2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

3-D Reconstruction Using the Kinect Sensor and Its Application to a Visualization System

Yong-Wan Lim, Hyuk-Zae Lee, Na-Eun Yang, and Rae-Hong Park
Dept. of Electronic Engineering, School of Engineering
Sogang University
Seoul, Republic of Korea
{lim86411, djttovnf, naeun, rhpark}@sogang.ac.kr

***Abstract*—These days, a large number of human-computer interactive systems have been developed. This paper proposes a visualization system of reconstructed three-dimensional (3-D) scene, which is based on user-interactive control using the Kinect sensor and a webcam. First, we acquire depth and color images from a number of viewpoints, from which 3-D reconstruction of a scene is performed. We pre-process the acquired data to improve 3-D reconstruction result. Then, we reconstruct the 3-D scene for the visualization system that uses face detection and tracking with a webcam. Experimental results show that our system gives a visualized 3-D scene reconstruction simply using the Kinect sensor and a webcam. It can be applied to a human-interaction display system at museum, amusement park, and so on.**

Keywords—3-D reconstruction; Kinect sensor; visualization system; face detection

I. INTRODUCTION

Research on three-dimensional (3-D) technology includes 3-D contents that give reality effects by interacting with a user. 3-D reconstruction of real scenes has been studied for decades [1], [2]. Taking advantage of 3-D reconstruction techniques, visualization methods of virtual 3-D scene or object using multiple two-dimensional (2-D) photos or pseudo 3-D images also have been developed [3], [4].

The Kinect depth sensor, a game controller by Microsoft, captures 3-D information using structured infrared light [5]. From the Kinect sensor, we easily acquire depth information that has been commonly obtained by conventional 3-D scanner and time-of-flight (TOF) cameras. However, like existing TOF cameras, the Kinect sensor has inherent problems [6], [7] that quality of 3-D reconstruction is degraded due to acquisition error in both color and depth images. Therefore, in order to reduce performance degradation, we pre-process color and depth images obtained from the Kinect sensor.

In this paper, we reconstruct a 3-D scene with respect to user's viewpoints to give a real perspective effect. We perform 3-D reconstruction based on two existing methods, [1], [2] that have been commonly used to reconstruct 3-D scenes. With the 3-D reconstruction, we also implement a real-time visualization system with which a user can directly interact. When a user moves around a fixed webcam, the user can observe 3-D scene through our visualization system depending on the user's viewpoint. Then, the user may be under the illusion that the user sees a real scene through a window, which is what we aim to achieve in this research.

The rest of the paper is structured as follows. Section II describes the proposed a 3-D reconstruction and application system. Experimental results and discussions are given in Section III. Finally, in Section IV, conclusions and future research directions are given.

II. PROPOSED 3-D RECONSTRUCTION AND VISUALIZATION SYSTEM USING THE KINECT SENSOR

We propose a reconstruction system of a 3-D scene which visualizes the 3-D scene depending on user's viewpoint using color and depth data set obtained by the Kinect sensor. Fig. 1 shows a block diagram of the proposed 3-D reconstruction and 3-D visualization system. At any given viewpoint, using the Kinect sensor, we can obtain color (RGB) images I_n^d and depth maps D_n^d, where the superscript d denotes the direction of data acquisition position and the subscript n represents n-th viewpoint along the direction d. After pre-processing of the inputs I_n^d and D_n^d, we obtain color transferred image $\hat{I}_n^d$ and filtered depth map $\hat{D}_n^d$. Then, we reconstruct a 3-D scene I_{3D}, which is composed of 3-D points and corresponding RGB data from I_n^d, $\hat{I}_n^d$, and $\hat{D}_n^d$. $p(t)$ is an estimated position of detected face of a user in input frame sequence $F(t)$ from a webcam, which gives user's viewpoint. Eyes are stable and crucial facial features on the face [8]. Generally, eyes are located on one-third of a face image vertically from the top. Therefore, we visualize 3-D scenes on display device using the estimated position of a user.

A. Data Acquisition

Using the Kinect sensor and OpenNI, we acquire registered color images and depth maps at any given viewpoints. We acquire image data of an object or a scene from multiple viewpoints to build a 3-D scene within the field of viewpoints that are used for reconstruction.

In this paper, we acquire images at the maximum of 16 viewpoints as shown in Fig. 2, in which the reference viewpoint marked by a square. First, from the front reference viewpoint, we capture the reference data set of a scene of

This work was supported in part by the Second Brain Korea 21 Project.

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

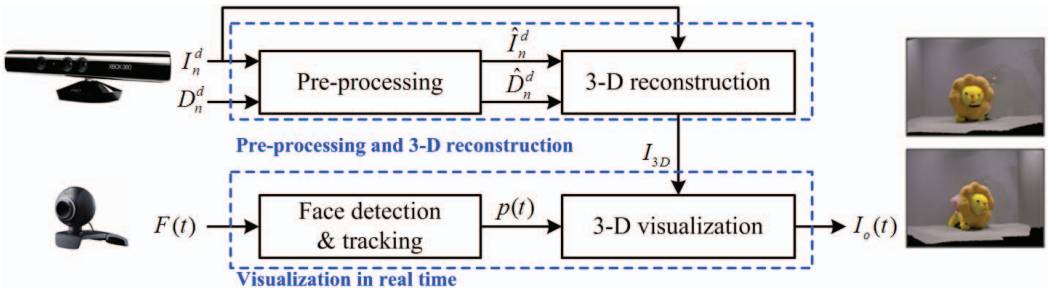


Figure 1. Block diagram of the proposed 3-D reconstruction and visualization system.

interest, which consists of color image I_{ref} and depth map D_{ref}. Then, in our experiments, multiple images are acquired by sequentially increasing the angle of viewpoint by about 15°, along the left, right, and upper directions with respect to the reference viewpoint. I_n^d is a color image and D_n^d be a depth map, $1 \le n \le N$, acquired from n-th viewpoint along direction d, where the superscript $d = l, r,$ and u denote left, right, and upper directions, respectively. Without loss of generality, I_0^d and D_0^d are regarded as I_{ref} and D_{ref}, respectively, i.e., $I_{ref} = I_0^l = I_0^r = I_0^u$ and $D_{ref} = D_0^l = D_0^r = D_0^u$.

Depth maps obtained by the Kinect sensor have some unstable points that degrade the quality of 3-D reconstruction [6], [7]. During data acquisition process, we perform acquisition steps K times to capture K images at the same viewpoint to reduce the number of unstable points. Then, the depth maps are pre-processed to compute mean and variance of K depth maps, which will be explained in the next section. It is also important that spatially adjacent images have common overlapped area, where the scale invariant feature transform (SIFT) [9] is used to extract the same feature points between two adjacent images.

B. Pre-processing

Acquired data have different color characteristics and some mis-aligned pixels because the data are acquired from different viewpoints. Therefore, in order to make acquired data have similar color characteristics among multiple images and to compensate for mis-aligned pixels between a depth map and its corresponding color image, we propose a two-step pre-processing method: transformation of statistical color information (mean and standard deviation) of color images and removal of unstable depth pixels.

1) Color transfer

In multiple color images acquired at different viewpoints, there may exist different color characteristics, because the location of light source can vary due to different viewpoints in data acquisition, and so on. Different color characteristics reduce the quality of 3-D reconstruction results. Thus, we need to transform color characteristics of color images to have similar color characteristics.

We use a color transfer technique [10] based on the following facts. A front reference view image covers most of the scene to be reconstructed. Thus, color information of the reconstructed image will not be significantly different from the scene image obtained at the front reference viewpoint. For color transfer, we set color image I_0^d as target image and color images I_n^d, $1 \le n \le N$, acquired from other viewpoints as source images. Then, using statistical measures such as means and standard deviations of both the target and source images, color characteristics of source images are transferred in $l\alpha\beta$ color space to those of the target image [10], in which resulting image of color transfer is denoted as $\hat{I}_n^d$.

2) Depth map filtering

The Kinect sensor has inherent drawbacks such as mis-aligned pixels between depth map and color image, temporal flickering artifacts, and missing depth information [7]. Because both color and depth images are not acquired at the same viewpoint, they are not matched accurately. This problem can be solved using homography, but some mis-aligned pixels

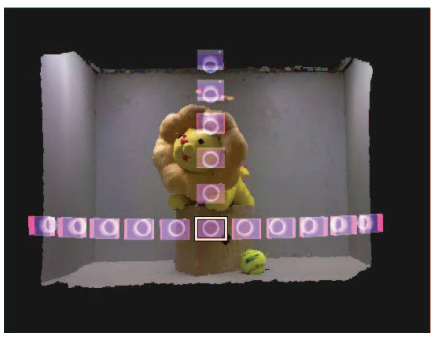
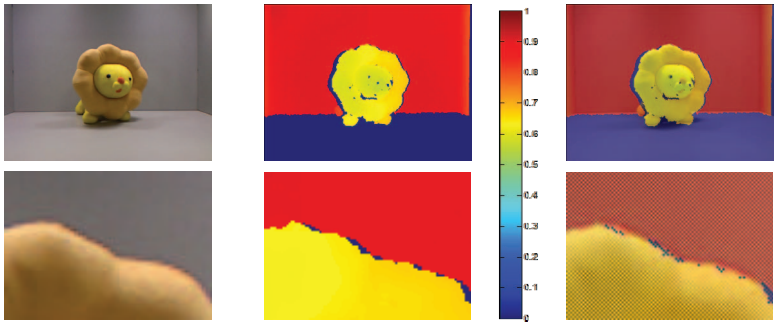


Figure 2. Locations of the Kinect sensor for data acquisition.



(a) (b) (c)

Figure 3. Mis-aligned pixels between acquired color and depth images (585 × 443). (a) color image, (b) depth image, (c) overlap of (a) and (b).

remain. Fig. 3 shows the mis-aligned pixels where top row shows entire images while bottom row shows zoomed and cropped images. Figs. 3(a) and 3(b) show an acquired color image and depth image, respectively. If Figs. 3(a) and 3(b) are overlapped, mis-aligned pixels are clearly observed. In Fig. 3(c), the mis-aligned pixels exist near the boundary of the lion's mane, where some pixels having false depth values appear. In order to eliminate mis-aligned pixels, we simply apply average filtering to depth maps. We obtain K images at each viewpoint to filter out mis-aligned points.

Let $D_{n,k}^d$ be k-th depth map of D_n^d, $1 \le k \le K$. To eliminate mis-aligned pixels between depth maps and corresponding color images, we calculate the mean $MD_n^d(x,y)$ and variance $VD_n^d(x,y)$ of the depth maps acquired from each viewpoint. At any given n-th viewpoint in direction d, we can determine a binary reliability map $R_n^d(x,y)$, which specifies unreliable pixels to be eliminated, if the variance $VD_n^d(x,y)$ is larger than threshold value, T_{VD}. Threshold value is experimentally set to 0.01, where $D_{n,k}^d$ is scaled to have a value from 0 to 1. The reliability map $R_n^d(x,y)$ can be expressed as,

$$R_n^d(x,y) = \begin{cases} 1, & \text{if } VD_n^d(x,y) \le T_{VD} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where pixels with value 1 are regarded as reliable elements of the depth map. Fig. 4(a) shows the reliability map of Fig. 3(b).

Near the unreliable pixels, some mis-aligned pixels with values smaller than the threshold are not detected as unreliable pixels. Therefore, we need to expand the initial unreliable region. We use morphological operations [11] with 5×5 circular structure element. The sequential operations are illustrated in Fig. 4. To expand the initial unreliable region, we perform dilation of $R_n^d(x,y)$, which is shown in Fig. 4(b). Then, closing of Fig. 4(b) is performed to fill the gaps in the contour as shown in Fig. 4(c). At last, we perform opening of Fig. 4(c) to eliminate small saliencies. Fig. 4(d) is the final result, $\hat{R}_n^d(x,y)$, in which expanded boundary pixels to be eliminated have value 0. We can eliminate mis-aligned pixels from $D_n^d(x,y)$ in Fig. 3(b) by using $\hat{R}_n^d(x,y)$ in Fig. 4(d). If $\hat{R}_n^d(x,y) = 0$, then $\hat{D}_n^d(x,y)$ is eliminated (set to zero) and otherwise $\hat{D}_n^d(x,y) = D_n^d(x,y)$. Hence, we get the filtered depth map $\hat{D}_n^d(x,y)$, as shown in Fig. 4(e). Fig. 4(f) is a zoomed and cropped image of Fig. 4(e). Fig. 4(g) shows an overlap image of input color image and output filtered depth map. Fig. 4(h) is a zoomed and cropped image of Fig. 4(g). As shown in Figs. 3(c) and 4(h), mis-aligned pixels near the boundary are eliminated through the depth map filtering.

Fig. 5 shows two integration results. Figs. 5(a) and 5(b) show integration results without and with filtering, respectively. Fig. 5(b) shows smaller integration error than Fig. 5(a). Mis-aligned pixels affect the quality of 3-D reconstruction result.

C. 3-D Reconstruction

In 3-D reconstruction, our goal is to register all pairs of adjacent images from multiple viewpoints and to integrate the registered images. In general, registration is to estimate rigid transformations; rotation matrix R and translation vector $\mathbf{t}$ between two adjacent images. Integration is to merge multiple registered images on a reference space.

We register the whole set of images, where the 2-D SIFT algorithm is used to match feature points [1], [2]. For each pair of two consecutive color images, the SIFT algorithm yields 2-D feature point pairs, where consecutive images represent images acquired consecutively along the same direction with respect to the reference viewpoint. Because 2-D feature points of the image pair have one-to-one correspondence relationship with their depth map points, we establish correspondences of feature points. Let $(\mathbf{p}_i, \mathbf{x}_i)$ be a 3-D matched point pair obtained from two consecutive images $\hat{I}_{n-1}^d$ and $\hat{I}_n^d$. Point $\mathbf{p}_i = (p_i, q_i)$ from $\hat{D}_{n-1}^d$ is assumed to correspond to point $\mathbf{x}_i = (x_i, y_i)$ from $\hat{D}_n^d$. Then, random sample consensus (RANSAC) algorithm [12] is used to eliminate outliers between the matched 3-D point pairs since iterative closest point (ICP) [13] algorithm is

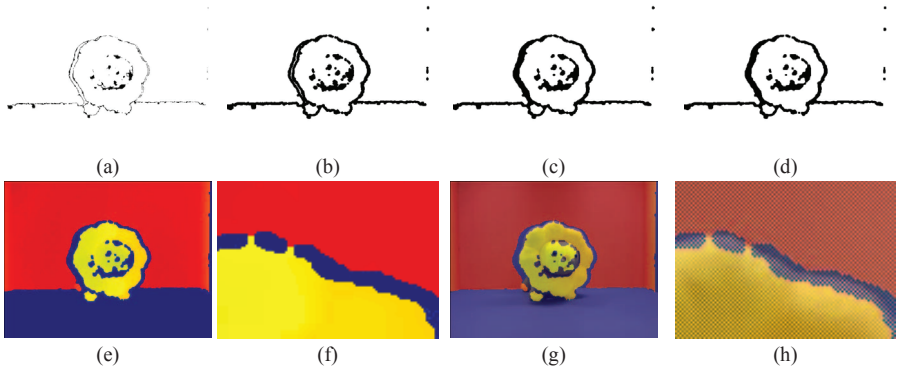


Figure 4. Depth map filtering. (a) reliability map $R_n^d(i,j)$, (b) dilation of (a), (c) closing of (b), (d) opening of (c), (e) output depth map, (f) zoomed and cropped image of (e), (g) overlap of (e) and color image, (h) zoomed and cropped image of (g).

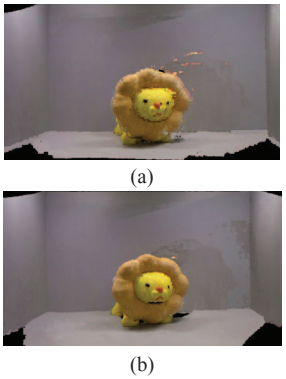


Figure 5. Integration results. (a) without depth map filtering, (b) with depth map filtering.

sensitive to initial input. The RANSAC gives appropriate initial point sets to the ICP. The ICP is used to estimate 3×3 rotation matrix $R_{n-1,n}^{d}$ and 3×1 translation vector $\mathbf{t}_{n-1,n}^{d}$ between two matched points, $\mathbf{p}_i$ and $\mathbf{x}_i$. Then, we register the whole points of two consecutive images using $R_{n-1,n}^{d}$ and $\mathbf{t}_{n-1,n}^{d}$.

In contrast to a global registration of multiple-view images [14], in this paper, pairwise registration of multi-view images is used. Although a global registration method minimizes accumulation errors and also reduces the registration error of all views by simultaneously estimating optimal transformations of all pairs, the method requires large memory and high computational complexity due to computing all possible combinations of pairs. So, we register consecutive image pairs sequentially for fast implementation.

To register multiple images to a reference image, R_n^d and $\mathbf{t}_n^d$ are calculated, which are expressed, respectively as [15]

$$R_n^d = R_{n-1}^d R_{n-1,n}^d \tag{2}$$

$$\mathbf{t}_n^d = R_{n-1,n}^d \mathbf{t}_{n-1}^d + \mathbf{t}_{n-1,n}^d \tag{3}$$

where d = l, r, and u, $1 \le n \le N$, $R_0^d = I$, and $\mathbf{t}_0^d = (0,0,0)^T$. N is the number of viewpoints along the direction, d. It is noted that $R_{n-1,n}^d$ and $\mathbf{t}_{n-1,n}^d$ are relative transformations between consecutive image pairs whereas R_n^d and $\mathbf{t}_n^d$ are transformations of n-th viewpoint image space with respect to the reference image space.

After all transformations are estimated, a whole set of registered images are merged together on the reference image space. Let P_n^d be a point in 3-D space, i.e., $P_n^d = \left(x, y, \hat{D}_n^d(x,y)\right)$ and $\widetilde{P}_n^d$ represent point transformed by R_n^d and t_n^d, i.e., $\widetilde{P}_n^d = R_n^d P_n^d + \mathbf{t}_n^d$. Let c_n^d be a set of color value (r,g,b) corresponding to $\hat{I}_n^d$ at location (x, y). Then, integration of 3-D points of the transformed pixels is expressed as

$$P_{3D} = \bigcup_d \bigcup_n \widetilde{P}_n^d, \tag{4}$$

where $\bigcup(\cdot)$ denotes union of all the points of images. Integration of color information is similarly written as

$$C_{3D} = \bigcup_d \bigcup_n \mathrm{c}_n^d \tag{5}$$

where C_{3D} is indexed in conjunction with P_{3D}. In visualization process, C_{3D} is mapped on corresponding points of P_{3D}. Then, reconstructed 3-D scene I_{3D} is composed of P_{3D} and C_{3D}.

D. Visulization System

1) Face detection

For the purpose of visualization, we detect and track user's face in real time to estimate user's viewpoint. If we estimate 3-D position of the face from detected face region, we can visualize a reconstructed image according to a viewpoint from which a user sees the object. We use a widely used face detection algorithm [16] based on adaboost algorithm [17].

For determining the viewpoint from a result of face detection, first, we use a bounding box that specifies the location of a detected face. Then, we estimate eye location within the detected face region. Whenever a face is detected, the z-coordinate, which represents a distance of the face from a webcam, is estimated using width and height of the box [4]. By doing this, we know the relative position of the face from the webcam with additional scaling to adjust a range of user control. However, it is difficult to determine which detected faces should be used if multiple faces are detected. In this paper, the largest face is selected because it is assumed that a face of a person closest to a webcam is the one to be selected among multiple faces detected in an image.

The position of the detected face, which is estimated from input image sequence $F(t)$, is denoted as

$$p(t) = (\bar{x}(t), \bar{y}(t), \bar{z}(t)) \tag{6}$$

where $p(t)$ represents the estimated current position of a face of a user. If a reconstructed image is visualized using the estimated position of a face frame by frame, the position is unstable to use. This is because the raw position of the detected face has temporal variation from frame to frame although the face location in the scene is fixed. To minimize the variation, the average value is used. Let $\bar{x}(t)$ be the average value of x-coordinates of successive frames up to frame t, then we can write

$$\bar{x}(t) = \frac{s}{M} \sum_{i=0}^{M-1} x(t-i) \tag{7}$$

where $x(t)$ is an estimated x-coordinate of a face at frame t, M is the number of frames averaged, and s is a constant specific to pixel size of input and output frames. Also, $\bar{y}(t)$ and $\bar{z}(t)$ can be computed similarly to $\bar{x}(t)$.

2) Visualization and display

One of objectives of the paper is how we realistically show the reconstructed image, which is related to not only quality of the reconstructed image but also realistic perspective effect in visualization process. We use OpenGL pipeline (including vertex operation, z-buffering, frame-buffering, fragment operation and so on) to implement visualization process [18].

The reconstructed 3-D scene I_{3D} is initially loaded into visualization loop once, whereas the estimated 3-D position of face $p(t)$ is updated in real time. Then, using generic OpenGL libraries and functions, 3-D point data P_{3D} of I_{3D} is transformed by a 3-D projection matrix that is updated using $p(t)$ in every frame.

Regardless of the viewpoints used in data acquisition process under the constraint such as range of $p(t)$, one sees novel parts of the 3-D reconstructed scene, on a display device from arbitrary viewpoint by moving one's face, otherwise occluded at the other viewpoints. In this way, one could experience realistic perspective effect and may be under the illusion that one sees real scene.

Fig. 6 shows graphic user interface (GUI), which is implemented based on Windows application programming interface. Users can see 3-D scene through the main window. Main section shows 3-D scene depending on the detected viewpoint. Upper right section shows captured images from a webcam in real time.

Figure 6. Graphic user interface.

III. Experimental Results and Discussions

We show the performance of our 3-D reconstruction using various objects in particular constraints. In order to minimize effects of light source, we capture multiple views under controllable illumination using spectra-light device (Macbeth SpectraLight III) [19]. The images are acquired along three directions with $N = 5$, i.e., a total of 16 views are obtained.

Fig. 7 illustrates three representative results. Each row in Fig. 7 shows some 3-D visualization results. From left to right, visualization results from front, upper-left, upper-right and upper viewpoints are shown. Our system can visualize a scene depending on user's viewpoint. Except for the front view, other result shows 3-D reconstruction of virtual views. The first row shows lion and octopus dolls which are reconstructed from 16 different viewpoints. In the left side view (second column), the part of pink octopus doll appears otherwise occluded by the lion doll at the other views. The performance of registration depends on how precisely feature points are matched. The second row shows a box that contains straight lines and complex textural pattern on the plane. The result is reconstructed from 10 different viewpoints. In the scene, most of feature points in the SIFT algorithm are extracted from the textural pattern. Therefore, the registration result of the side of the box, which has lots of textural pattern, looks fine. However, top of the box is incompletely reconstructed since the area has less feature points. At last, the last row shows result of a man, in which feature points are mainly extracted from the body. Transformations for registration are estimated using feature points detected dominantly from the body. The visualization result of the body is fine. The man wears transparent glasses. To visualize transparent region is a difficult problem because the depth and color values are not correctly matched. Therefore, there are some unsatisfactory regions on face.

Fig. 8 shows integration results with different numbers of integrated views. Increasing the number of integrated views decreases the number of holes that do not have object information. Fig. 8(a) has lots of holes on floor and on the back of the lion. On the other hand, in Fig. 8(d), holes are noticeably reduced. Although we filter the acquired depth map, still some mis-aligned pixels remain. As a result, in Figs. 8(b)–8(d) some outlier points are observed on the upper side of lion's mane. There are some artifacts which look like mottled, because we simply take the union of all reliable data in the process of 3-D reconstruction. Some part of 3-D scene may appear in multiple

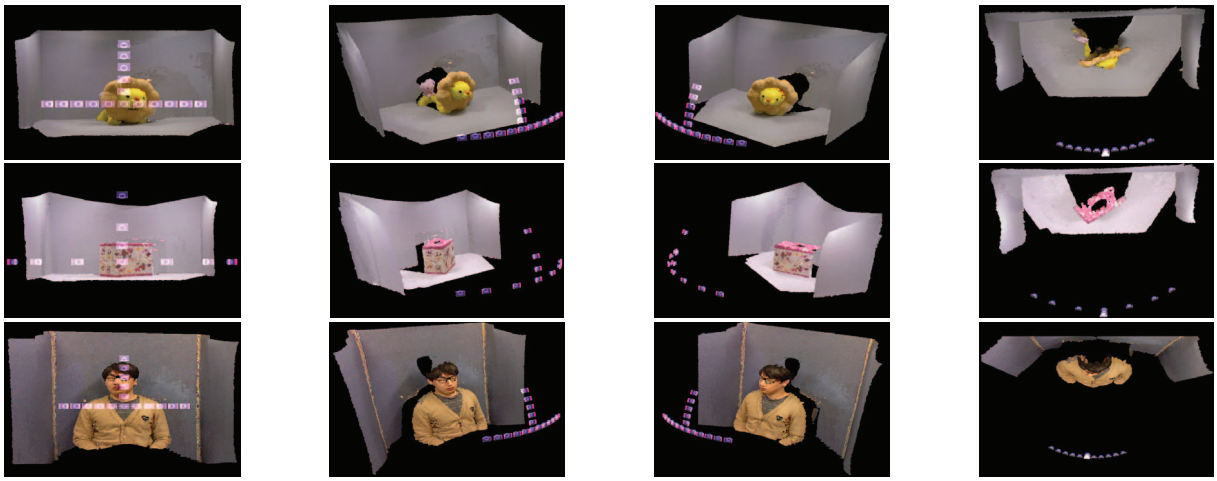
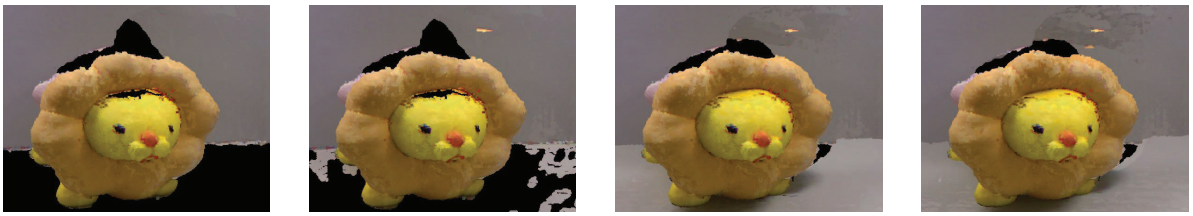


Figure 7. Results of 3-D reconstruction at four different viewpoints (a lion and an octopus, a box, and a man are shown from top to bottom).



(a) (b) (c) (d)

Figure 8. Integration results of different number of integrated views. (a) 11 views, (b) 12 views, (c) 15 views, (d) 16 views.

TABLE I. RENDERING PERFORMANCE

Scene	Number of viewpoints	Number of vertices	Rendering performance	
			Face mode (fps)	*Mouse mode (fps)*
Box	10	1,761,420	13.4	61.8
Lion & octopus	16	2,921,328	8.5	58.6
Man	16	3,875,147	8.4	32.2

images. However, integration process does not remove the same points in multiple images. There are some points which can be clustered in 3-D reconstruction. If we consider the duplicated points, mottled artifacts can be reduced.

In implementing GUI, we have two modes: face and mouse modes. Face mode is to visualize 3-D scenes using position of a face of a user, whereas mouse mode is to visualize 3-D scenes using mouse. In mouse mode, face detection and tracking process are not needed. We summarize experimental results in Table I, which shows a performance of the proposed system. It takes about 57.6ms to acquire 585×443 frame sequence of a face, which takes a large portion of the visualization time whereas face detection time is about 13.4ms. Since mouse mode does not require image acquisition from a webcam and face detection, the frame rate is obviously higher than that of face mode. We implement the proposed 3-D reconstruction method in Matlab whereas face detection based on OpenCV. 3-D visualization is implemented based on OpenGL with a 2.67GHz CPU, 4GB RAM, GeForce GTS 250 graphic card desk top.

IV. CONCLUSIONS

In this paper, we propose a reconstructed 3-D scene visualization system simply using the Kinect sensor and a webcam. The proposed method performs pre-processes to improve the reconstruction performance: color image transformation using statistical information and removal of unstable depth pixels from depth map. We register the pre-processed image using the SIFT to reconstruct a 3-D scene. Then, using the proposed visualization system, a user can use the acquired data in a 3-D visualized plot. The proposed visualization system is a human-computer interactive display system that can be used at a museum or amusement park, causing an interest from audiences.

REFERENCES

[1] J. Chu and C. M. Nie, "Multi-view point clouds registration and stitching based on SIFT feature," in *Proc. Int. Conf. Computer Research and Development*, pp. 274–278, Shanghai, China, Mar. 2011.

[2] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox "RGB-D mapping: Using depth cameras for dense 3-D modeling of indoor environments," in *Proc. Int. Symp. Experimental Robotics*, New Delhi, India, Dec. 2010.

[3] N. Snavely, I. Simor, M. Goesele, R. Szeliski, and S. M. Seitz, "Scene reconstruction and visualization from community photo collections," *Proc. of the IEEE*, vol. 98, no. 8, pp. 1370–1390, Aug. 2010.

[4] J. Francone and L. Nigay, "Using the user's point of view for interaction on mobile devices," in *Proc. 23rd French Speaking Conf. Human-Computer Interaction*, pp. 1–8, New York, Oct. 2011.

[5] J. Smisek, M. Jancosek, and T. Pajdla, "3-D with Kinect," in *Proc. 2011 IEEE Int. Conf. Computer Vision Workshops*, pp. 6–13, Barcelona, Spain, Nov. 2011.

[6] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by Kinect depth camera," in *Proc. 3-DTV Conference*, pp. 1–4, Antalya, Turkey, May 2011.

[7] S.Y. Kim, J. H. Cho, A. Koschan, and M. A. Abidi, "Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor," in *Proc. IEEE Int. Conf. Pattern Recognition*, pp. 2358–2361, Istanbul, Turkey, Aug. 2010.

[8] R. Bruncllli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 10, pp. 1042–1052, Oct. 1993.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. Journ. Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[10] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, Sept./Oct. 2001.

[11] R.C. Gonzalez and R. E. Woods, *Digital Image Processing*, *Third edition*. Upper Saddle River, NJ: Pearson Education Inc., 2010.

[12] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 71–88, Jun. 1981.

[13] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[14] J. Williams and M. Bennamoun, "Simultaneous registration of multiple corresponding point sets," *Computer Vision and Image Understanding*, vol. 81, no. 1, pp. 117–142, Jan. 2001.

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, *second edition*. Cambridge, UK: Cambridge University Press, 2003.

[16] P. Viola and M. Jones, "Robust real-time face detection," *Int. Journ. Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. Conf. Computational Learning Theory: Eurocolt '95*, vol. 904, pp. 23–37, Barcelona, Spain, Mar. 1995.

[18] Open Graphic Library (official website), http://www.opengl.org.

[19] U. Yang and K. Sohn, "Image-based colour temperature estimation for colour constancy," *Electron. Lett.*, vol. 47, no. 5, pp. 322–324, Mar. 2011.