proposed algorithm performs much better than the conventional algorithms.

Keywords: 3D-warping, Kinect, Virtual-viewpoint, Occlusion region, Hole filling

1. Introduction

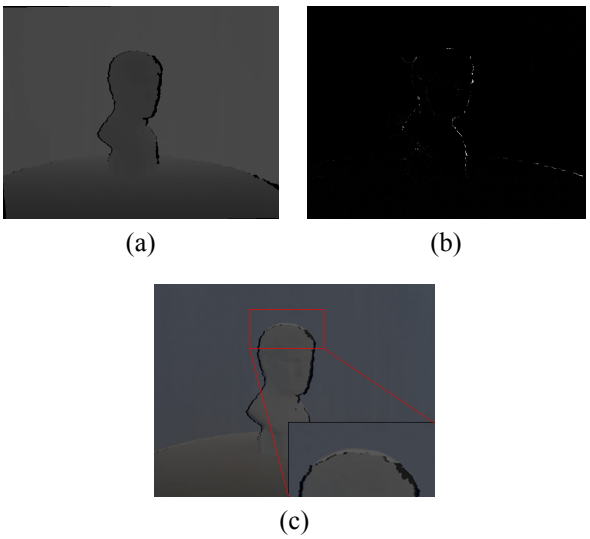
After the success of the 3D movie Avatar released in 2009, 3D stereoscopic content has been used in many areas such as movies, anime, games and sports broadcast. As products having 3D functions such as laptops, cell-phones and game devices have been released, the 3D market has been growing immensely. However, because of the inconvenience of wearing glasses and the lack of content, it has not yet been commonplace. Also, according to the data announced by the Retrevo, a market research company, 55% of consumers wanting to purchase HDTV have questioned themselves about the need of the 3D function [1].

Another reason why the 3D function is not as popular is because it only has a single view point. If the point of view is changed, the object appears unnatural. Therefore, multi-view display system without glasses has been developed recently [2]. Since the multi-view display system does not restrict users to a single-view, it allows them to watch dynamic scenes from any angle.

To implement the multi-view display systems, we need to generate virtual views to overcome the discomfort of using multiple cameras. In order to generate a virtual view, we need the depth information corresponding to the color image. A depth camera is generally used to obtain the depth information. A stereo matching algorithm can also be used to obtain depth information. The depth camera can acquire a high-accuracy depth image but its resolution is still low and the device is expensive. The stereo matching algorithm is relatively robust to environment but the execution time is long, and the quality of depth image is low [3].

Recently, cameras with a good price and performance ratio such as the Kinect have been released in the market so that the depth information can be obtained with ease [4]. However, despite the low cost and easy acquisition of such products as Kinect, defects such as holes and noises around boundary region of the depth image exist as shown in Fig. 1(a) [5]. Between the successive depth image frames, flickering phenomenon also occurs near boundaries as shown in Fig. 1(b) which shows the difference between current and previous frames. The depth image produced by Kinect has another problem where edges between color and depth images are unmatched as shown in Fig. 1(c) [6]. Currently, many researches are conducting research to compensate for these problems.

In this paper, we have proposed a real time virtual-view



(a) (b)

(c)

Fig. 1. (a) Depth image acquired by Kinect; (b) Images showing the difference of the current and previous frames; (c) Unmatched edges problem

† Corresponding Author: Dept. of Electronic Engineering, Kwangwoon University, Korea. (jsyoo@kw.ac.kr)
* Dept. of Electronic Engineering, Kwangwoon University, Korea. (gyucheol0116@gmail.com)
Received: June 9, 2013; Accepted: March 13, 2014

synthesis algorithm by using the Kinect camera system. To fill the holes in the depth image, many algorithms including in-painting or joint bilateral filter are proposed [7, 8]. They work well to fill holes but cannot solve the boundary flickering phenomenon that exists between successive frames because they work only on a single frame [9]. There is another problem where edges between the color and depth images do not match. In the proposed algorithm, we solve these problems during the hole filling process. First, we remove the wrong depth values using a region growing method [10]. Then, we fill the holes by using a joint bilateral filter. The flickering phenomenon is also solved by analyzing variation of intensity and depth images [9].

In general, bidirectional linear interpolation algorithm or 3D warping technique is used to generate a virtual view. Bidirectional linear interpolation is a method of generating an intermediate view between reference views with disparity information [11]. In 3D warping, world coordinates of all pixels in an image are calculated by using extrinsic and intrinsic parameters of a camera, and then projected back to a virtual-view image plane [12]. It synthesizes not only an intermediate view, but also a virtual view so that various points of view are provided.

We synthesize a virtual-view using 3D warping algorithm. However, 3D warping is not able to project pixels to disoccluded regions, because these regions only emerge from the virtual viewpoint. To make the virtual view look more natural and reasonable, a hole filling algorithm is designed to recover such disocclusion [13]. Oh et al. filled the disocclusions with an average of the available background pixels in their neighborhood [14]. However, the filling results tend to be over-smoothed when there are complex textures in the background. To overcome this, classical spatial filling method such as exemplar based in-painting by Criminisi et al. was used [15]. However, this approach performs worse for hole filling than for image restoration, because the neighborhoods of the in-painting area are on the depth boundaries which contain a lot of ambiguities. Ko et al. [12] proposed spiral weighted average method. However, this approach is still far from real-time processing.

Thus, we have proposed a new hole filling algorithm. The proposed algorithm uses gradient information of an image in order to synthesize natural views. Also, we have warped the depth image, so that the holes in the synthesized view could be filled effectively and efficiently. The proposed algorithm is also implemented for real-time applications by GPU programming.

The rest of the paper is organized as follows. Section 2 describes a depth image enhancement algorithm and in Section 3, the virtual view synthesis method is explained. Experimental results of the proposed method are given and discussed in Section 4. Finally Section 5 contains the conclusion.

2. Depth Image Enhancement

The Kinect camera system yields holes and noises around objects' boundaries in its depth image with a flickering phenomenon. Therefore, it is essential to enhance the quality of the depth image in order to generate a high quality virtual-view image. Fig. 2 shows a block diagram of the proposed depth image enhancement algorithm.

2.1 Correction of wrong depth values [6]

Pixels with wrong depth values are usually located between the boundaries of depth image and the corresponding color image. In the proposed algorithm, boundary regions are extended in a spiral direction in order to remove these pixels. First, the boundary region of the depth image (green line) is extended until it reaches the boundary of corresponding color image (red line) as shown in Fig. 3(b). The similar process is performed for the color image boundary as shown in Fig. 3(c). Regions that are white in Figs. 3(d) and (e) are the extended regions from Figs. 3(b) and (c), respectively. Then, we take the AND operator on these two extended regions and the result is pixels with the wrong depth value as in Fig. 3(f). However, some of the pixels located near the boundary may be excluded in this process. Thus, the region in Fig. 3(f) is expanded by dilation operation with a 3x3 structuring element as shown in Fig. 3(g) to fix this error. Finally, we consider these extended regions as part of the hole region and remove them just as we remove a hole region.

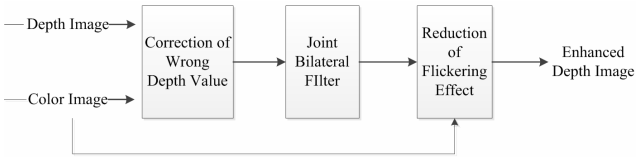


Fig. 2. Depth image enhancement

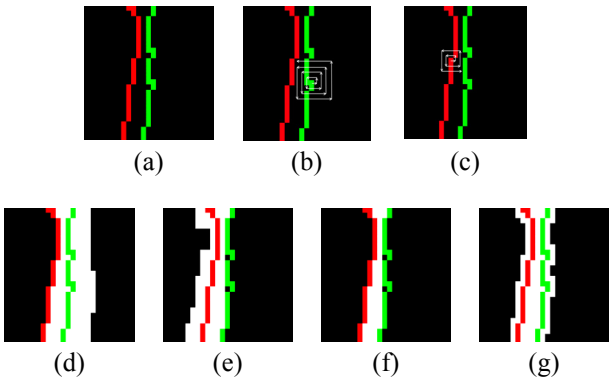


Fig. 3. Searching for a region with the wrong depth values (green: boundary of the depth image, red : boundary of the color image)

2.2 Hole filling

The depth images produced by Kinect have holes because of the different positions of the infrared light-emitting sensor and the receiving sensor. The holes in the depth image are created when an object has a smooth surface. Because of the diffuse reflection of the infrared light, the receiving sensor cannot receive the laser signal. Therefore, it is essential to fill these holes by using a joint bilateral filter which results in edges of depth image similar to those of color image to generate a high quality virtual-view image. In general, a joint bilateral filter such as (1) makes an image smooth while preserving the edges.

$$D'_p = \frac{1}{W_p} \sum_{q \in s} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(I_p - I_q) D_q \tag{1}$$

$$W_p = \sum_{q \in s} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(I_p - I_q) \tag{2}$$

where D is the depth image and I is the intensity image and D'_p is the pixel value generated by applying a joint bilateral filter to D and I. G is the Gaussian function and $\|p - q\|$ is the Euclidean distance between p and q. s is a set of neighboring pixels of p. σ_s and σ_r are parameters defining the size of neighborhood and W_p is the normalization constant. Fig. 4(b) is the result of joint bilateral filtering which shows that the holes are filled.

2.3 Reduction of flickering effect

Even when the holes in the depth image are filled by the joint bilateral filter, the flickering phenomenon still remains around boundary regions. First, the flickering pixels must be detected in order to reduce this flickering phenomenon. We take average value of each pixel from N previous frames and if the difference between pixel values of the current depth frame and the averaged value is greater than a threshold α1, the pixel is determined to be flickering as shown in Eq. (3). Similarly, we obtain an average value with intensity images. If the difference between a pixel value of the current intensity frame and the averaged intensity value is smaller than a threshold α2, it is also determined to be a flickering pixel as shown in Eq. (4).

(a) (b)

Fig. 4. (a) Depth image with the wrong depth value removed; (b) Result of joint bilateral filtering

$$D_N - \frac{1}{N} \sum_{n=1}^{N} D_n > \alpha_1 \tag{3}$$

$$I_N - \frac{1}{N} \sum_{n=1}^{N} I_n < \alpha_2 \tag{4}$$

where D_n is the pixel value of the nth depth image with its holes filled and I_n is the pixel value of nth intensity image. N is the number of reference frames.

Detected flickering pixels are regarded as a part of the background and are replaced with the maximum pixel value at the same location from the previous filled depth images of N. This proposed algorithm applies only in the boundary regions because the flickering phenomenon usually occurs at the boundary regions.

3. Virtual View Image Synthesis

3.1 Virtual view image synthesis

A virtual view is generated by the 3D warping technique with the enhanced depth image. In 3D warping, world coordinates of all pixels in an image are calculated by using extrinsic and intrinsic parameters of a camera and they are projected back onto a virtual-view image plane. Referring to the pinhole camera model, a camera matrix in Eq. (5) is used to denote a projective mapping from world coordinates to pixel coordinates.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} R \mid T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

where x and y represent the 2D point position in pixel coordinate and X, Y, Z means 3D point position in world coordinates. K is the intrinsic parameter, R is a 3 x 3 rotation matrix and T is a 3 x 1 translation vector. X, Y, Z must be calculated to generate a virtual view. X, Y are calculated by applying the transposed matrix and the inverse matrix to Eq. (5) as shown in Eq. (6). Z is also calculated by Eq. (7)

$$\begin{bmatrix} x_v \\ y_v \\ 1 \end{bmatrix} = K_v \begin{bmatrix} R_v \mid T_v \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \Rightarrow K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
$$\Rightarrow R^T K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - R^T T = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6}$$

$$Z(i,j)=\frac{1.0}{\left(\frac{D(i,j)}{255.0}\left(\frac{1.0}{MinZ}-\frac{1.0}{MaxZ}\right)+\frac{1.0}{MaxZ}\right)} \tag{7}$$

where *Z(i,j)* is the distance between an object and the camera, and *D(i,j)* is the pixel value of the depth image at *(i,j)*. *MaxZ* and *MinZ* are the maximum and minimum of *Z(i, j),* respectively. Coordinates of virtual view can be calculated by applying the intrinsic and extrinsic parameters to Eq. (8).

$$\begin{bmatrix} x_v \\ y_v \\ 1 \end{bmatrix} = K_v\left[R_v \mid T_v\right]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{8}$$

where x_v and y_v represent the 2D point position in pixel coordinates of the virtual view. K_v, R_v and T_v are the intrinsic parameter, rotation matrix and translation matrix of camera of virtual view, respectively.

3.2 Hole filling

A critical problem occurs when generating a virtual view: that is, regions covered by the foreground objects in the original view may be disoccluded in the synthesized view. In this section, we explore and identify the fundamental mechanism of the hole filling process in the generated virtual view. In general, there are two methods used to fill disocclusion regions. The first method is to make the depth image smooth before rendering a new view to reduce the size of the big hole after warping. However, this pre-processing of the depth map with smoothing results in a geometric distortion. Second method is to fill the disocclusion region with available background pixels from the surrounding area. We use the second method in order to distinguish objects from the background effectively. Fig. 5 shows the block diagram of the proposed hole filling algorithm.

The simplest hole filling method is a horizontal interpolation where the holes are filled with the background pixels horizontally. However, this method brings unnatural results when the vertical edge elements or irregular structures of the background exist.

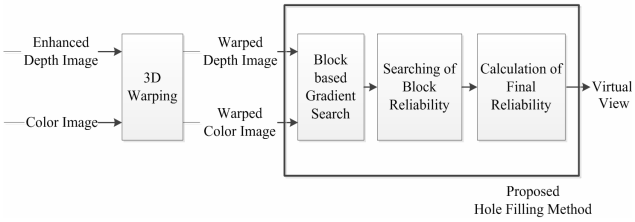


Fig. 5. Hole filling method

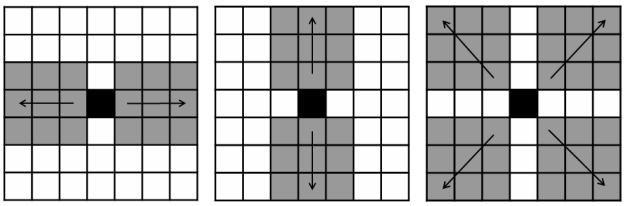


Fig. 6. Block based gradient searching method

We use a block based gradient search algorithm in order to increase continuity between background region and the hole region. Fig. 6 shows 8-blocks around the hole and gradient variation of each block is calculated by using first derivatives. This variation is divided into the maximum gradient variation as shown in Eq. (9) to convert gradient variation into a percentage,

$$\Delta G_B=\frac{1}{K}\sum_{\varepsilon\in B}\left\|I_c-I_\varepsilon\right\| \tag{9}$$

where ΔG_B is the gradient variation, and I_c is the pixel value of the center of the block. I_ε means the pixel value inside the block and K is the maximum gradient variation.

Block based gradient search algorithm is a method where the block that has the largest gradient variation is preferred. However, if a block has many holes, the holes may not be filled perfectly. To solve this problem, reliability of the block must be considered. The block reliability represents the ratio of the block size to the number of pixels that belong to the non-hole region in a block as shown in Eq. (10).

$$C=\frac{\sum_{q\in B}D(q)}{n(B)} \qquad where,\ D(q)=\begin{cases}0, & q\in hole\\ 1, & otherwise\end{cases} \tag{10}$$

where *C* is the block reliability and *n(B)* is the block size. *q* represents a pixel in the block.

The gradient variation and the block reliability are multiplied to calculate the final block reliability. Then, the block which has the highest reliability is used for hole filling. A hole is filled with a proximate pixel from the hole in the block. The final block reliability is calculated by Eq. (11) and Eq. (12).

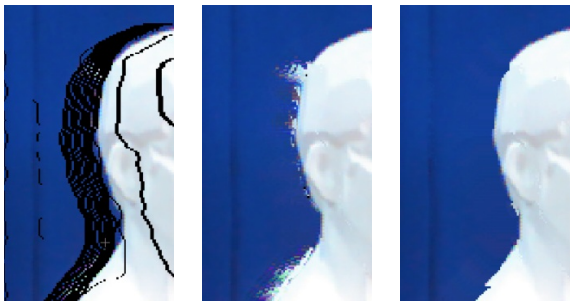
$$T_n=\Delta G_n C_n \quad n\in N(B) \tag{11}$$

$$T=\arg\max_{n\in N(B)}(T_n) \tag{12}$$

where T_n is the final reliability of nth block and ΔG_n is the gradient difference of *n*th block. C_n is the reliability of nth block and *N(B)* is the number of blocks.

3.3 Boundary noise removal

The quality of the virtual view synthesis highly depends



(a) (b) (c)

Fig. 7. (a) Boundary noises; (b) Hole filling with boundary noises; (c) Hole filling without boundary noises

on the accuracy of depth image. During the 3D warping, boundary noises may occur due to the mismatch of boundaries in the depth image and the corresponding color image. This boundary noise yields some pixels to be projected in wrong positions in the virtual view, resulting in geometry distortions. As shown in Fig. 7(b), fractions of the head exist in the background. We also consider these boundary noises as a part of hole region and boundary noises can be detected by analyzing the variation of pixel values around the hole region.

4. Experimental Results

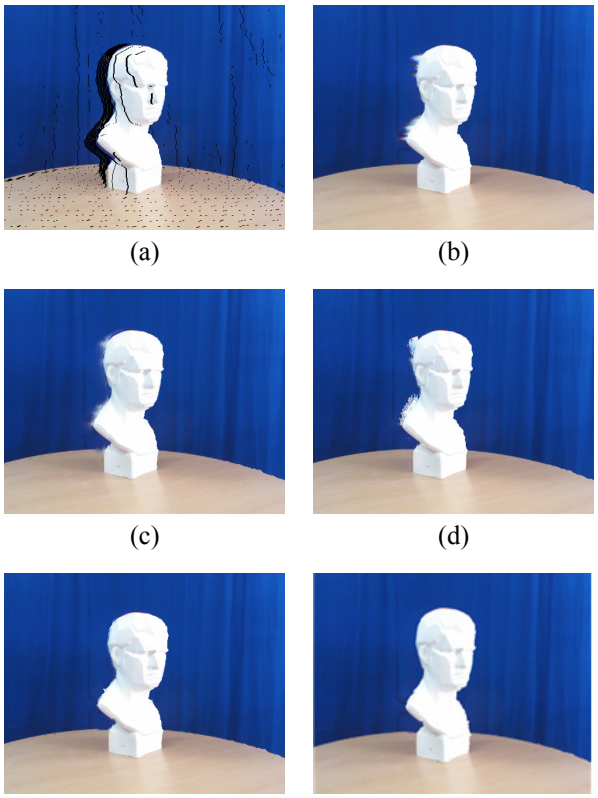
To evaluate the performance of the proposed algorithm, we used 'David', 'Roundtable', and 'Bookshelf' sequences with a size of 640 x 480 acquired from the Kinect camera system. We used the Laplacian operator to obtain the edge maps of both the depth and color images. Five successive frames were used to detect flickering pixels. We also set the values of α1 and α2 in Eq. (3) and Eq. (4) as 1 and 3, respectively, which gave the best results of the proposed algorithm.

We compared the performance of the proposed algorithm with linear interpolation, Telea's in-painting [7], Criminisi's in-painting [15], and Spiral weighted average algorithms [12]. In Figs. 8, 9 and 10, the generated virtual views by five different algorithms are shown. In Figs. 8(f), 9(f) and 10(f), the holes were filled by the proposed algorithm and we obtained a more natural virtual-view than that of other algorithms. The results of the proposed algorithms are also uploaded on our ftp server (ftp://128.134.65.1, ID:guest).

Fig. 11 shows the difference image of the current and previous frames. Fig. 12 shows the temporal variations of PSNR between two successive frames. As shown in Fig. 11(b) and Fig. 12, the flickering phenomenon is greatly reduced by the proposed algorithm.

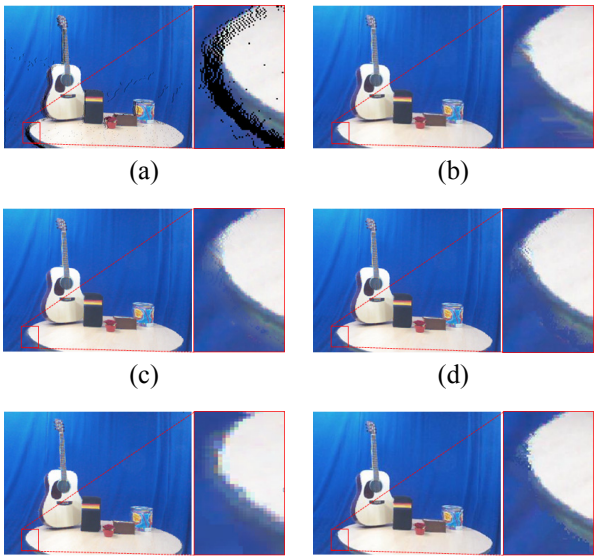
Table1. PSNR between successive frames

Flicking Process	PSNR(dB)
Before	44.06
After	56.91



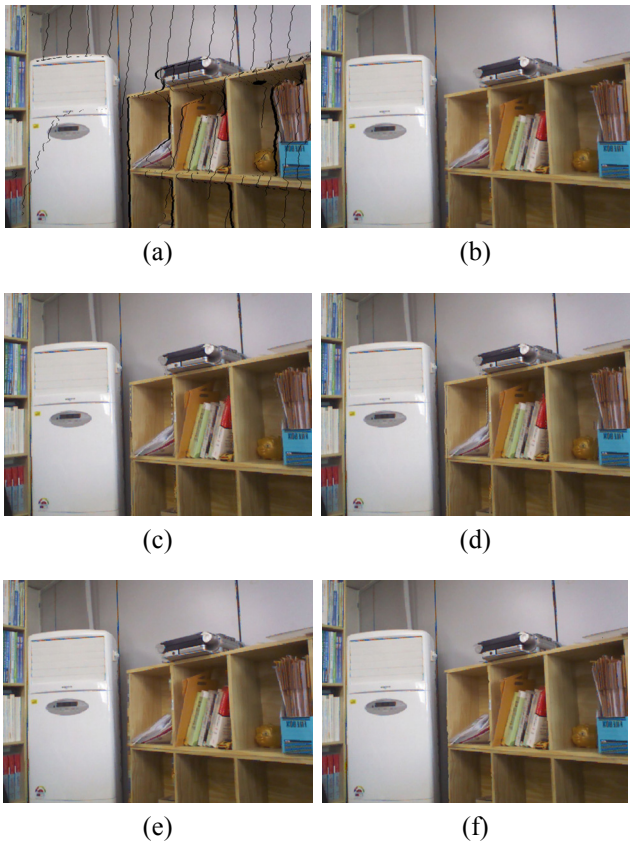
(a) (b) (c) (d) (e) (f)

Fig. 8. Generated virtual views for the 'David' sequence: (a) Before processing; (b) Linear interpolation; (c) Telea's in-painting; (d) Criminisi's in-painting; (e) Spiral weighted average method; (f) Proposed method



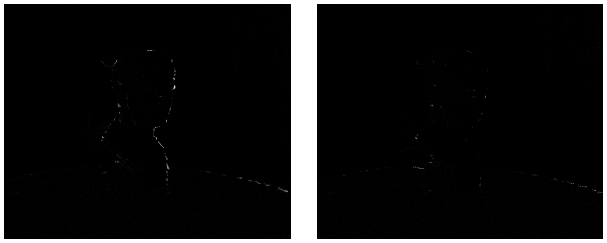
(a) (b) (c) (d) (e) (f)

Fig. 9. Generated virtual views for 'Roundtable' sequence: (a) Before processing; (b) Linear interpolation; (c) Telea's in-painting; (d) Criminisi's in-painting; (e) Spiral weighted average method; (f) Proposed method



(a) (b) (c) (d) (e) (f)

Fig. 10. Generated virtual views for 'Bookshelf' sequence: (a) Before processing; (b) Linear interpolation; (c) Telea's in-painting; (d) Criminisi's in-painting; (e) Spiral weighted average method; (f) Proposed method



(a) (b)

Fig. 11. Images showing the difference of the current and previous frames after hole filling: (a) Before flickering pixel processing; (b) After flickering pixel processing

Table 1 shows the average PSNR between 30 successive frames before and after applying the flickering process. The PNSR of the proposed method is greater than that of original method. It is clear that the flickering phenomenon is greatly reduced by the proposed algorithm.

Table 2 and Table 3 show specifications of the computer used in this experiment and the computational time of the hole filling process for each algorithm, respectively. From Table 2, it is clear that the proposed algorithm is faster than

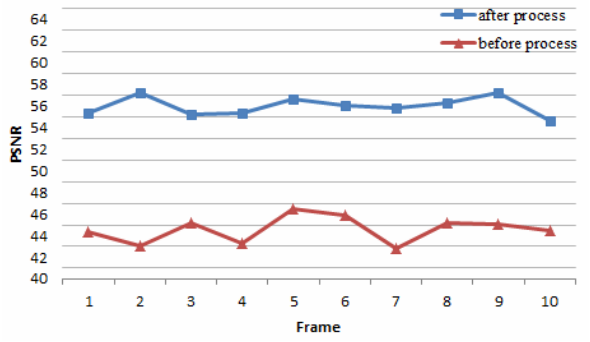


Fig. 12. Temporal variation of the PSNR on successive frames

Table 2. Computer spec. used in the experiments

Hardware	Name of product
CPU	Intel Core i5-2500@ 3.30GHz
VGA	NVIDIA GeForce GTX 670
RAM	8GB

Table 3. Performance comparison (processing speed)

Method	Computational time (ms)
Linear interpolation	0.3
In-painting(Telea)	100.4
In-painting(Criminisi)	4596.7
Spiral weighted average method	12.0
The proposed method	10.0

Table4. Computational time of each process

Process	Time(ms)
Removal the wrong depth value	1.5
Joint bilateral filter	11.5
Flickering pixel processing	6.5
3D warping	5.0
Boundary noises removal	2.0
Block based gradient search	8.0
Total time consumption	34.5(29.0fps)

Criminisi's In-painting algorithm and as fast as the spiral weighted average algorithm.

Table 4 shows the computational time of each process in the proposed algorithm. To reduce the processing time, we implemented the joint bilateral filtering with GPU programming and thus the proposed algorithm can be implemented for some of real-time applications.

5. Conclusion

In this paper, we proposed a new virtual-view synthesis algorithm by using the Kinect camera system for real-time application. In the proposed algorithm, we enhanced the quality of the obtained depth map by correcting the wrong depth values and used a joint bilateral filter to fill holes. We also reduced the flickering phenomenon in the obtained depth map.

We finally synthesized the virtual-view by using the 3D

warping algorithm and removed disocclusion regions by using a block based gradient searching algorithm. Experimental results show that the proposed algorithm resulted in a better quality of the synthesized virtual-view image even in real time.

Acknowledgements

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP). (2011-0006791)

References

[1] Retrevo Corporation, Could low interest in 3DTV hurt the TV business?, Retrieved Nov., 2011, from http://www.retrevo.com/content/node/1915.

[2] G. M. Um, G. H. Cheong, W. S. Cheong and N. H. Hur, "Technical development and standardization trends of multi-view 3D and free-viewpoint video," The Magazine of the IEEK, vol. 38, no. 2, pp. 18-23, Feb. 2011.

[3] T.J. Kim and J.S. Yoo, "Hierarchical stereo matching with color information," The Journal of Korea Institute of Communications and Information Sciences, vol. 34, no. 3, pp. 279-287, Mar. 2009.

[4] T. Leyvand, C. Meekhof, Y. C. Wei, J. Sun and B. Guo, "Kinect identity: technology and experience," Computer published by the IEEE Computer Society, vol. 44, no. 4, pp. 94-96, Apr. 2011.

[5] Y. S. Park, S. M. Yun and C. S. Won, "Hole filling for kinect depth image according to the causes of the holes," The Conference of Korean Society of Broadcast Engineers, Jeju univ, Korea, pp. 75-80, Jul. 2012.

[6] L. Chen and S. Li, "Depth image enhancement for Kinect using region growing and bilateral filter," ICPR 2012, pp. 3070-3073, japan, Nov. 2012

[7] A. Telea, "An image inpainting technique based on the fast marching method," J. Graphics Tools, vol. 9, no. 1, pp. 25-36, Dec. 2004.

[8] L. Zhao and H. Wang, "Image denoising using trivariate shrinkage filter in the wavelet domain and joint bilateral filter in the spatial Domain," IEEE Trans. Image Process., vol. 18, no. 10, pp. 2364-2369, Oct. 2009.

[9] G. C. Lee, Y. H. Seo and J. S. Yoo, "GPGPU-based multiview synthesis using kinect depth image," The Conference of Korea Institute of Communications and Information Sciences, Yongpyong, Korea, Jan. 2012.

[10] S. A. Hojjatoleslami andJ. Kittler, "Region Growing: a New Approach," IEEE Transactions on Image Processing, vol. 7, no. 7, pp. 1079-1084, July. 1998.

[11] C. J. Park, J. H. Ko, and E. S. Kim, "A new intermediate view reconstruction scheme based-on stereo image rectification algorithm," J. KICS, vol. 29, no. 5C, pp. 632-641, May. 2004.

[12] M. S. Ko and J. S. Yoo, "Boundary noises removal and hole filling algorithm for virtual viewpoint image generation," J. KICS, vol. 37, no. 8, pp. 679-688, Aug. 2012.

[13] W. Sun, O. Au, L. Xu, Y.Li and W. Hu, "Novel temporal domain hole filling based on bakground modeling for view synthesis," IEEE International Conference on Image Processing 2012, Florida, USA, Oct. 2012

[14] K.J. Oh, S. Yea, and Y.S. Ho, "Hole filling method using depth based inpainting for view synthesis in free viewpoint television and 3-d video," Proc. of the 27th conference on Picture Coding Symposium (PCS'09), pp. 233-236, May. 2009.

[15] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image in-painting," IEEE Trans. Image Process., vol. 13, no. 9, pp. 1200-1212, Sep. 2004.

[16] G. C. Lee and J. S. Yoo, "Real-time virtual-view image synthesis algorithm using Kinect camera," The Journal of Korea Institute of Communications and Information Sciences, vol.38, no.5, pp. 409-419, May. 2013.

Gyu-cheol Lee He received B.S. degree in electronics engineering from Kwangwoon University in 2013. His research interests are in digital image processing, multi-view synthesis, 3D-warping, stereo matching.

Jisang Yoo He was born in Seoul, Korea in 1962. He received the B.S. and M.S. degrees from Seoul national university, Seoul, Korea in 1985 and 1987, all in electronics engineering, and Ph.D. degree from Purdue University, West Lafayette, IN, USA, in electrical engineering in 1993, respectively. From september 1993 to august 1994, he worked as a senior research engineer in industrial electronics R&D center at Hyundai Electronics Industries Co., Ltd, ichon, Korea, in the area of image compression and HDTV. He is currently a professor with the department of electronics engineering, Kwangwoon University, Seoul, Korea. His research interests are in signal and image processing, nonlinear digital filtering, and computer vision. He is now leading 3DTV broadcast trial project in Korea.