Wide Angle Virtual View Synthesis Using Two-by-Two Kinect V2

Huan-Rui Chang and Hsueh-Ming Hang
Electronics Inst., Electrical and Computer Engr College,
National Chiao Tung University, Hsinchu, Taiwan
Email: jackychang@commlab.tw, hmhang@mail.nctu.edu.tw

***Abstract*—In this paper, we propose a wide-angle view synthesis system based on captured multiple images and depth maps. We use a two-by–two array of Kinect v2 cameras to facilitate virtual camera tilt and zoom-in cases. We integrate two essential components developed by our lab colleagues to form a complete and practical system. Two additional elements are designed and implemented in this study. The first element is an improved multi-view blending algorithm that provides a clear improvement on the synthesized image quality even though the warping process induces some artifacts. By using four RGB-D cameras to capture the reference views, we have sufficient information to decide which pixel in the warped views is erroneous and we replace it by choosing the correct virtual color pixel from the other warped views to generate the synthesized pixel. The second element is solving the asynchronization problem among multiple Kinects in capturing images/videos. For simplicity, our first attempt is implementing a clock calibration system based on PC clock synchronization software. We examine the quality of synthesized views for various camera tilt and zoom-in/out cases. Experimental results show that the proposed multi-view blending algorithm achieves good synthesized image subjective quality on the real world images captured by four Kinect v2 sensors.**

I. INTRODUCTION

Recently, with the development of science and technology, multimedia has increasing more applications in our daily life. For example, 3DTV such as stereoscopic TV, and free-viewpoint television (FTV) can provide a 3D impression to the users. The free-viewpoint television (FTV) can display any viewpoint specified by the users.

Depth image based rendering (DIBR) technique [1, 2] is a popular view synthesis method adopted by the ITU/MPEG standard. DIBR uses the reference color images and depth maps captured from different viewpoints but at the same time instance. The depth information can be calculated from two aligned stereo images or it can be captured by an active Time-of-Flight (ToF) sensor such as Kinect v2 or SR4000. By using the DIBR technique and 3D video coding, the transmitter only transmits the coded color and depth information of the reference views to the receiver. When the receiver receives this information, it applies the decoding procedures to recover the color and depth images/videos. And then the view synthesis algorithm synthesizes the virtual viewpoint pictures by using the decoded color images and depth maps. With the help of view synthesis algorithm, we can generate arbitrary viewpoints for FTV applications.

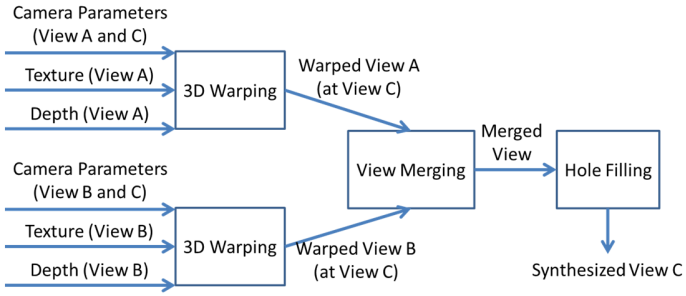

Fig. 1. Flowchart of a basic view synthesis system [1].

Fig. 1 shows the flowchart of a general view synthesis system with two given reference views. The goal is to synthesize a requested target virtual view. 3D warping maps the input reference views into the virtual views using the input camera parameters, reference color images, and depth maps. The view blending procedure aims to merge the images of all warped views and to reduce the disocclusion holes as well. Disocclusion holes are pixels which cannot be seen from one reference view but appears in the virtual view. The remaining holes after view blending will be fixed by the hole filling method.

Most of the conventional view synthesis system typically arranges two or more RGB-D cameras along a baseline to capture the reference views. The conventional view synthesis system typically synthesizes the target virtual view located between two input reference views laying on the baseline. Few view synthesis papers reported 2-dimensional camera array systems. One related work is [3], which presents an experimental FTV system, in which 16 CCD cameras were used in forming a camera array in four-by-four matrix. It adopts an adaptive ray-space data interpolation for view synthesis. The synthesized image quality is good, but the rendered image resolution is low. Furthermore, the camera distance is only 2cm apart and the maximum disparity between adjacent images is 10 pixels. This constraint makes the synthesis cases easier than the wide baseline camera cases. A DIBR method for FTV was demonstrated in [4]. It employs

post-filtering and boundary matting and inpainting techniques were used to fill up holes. An implementation of [4] is adopted as the reference software in MPEG 3D Video standardization process. A residual error feedback scheme was presented in [5]. It notices the error depth pixel problem in image rendering. However, the synthesized virtual view still has some artifacts. Reliability reasoning has been proposed in [6], in which reliability check and artifacts elimination are their main contributions to reduce the commonly seen artifacts. A view synthesis quality refinement algorithm was presented in [7]. Unreliable region extraction, fast backward warping, adaptive blending, and hole filling by using the constructed background models were proposed to produce a high-quality synthesized view.

Recently, a deep learning based view synthesis algorithm was presented in [8], in which a three layers of convolutional neural network were used for patch extraction and feature representation, and hole filling. Reference [9] proposed a view synthesis algorithm based on deep learning for light field cameras, in which two sequential convolutional neural networks were used for disparity estimation and virtual view synthesis, respectively. The synthesized image quality in [9] is good. However, the view synthesis algorithms, as aforementioned, were design for conventional view synthesis (i.e., synthesize virtual view without virtual camera tilt or zoom-in/out and typically with two given input cameras). Furthermore, the parallax between different viewpoints in the dataset used in the training phase of the two deep learning based view synthesis scheme [8, 9] is very small. Hence, they can synthesize virtual view within a small baseline range. In many applications (e.g., watching soccer game in the user specified arbitrary viewpoint) we need to synthesize a virtual view beyond the baseline, which is referred as wide-angle view synthesis. The wide-angle view synthesis aims to synthesize a virtual view that has zoom-in/out or tilt-up/down. In this case, there are often lots of cracks and large disocclusion regions in the synthesized virtual view.

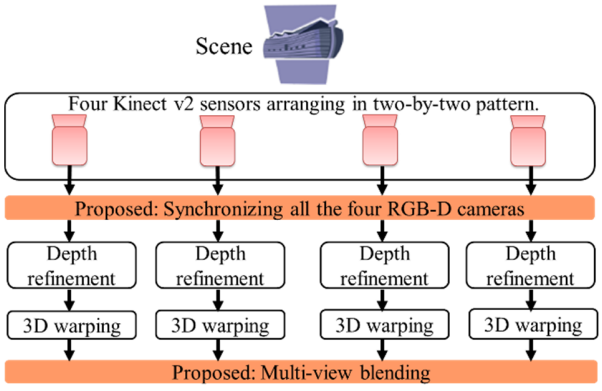

Fig. 2. Flow chart of the proposed wide angle view synthesis system.

In this study, we present a multi-view blending algorithm for wide angle view synthesis. One specific feature of this scheme is that it makes use of four Kinect v2 sensors arranged in the two-by-two matrix manner. The reason we use four Kinect v2 sensors is that a 2-D array of cameras can cover a wider viewing region for accurate view synthesis when it is compared to the popular two camera system or a single-line camera array. However, to take full the advantages offered by a 2x2 camera array, we found that we cannot simply adopt the traditional 1-D camera array methods. Thus, our focus in this study is proposing new multi-view blending algorithm can achieve a good synthesized image quality for more difficult view synthesis cases that involve camera title, zoom-in, and zoom-out. To our knowledge, this is the first proposal that uses a 2x2 RGB-D camera framework for wide-angle view synthesis. The overall system of the proposed 4-camrera wide-angle view synthesis system is shown in Fig. 2. The entire process consists of six steps: (1) camera parameters extraction, (2) multi-camera image rectification, (3) synchronizing all four RGB-D cameras, (4) depth map refinement, (5) 3D warping (of depth maps and color images), and (6) multi-view blending. The rest of this paper is organized as follows. Section 2 describes the proposed wide angle view synthesis system. The experimental results are shown in Section 3. And Section 4 concludes this paper.

II. Proposed View Synthesis System

The flowchart of our wide-angle view synthesis system, as shown in Fig. 2, consists of synchronization of four RGB-D cameras, depth refinement [10], 3D warping [7], and multi-view blending. We integrate two essential components developed by our lab colleagues into the proposed view synthesis system. They are depth refinement algorithm [10] and 3D warping algorithm [7]. We will describe in the rest of this section, step-by-step, the proposed wide-angle view synthesis system.

First, four Kinect v2 devices are placed in a two-by-two pattern in our experiments. Due to the hardware limitations, each Kinect v2 sensor is connected to different computer. In the view synthesis process, it is essential to know the camera parameters. The view synthesis process uses the camera parameters, the captured color images, and depth maps to synthesize the virtual view. Due to the resolution difference in the captured color image (1920*1080) and the depth map (512*424). First step, we use the *MapDepthFrameToColorSpace* function in Kinect SDK to convert/interpolate the depth map to the color image (1920*1080). Next, we estimate the camera parameters of each Kinect. The camera parameters include camera intrinsic and extrinsic parameters. We estimate the camera intrinsic parameter of the four Kinect sensors, individually, using the method described in [11]. In order to estimate camera extrinsic parameter of four Kinect sensors, we capture a checkerboard image that can be seen from all the four Kinect sensors. The camera extrinsic parameter of each Kinect can then be calculated using the extrinsic parameter estimation method provided by [11]. Now, we have the camera intrinsic and extrinsic parameters, we apply the multi-camera image rectification algorithm [12] to obtain the rectified camera parameters. The method in [12] also rectifies the epipolar images.

Using Kinect v2 to record video has a shortcoming: there is no hardware sync signal in Kinect v2 to synchronize multiple Kinect recording. Hence, we need to synchronize these four Kinect v2 sensors for recording each frame at the exactly same instance. We implement a clock adjustment system based on the PC clock synchronization software [14] and control the capturing timing to achieve the synchronization purpose. The synchronization mechanism is only used in synchronizing the color camera of the four Kinect devices. Typically, Kinect captures depth map right after capturing color image. The method of aligning depth map with color frame has been studied and a solution is provided in [10], which will be briefly described in the next paragraph. Hence, we only need to synchronize the four Kinect devices on the color cameras. The importance of synchronization. (i.e., zero time skew in capturing images from different cameras.) will be seen in Section 2 part A.

The quality and resolution of depth map captured by Kinect v2 sensor are noisy. The noisy depth map will jeopardize the synthesized image quality. Therefore, we adopt the depth refinement algorithm proposed by [10] to refine the depth map captured by the Kinect v2 sensor. The flowchart of [10] is shown in Fig. 3, it first constructs the background depth map by using all the color and depth frames of a sequence. Second, the captured color images and depth maps may have a bit of misalignment in the moving object due to Kinect v2 capturing the color image first and then the depth map. Hence, the algorithm in [10] aligns the captured depth map to the captured color image before refining the depth map. Third, the constructed background depth map is used to remove occlusion regions, unreliable depth pixels, holes, and reflection regions. Finally, a bilateral filter is employed to smooth the depth map.

We use the 3D warping algorithm in [7] to warp four reference views to the virtual viewpoint. There are many components in the original algorithm [7], we adopt only the forward depth warping (FDW), the fast backward depth warping (BDW), the fast backward texture warping (BTW), the disocclusion holes detection, and the background color and depth construction in the 3D warping procedure in our system, as shown in Fig. 4. First, FDW is employed to warp the reference color image and depth map to the virtual viewpoint. As we know, FDW often produces cracks, which can be reduced when it is replaced by the BDW [7]. To reduce the computational complexity, BDW is activated only on the selected pixels, and this modified scheme is called fast BDW. After the fast BDW, an opening-by-reconstruction operation is applied to refine the virtual-view depth map. Finally, the warped views can be rendered by using fast BTW and the constructed background color image and depth map.

Up to this step, we already have four warped color images and depth maps. Next, we perform a modified multi-view blending algorithm to produce the final synthesized image/video at the virtual view. In Section 2 part C, we will elaborate on the details of the proposed multi-view blending algorithm.

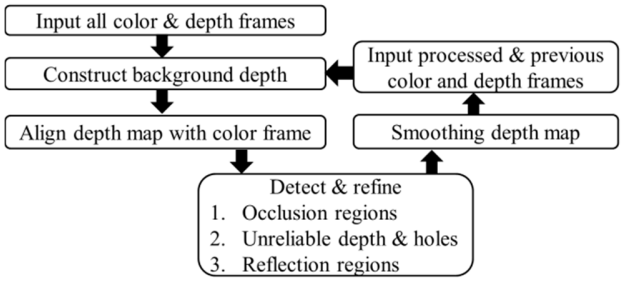

Fig. 3. Flow chart of depth refinement algorithm [4].

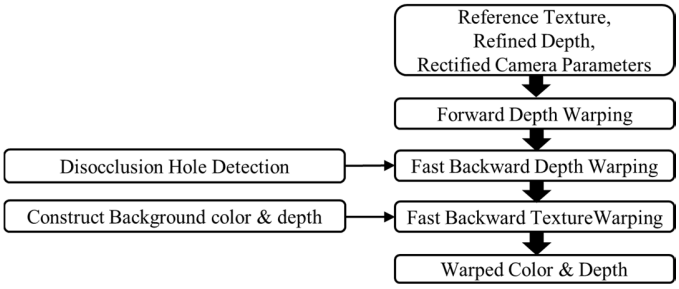

Fig. 4. Flow chart of the 3D warping process [5].

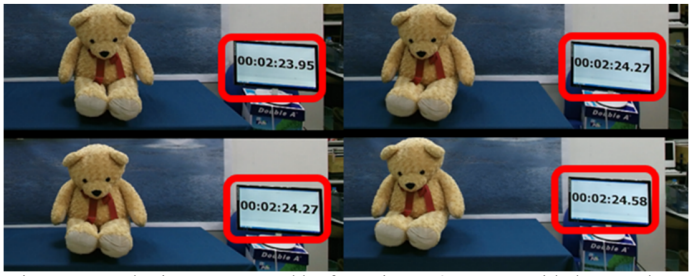

Fig. 5. Four color images captured by four Kinect v2 cameras with the asynchronization problem in a video sequence.

A. Synchronization Problem of Four RGB-D Cameras

For a moving object, if four Kinect v2 devices cannot record at the same time, the captured scenes of four cameras are not exactly at the same location in the recorded images. This asynchronization problem violates the basic assumption of DIBR [2], and thus cannot produce the desired good quality synthesized images. Therefore, we need to synchronize these four Kinect v2 devices so that they are recording each frame at exactly the same time instance. Otherwise, the warped virtual views have an obvious ghost effect on the moving objects. Fig. 5 shows four color images captured from four Kinect v2 cameras with asynchronization problem in a video sequence.(The toy bear moves cross the table top in our video sequence.) A large capturing time skew between four Kinect devices induces serious ghost effects in the synthesized image.

Reference [13] proposed a method that enables four Kinect v2 sensors to communicate via TCP/IP and Ethernet network. However, we still noticed minor time skew due to the communication delay. For simplicity, we use a system time calibration software, NTPClock.exe [14], developed by National Time and

Frequency Standard Laboratory. It checks with https://time.is/ to make sure the system time of the four computers is fully synchronized. As mentioned in second line of the second paragraph of Section 2, four Kinect v2 sensors are connected to different computers due to the hardware limitations. The NTPClock.exe software calibrates the system time clock of the computer. It can achieve (nearly) zero second time skew with the national standard time. Then, we control the four Kinect v2 cameras to start recording at the same system time and capture the color images and depth maps at every 0ms of system time.(The system time of each computer can be extracted by using *GetLocalTime (&sys)* command in Visual Studio C++ environment.) Hence, we can successfully synchronize all four cameras recording at the same instance, as shown in Fig. 6. The current drawback is the frame rate is only 1 fps because we trigger the camera to capture images at each 0ms. If we trigger the camera at a higher rate, sometimes frames are miss-captured. This problem may be solved by using RAMDrive mechanism and to be further investigated.

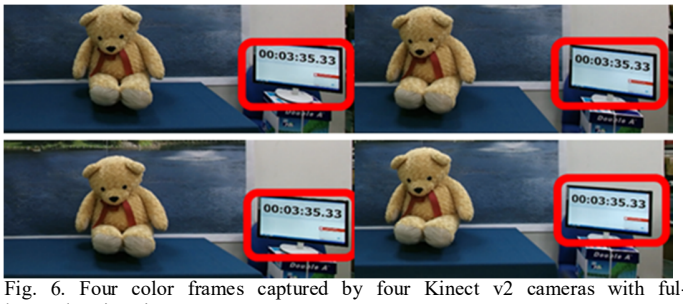

Fig. 6. Four color frames captured by four Kinect v2 cameras with fully-synchronizatyion.

B. *Alignment Problems of Four Warped Views*

Before describing the proposed multi-view blending algorithm, we first look into the alignment problems of the four warped views. As mentioned earlier, we align the depth map with its associated color image in the depth refinement phase by using the method proposed in [10]. However, there is from time to time some remaining misaligned pixels between color image and depth map around the object boundary. The first problem is that the misalignment between the reference color image and depth map leads to wrongly warped pixels after 3D warping because the color pixels use the wrong associated depth value in warping procedure, as shown in Fig. 7. We call this phenomenon, *texture-depth misalignment*. The ghost-image-like foreground object pixels appear (next to the black gaps) due to *texture-depth misalignment*, as shown in Fig. 8.

The second problem is the slight misalignment between four warped color images, as shown in Fig. 9. We call this problem, *texture-texture misalignment*. Most of the pixels of four warped views have *texture-texture misalignment* because it is hard to have perfectly accurate camera parameters and perfect depth maps. This problem produces a light ghost effects on the blended pictures and it blurs the object boundaries and also the textures in the depth smooth regions, as shown in Fig. 10.

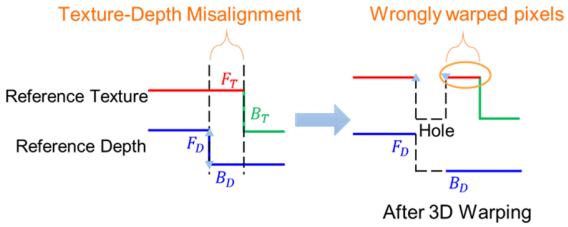

Fig. 7. Illustration of misalignment between reference color and depth images can cause wrongly warped pixels [7]. "F" means foreground. "*B*" means background. The subscript "*T*" and "*D*" means texture and depth images, respectively.

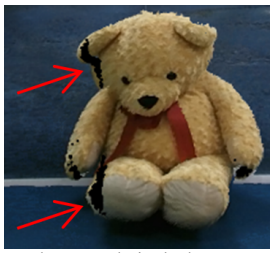

Fig. 8. Example of the wrongly warped pixels due to *texture-depth misalignment*.

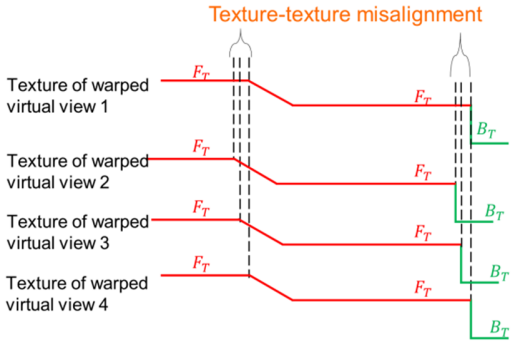

Fig. 9. Illustration of misalignment between four warped virtual views.

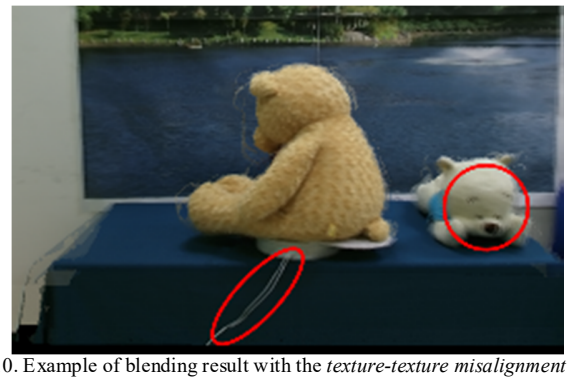

Fig. 10. Example of blending result with the *texture-texture misalignment* effect.

C. *Proposed Multi-view Blending Algorithm*

The proposed multi-view blending algorithm consists of three parts. We have observed that the depth value of object boundaries suffers *texture-depth misalignment* and *texture-texture misalignment*. The smooth regions typically only has *texture-texture misalignment*. Hence, for synthesizing pixels near depth bounda-

ries, we first extract the depth edge from the reference depth map, dilate the depth edge 10-pixel-width (pre-warp dilation), warp the dilated depth edge to the virtual view position by FDW, and dilate 3-pixel-width (post-warp dilation). Finally, we mark those pixels as "True" in the *DepthEdge* variable, as shown in Fig. 11; otherwise, mark it as "False" in *DepthEdge* variable. The extracted *DepthEdge* will be used in the proposed multi-view blending algorithm part 1 and 2 for rendering the virtual pixels near the depth edge. The reason we do pre-warp dilation and post-warp dilation is that the warped edge is not aligned precisely with the depth edge of virtual view; we widen the warped depth edge coverage by both pre-warp and post-warp dilations.

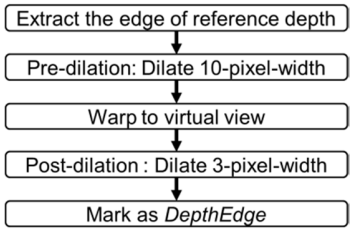

Fig. 11. Flow chart of extracting *DepthEdge*.

The proposed multi-view blending algorithm renders the virtual view color and depth images based on the notion of the dominant virtual view. The dominant virtual view is chosen according to the position of virtual camera and is denoted as "dom". We choose the warped view whose original reference camera position is closest to the virtual camera. The remaining three warped views are denoted as rem1, rem2, and rem3. Fig. 12 gives an example, the position of virtual camera is closer to camera 3. Thus, we choose the warped view of camera 3 as the dominant virtual view (denoted as "dom") and the warped views of the remaining three cameras are denoted as rem1, rem2, and rem3. The pixels in the dominant virtual view would be used to synthesize the virtual view if they satisfy certain criteria stated in the following subsections. Fig. 13 is an overview of the proposed multi-view blending algorithm. The input of our algorithm is the warped color images and depth maps. There are three parts in the proposed multi-view blending algorithm. The following three subsections *C.1*, *C.2*, and *C.3* describe the details of the proposed multi-view blending algorithm part 1, 2, and 3, individually.

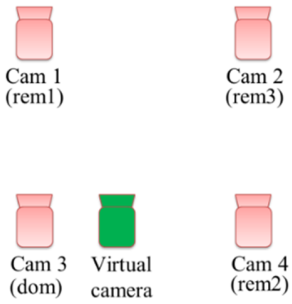

Fig. 12. Illustration for choosing dominant virtual view (dom) and rem1, rem2, and rem3.

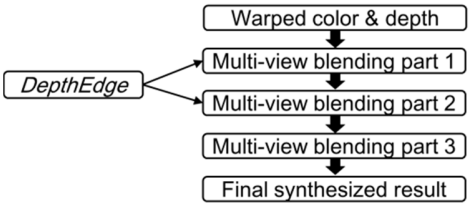

Fig. 13. Flow chart of the proposed multi-view blending algorithm.

C.1. Proposed Multi-view Blending Algorithm Part 1

The operation of Multi-view Blending Algorithm Part 1 is shown in Fig. 14. The purpose of Multi-view Blending Algorithm Part 1 is to effectively solve the *texture-texture misalignment* problem and maintain the synthesized image sharpness in the smooth depth regions by using the dominant virtual view in rendering pixels away from the depth edges (i.e., *DepthEdge*(u,v)=False). The condition is that the depth value of dominant virtual view at that pixel is valid (i.e., $D_{dom}(u,v)$ exist), then the dominant view color pixel value is assigned to the virtual view pixel.

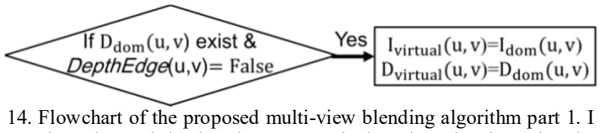

Fig. 14. Flowchart of the proposed multi-view blending algorithm part 1. I and D indicate the color and depth values, respectively. The subscript "virtual" means the virtual view. The subscript "bk" is the associated background. The same notations are used below. "u" and "v" are the coordinate in x-axis and y-axis, respectively.

C.2. Proposed Multi-view Blending Algorithm Part 2

The flowchart of Multi-view Blending Algorithm Part 2 is shown in Fig. 15. Its purpose is to prevent the false rendering due to the misalignment of four warped virtual views (*texture-texture misalignment*) near the depth edge. Second, it also removes the wrong rendering when the color and depth of the reference view do not match well (due to *texture-depth misalignment*). As a result, it fixes the wrong pixels in depth edge regions. By using four RGB-D cameras, we have sufficient information to decide which pixel in the warped views is incorrect and we remove it by choosing the correct virtual color pixel from the other warped virtual views.

This part handles the virtual view pixels near the depth edge (i.e., *DepthEdge*(u,v)=True). We first check the difference between the depth of dominant virtual view and its associated background depth. If the difference in depth value is less than 5, then we record the color value of dominant virtual view in I_{temp}. Then we check and record the remaining three warped depth pixels in the same manner. If only the dominant virtual color is recorded in I_{temp}, then we skip this part and go to multi-view blending algorithm part 3 directly. Otherwise, we increase the weight when the color difference among the recorded colors is less than 20, as shown in the middle of Fig. 15. The variance and mean values are computed based on the color values of largest weight.

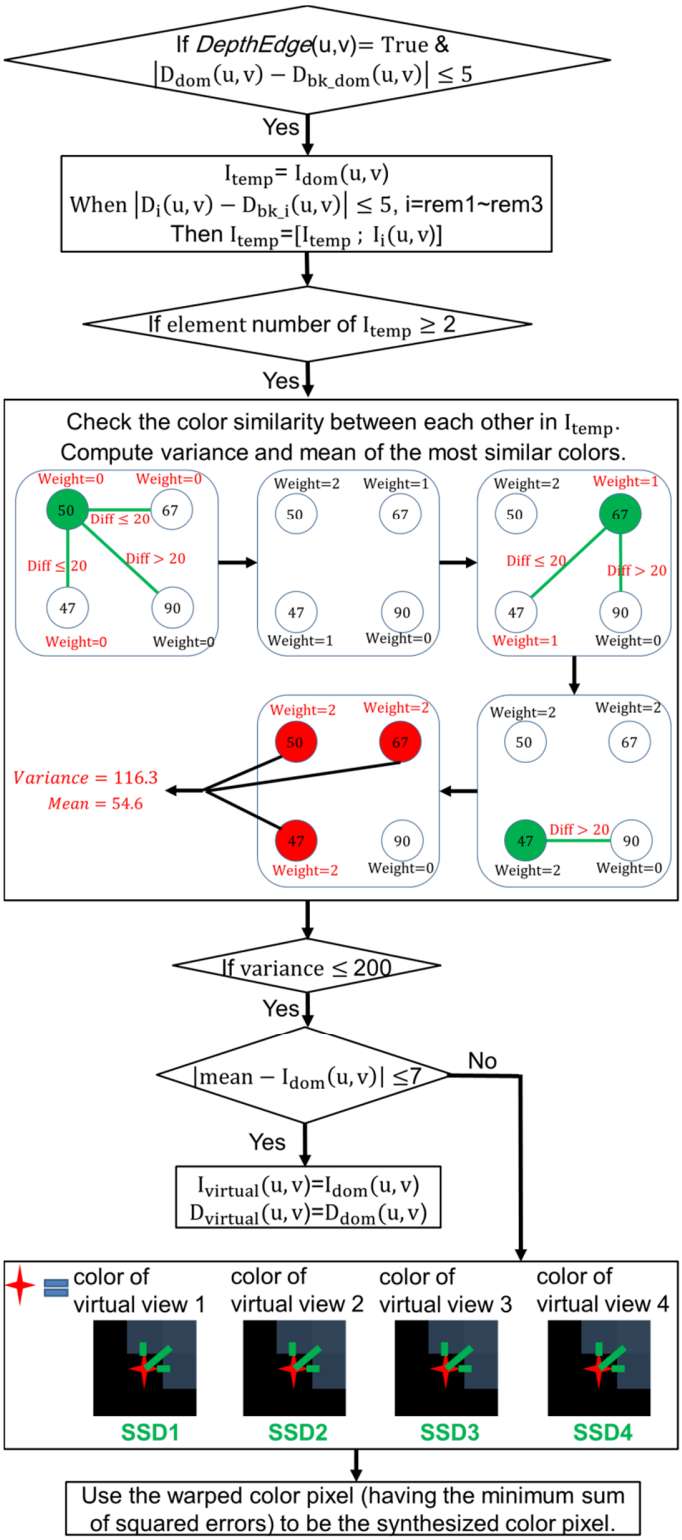

Fig. 15. Flow chart of the proposed blending algorithm part 2. "SSD" means sum of square differences.

If the variance less than 200 (i.e., the retained colors have similar color), and the difference between mean and the dominant virtual color is less than 7 (i.e., the retained colors are similar to the dominant virtual color), we render the virtual-view color and depth by the dominant virtual color and depth. If the difference between mean and the dominant virtual color is larger than 7 (i.e., either the dominant color or the retained colors is wrong), we then choose one of the warped virtual colors that has the minimum sum of square difference (SSD) compared to the valid 8-neighbor colors of the synthesized virtual pixel. If a pixel has an invalid depth value, it is not counted as a valid 8-neighbor.

C.3. Proposed Multi-view Blending Algorithm Part 3

The flowchart of Multi-view Blending Algorithm Part 3 is shown in Fig. 16. The purpose is to eliminate the wrong pixels caused by *texture-texture misalignment* in the depth edge region. So far, the remaining depth holes of the rendered virtual view belong to three cases. First, the pixel of dominant virtual view does not exist. Second, the matched warped views (i.e., element number of I_{temp}) in the multi-view blending algorithm part 2 is less than 2. Third, the variance value in the blending algorithm part 2 is larger than 200. Hence, we fill in the pixel using one of the warped color views, whose color has the minimum sum of square difference (SSD) to the valid 8-neighbor color pixels of the target pixel. This is an iterative procedure. In each iteration a certain number pixels are filled in using the above procedure. The valid pixel number thus increases iteratively. Therefore, more pixels are nailed down in the next iteration. If the 8-neighbor color pixels are all invalid, we skip the rendering process in the current pixel and render it when there exist a valid color in the 8-neighbor afterwards.

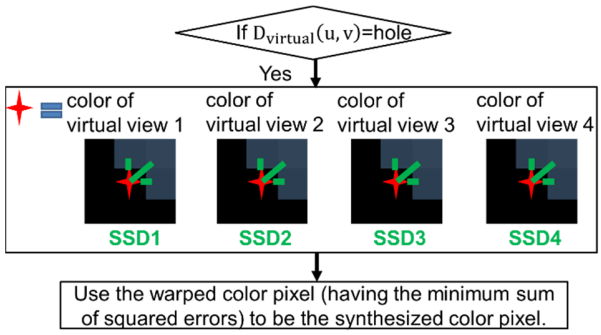

Fig. 16. Flow chart of the proposed blending algorithm part 3.

III. Experimental Results

In this study, our goal is to design a wide angle view synthesis system which can be applied to the real world cases. Hence, we captured videos and depth maps using four Kinect v2 sensors in two-by-two matrix pattern, as shown in Fig. 17. The traditional view synthesis standard test sequences did not provide a 2-D array RGB-D sequences.

To evaluate the effectiveness of the proposed multi-view blending algorithm, we capture two video sequences in our Lab.

The first test sequence is a brown toy bear moving cross the table top; it consists of 7 frames. The second test sequence is a brown toy bear rotating and a white toy bear stays stationary on the table; it consists of 49 frames. We show the need of a synchronization mechanism in subsection 3.A. Subsection 3.B shows the experimental results of our multi-view blending algorithm and the comparison of our results to adaptive blending algorithm in [7].

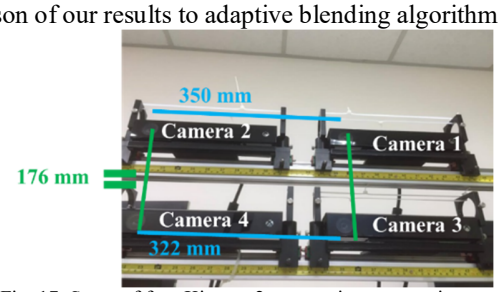

Fig. 17. Setup of four Kinect v2 sensors in our experiments.

A. Experimental results of Synchronization

To visualize the effect of asynchronization among four cameras, Fig. 18(a)(b) shows two synthesized virtual images with asynchronization problem. If the maximum capturing time skew of four cameras is 0.12 seconds, the ghost effects is obvious as seen in Fig. 18(a). When the capturing time skew of four cameras grows to 0.81 seconds, the synthesized image suffers serious ghost effects, as shown in Fig. 18(b). The proposed synchronization scheme in this study can reach a good synchronization for four Kinect v2 devices. The synthesized image aligns well for moving objects, as shown in Fig. 18(c).

(a) (b) (c)

Fig. 18. Examples of synthesized image with asynchronization problem and the maximum capturing time skew of the four Kinect cameras is (a) 0.12, and (b) 0.81 seconds. (c) Example of synthesized image using synchronized four Kinect cameras.

B. Experimental results of Multi-view Blending Algorithm

Fig. 8 and Fig. 10 show the wrongly warped pixels caused by *texture-depth misalignment* and *texture-texture misalignment*, respectively. The proposed multi-view blending algorithm can remove these incorrect pixels and produce a good synthesized image subjective quality. We show the synthesized results of two test sequences at two virtual camera positions. Fig. 19(a) shows the geometric location of the virtual camera with respect to the four reference cameras, and the virtual camera has a 5-degrees-tilt-down and 100mm zoom-in. Fig. 19(b) shows the second virtual camera location with only zoom-in or zoom-out cases. The synthesized results using the adaptive blending algorithm [7] with two upper cameras are shown in Fig. 20(a)(d), Fig. 21(a)(d), and Fig. 22(a)(d). The synthesized images using the proposed multi-view blending algorithm with two upper cameras are shown in Fig. 20(b)(e), Fig. 21(b)(e), and Fig. 22(b)(e). The synthesized images using the proposed multi-view blending algorithm with all four cameras are shown in Fig. 20(c)(f), Fig. 21(c)(f), and Fig. 22(c)(f). The cropped details of the synthesized images are also shown in Fig. 20, Fig 21, and Fig. 22 on the right hand side of each case.

(a) (b)

Fig. 19. Geometric location of the virtual camera (with respect to four reference cameras) (a) with a 5-degrees-tilt-down and 100mm zoom-in; (b) with only zoom-in or zoom-out.

The synthesized results using the adaptive blending technique [7] cannot handle the wrongly warped pixels due to *texture-depth misalignment* near the object boundaries, as shown in Fig. 20(a)(d), Fig. 21(a)(d), and Fig. 22(a)(d). Using the proposed multi-view blending algorithm with the input cameras same as adaptive blending, we can successfully remove the wrongly warped pixels near the object edge, as illustrated in Fig. 20(b)(e), Fig. 21(b)(e), and Fig. 22(b)(e). However, there are some wrongly synthesized pixels near the object edge. The blue pixels under the toy bear's right hand are the wrongly synthesized pixels, as shown in Fig. 20(b). This is because this region is an occlusion region. If 4 input cameras are used, the proposed blending algorithm with four input cameras can effectively remove both the *texture-depth misalignment* and *texture-texture misalignment* problems, and also fix the occlusion regions, as illustrated in Fig. 20(c)(f), Fig. 21(c)(f), and Fig. 22(c)(f). To visualize the effectiveness of our proposed multi-view blending algorithm can robustly suppress the *texture-texture misalignment* problem, Fig. 10 is the synthesized result with *texture-texture misalignment* problem while the proposed multi-view blending algorithm can handle this problem, as illustrated in Fig. 20(f) and Fig. 22(f).

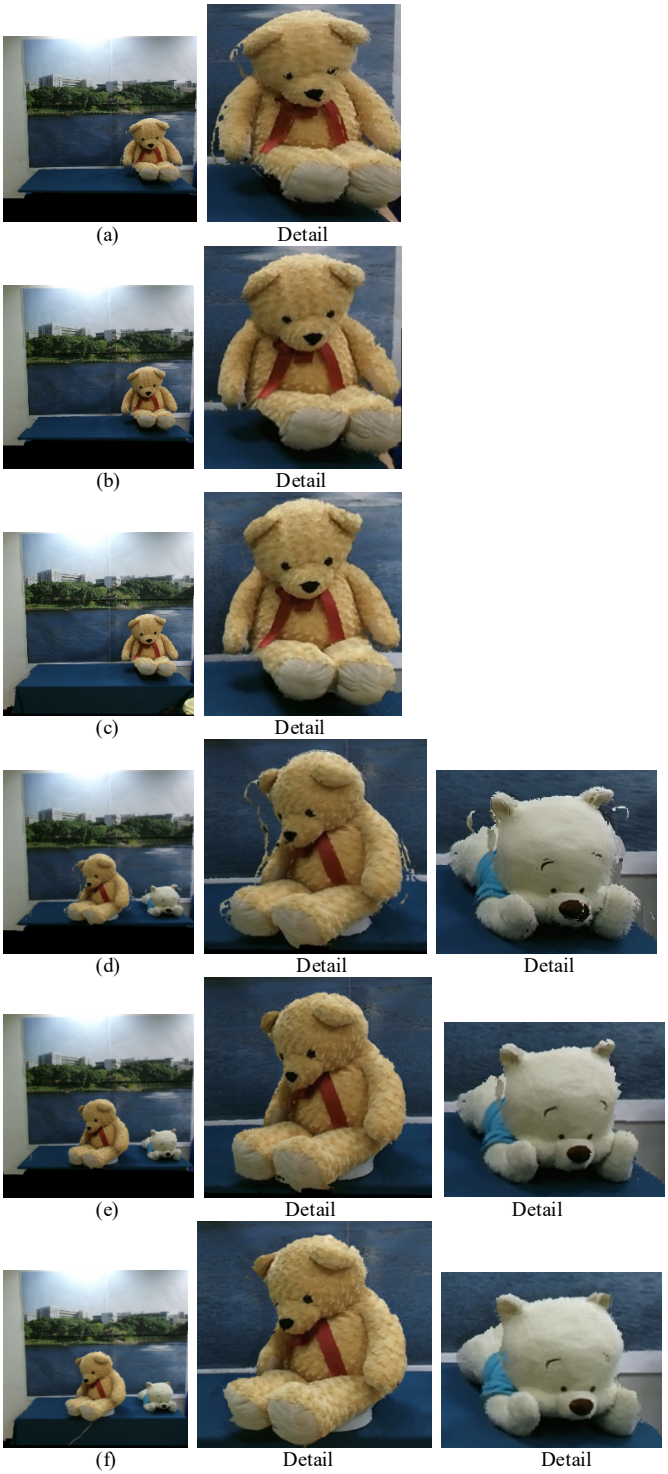

Fig. 20. Blending results of virtual camera with a 5-degrees-tilt-down and 100mm zoom-in using adaptive blending algorithm [7] (a)(d) with two upper cameras; proposed blending algorithm (b)(e) with two upper cameras, and (c)(f) with four cameras. The blending results of first sequence are (a)(b)(c). The blending results of second sequence are (d)(e)(f).

Fig. 21. Blending results of virtual camera with (a)(b)(c) 200mm and (d)(e)(f) 400mm zoom-in using adaptive blending algorithm [7] (a)(d) with two upper cameras; proposed blending algorithm (b)(e) with two upper cameras, and (c)(f) with four cameras. The blending results of first sequence are (a)(b)(c). The blending results of second sequence are (d)(e)(f).

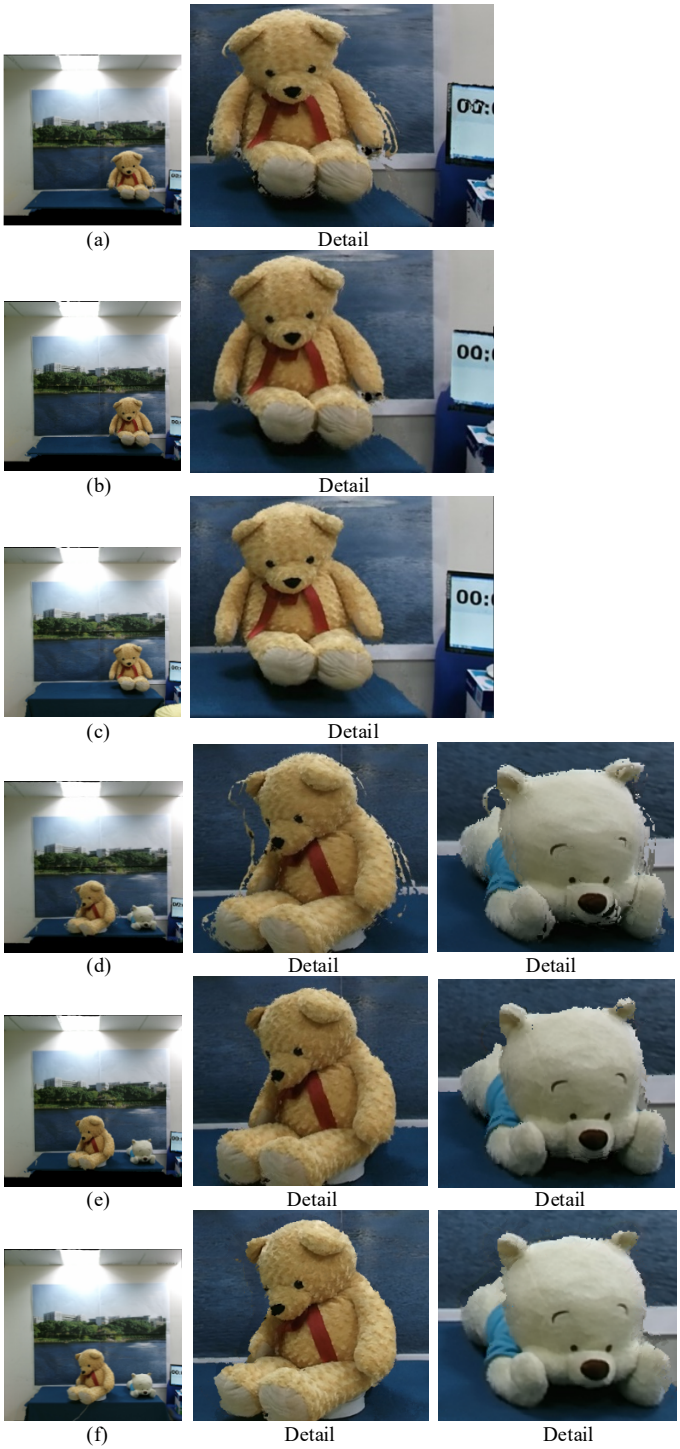

Fig. 22. Blending results of virtual camera with a 400mm zoom-out using adaptive blending algorithm [7] (a)(d)with two upper cameras; proposed blending algorithm (b)(e) with two upper cameras, and (c)(f) with four cameras. The blending results of first sequence are (a)(b)(c). The blending results of second sequence are (d)(e)(f).

IV. CONCLUSIONS

In the practical view synthesis cases where the reference videos and depth maps are captured by the consumer-grade cameras, the depth map contains noises, holes and occlusion regions. We implement a clock adjustment system based on PC clock synchronization software to synchronize four Kinect v2 sensors in video recording. This is essential to prevent the synthesized virtual image suffering from ghost effect on moving objects. Furthermore, the main contribution in this paper is that we have developed a new multi-view blending algorithm that can produce a good synthesized image quality when the warped four views are not perfect. This is the practical situation we encounter in capturing pictures using the commercial RGB-D cameras. Compared to the adaptive blending method [7], the proposed multi-view blending algorithm is robust against the texture-depth misalignment and texture-texture misalignment in the warped views. Experimental results show that we can achieve better subjective synthesized image quality in these real cases.

ACKNOWLEDGMENT

This work was supported in part by the NSC, Taiwan under Grant NSC 103-2221-E-009-065-MY3 and by the Aim for the Top University Project of National Chiao Tung University, Taiwan.

REFERENCES

[1] C. Zhu, Y. Zhao, L. Yu, and M. Tanimoto, 3D-TVSystem With Depth-Image-Based Rendering: Architectures, Techniques and Challenges, 2013 :Springer

[2] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3DTV," in *Electronic Imaging* 2004, pp. 93-104.

[3] P. N. Bangchang, T. Fujii, and M. Tanimoto. "Experimental system of free viewpoint television."*Electronic Imaging* 2003. *International Society for Optics and Photonics*, p. 554-563, 2003.

[4] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto. "View generation with 3D warping using depth information for FTV". *Signal Processing: Image Communication*, 24(1), 65-72, 2009.

[5] H. Furihata, T. Yendo, M. P. Tehrani, T. Fujii, and M. Tanimoto. "Novel view synthesis with residual error feedback for FTV." *IS&T/SPIE Electronic Imaging*, 2010.*International Society for Optics and Photonics*, p. 75240K-75240K-12, 2010.

[6] L. Yang, T. Yendo, M. P. Tehrani, T. Fujii, and M. Tanimoto. "Artifact reduction using reliability reasoning for image generation of FTV", *Journal of Visual Communication and Image Representation*, *21*(5), 542-560, 2010.

[7] T.-C. Lee, C. –L. Chien, and H.-M. Hang, "Virtual view synthesis using quality refiement," in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, July 2016.

[8] H.-T. Lim, H. G. Kim, and Y. M. Ro "Learning based hole filling method using deep convolutional neural network for view synthesis," *IS&T International symposium on Electronic Imaging (EI)*, Feb 2016

[9] N. K. Kalantari, T. –C. Wang, and R. Ramamoorth, "Learning-Based View Synthesis for Light Field Cameras," *Journal ACM Transactions*

on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2016, Volume 35 Issue 6, Article No. 193, Nov. 2016.

[10] K.-Y. Lin, “Depth Map Enhancement on RGB-D Video Captured by Kinect V2”, MS thesis, NCTU, July, 2017.

[11] Camera Calibration Toolbox for Matlab. Available software at http://www.vision.caltech.edu/bouguetj/calib_doc/

[12] J. Yang, F. Guo, H. Wang, and Z. Ding, "A multi-view image rectification algorithm for matrix camera arrangement", *Artificial Intelligence Research*, Vol. 3, No. 1, pp18~29, November 2013.

[13] M. Kowalski, et al, "Livescan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors," 2015 *International Conference on 3D Vision*, pp. 318-325, 2015.

[14] Computer system time calibration software. Available at http://www.stdtime.gov.tw/chinese/EXE/NTPClock.exe