Region-Aware 3-D Warping for DIBR

Jian Jin, Anhong Wang, Yao Zhao, *Senior Member, IEEE*, Chunyu Lin, and Bing Zeng, *Fellow, IEEE*

***Abstract*—In 3-D video (3DV) applications, depth-image-based rendering (DIBR) has been widely employed to synthesize virtual views. However, this approach is performed in a frame-based way, meaning each whole frame is dealt with and the characteristics of different regions in the frame are ignored. As a result, redundant pixels in some regions are abused during the subsequent warping and blending stage. This paper proposes a region-aware 3-D warping approach for DIBR in which warped frames are reasonably divided beforehand so that only the indispensable regions are used. With the proposed scheme, it is possible to avoid noneffective and repeated pixels during the warping stage. In addition, the blending process is also saved. The experimental results show that compared to the state-of-the-art VSRS3.5 and VSRS-1D-fast algorithms, our approach can achieve significant computation savings without sacrificing synthesis quality.**

***Index Terms*—3-D video (3DV), depth-image-based rendering (DIBR), multi-view, region partitioning, region-aware 3-D warping.**

I. Introduction

THREE-DIMENSIONAL video (3DV) is becoming increasingly popular because it offers an immersive experience of real 3-D scenarios. A typical approach to implementing 3DV is to shoot 3-D scenarios using several cameras at different viewing angles. To support more viewing angles in this circumstance, it is important to generate some (or many) so-called virtual views. This process is called the view synthesis, which often exploits the depth-image-based rendering (DIBR) technique [1].

The core of DIBR is to warp pixels from the original views to synthesized views by utilizing the depth information and the cameras' parameters. To this end, two DIBR approaches have been proposed [2]: the single-view approach and the multiple-view approach. A single-view approach proposed by Oliveira *et al.* [3] utilizes a single original view plus its depth information to extrapolate the synthesized view. However, this approach has a drawback: It often leads to large holes in the synthesized view because the corresponding information of the background is occluded by the foreground objects in the original view. A multiple-view approach proposed by Min *et al.* [4] makes use of two original views (the left and right) and their depth [5] to interpolate the intermediate views. This way, the occluded background can be filled using the available background from either the left or right view. Clearly, the multiple-view approach can generate better synthesized views compared to the single-view scheme. Hence, the multiple-view approach will be adopted in our study presented in this paper.

Recently, the 3DV Coding Team of ISO/IEC's Moving Pictures Experts Group (MPEG) developed the multiple-view approach in the view synthesis reference software (VSRS) [6]. However, it is still time consuming in practical 3DV applications. For example, using two views to synthesize one virtual view, 120 frames (including texture and depth frames from both views) need to be decoded and 60 frames of the virtual view need to be synthesized per second.

Many approaches have been proposed to speed up the DIBR-based VSRS. For instance, Horng *et al.* presented a hardware solution for view synthesis that achieves 32 frames/s for HD1080 video [7]. In view of the software solution, many researches have focused on 3-D warping because it is the most time-consuming operation in DIBR. To this end, Tsung *et al.* provided a view interpolation scheme for multi-view video applications, which reduces 86% of the 3-D warping operations [8]; Chen proposed a shape-adaptive low-complexity technique for removing spatial computation redundancy of view synthesis and saved as much as 96% computation for view synthesis [9]; and Vijayanagar *et al.* presented a simple but highly efficient view synthesis process, which has significantly optimized the process of DIBR [10].

Vijayanagar's method is fast and has good objective quality. Furthermore, it is inherently parallel, thus making it suitable for high-speed view synthesis on a GP-GPU. Therefore, it has been developed into the latest VSRS-1D-fast algorithm [11] as a part of JCT3V based on high efficiency video coding (HEVC). This 1-D algorithm is an optimized view-synthesized algorithm for the 1-D parallel model in which the virtual view is aligned vertically with the original views; thus, only horizontal disparity needs to be considered. Compared to the conventional VSRS, it calculates the disparity by utilizing a separable look-up table-based technique that considerably speeds up the 3-D warping process. Additionally, a novel blending approach based on similarity maps can further improve the quality of the virtual views [12]. In our study, we choose VSRS3.5 [13] (which is considered

Manuscript received August 31, 2015; revised December 26, 2015; accepted February 23, 2016. Date of publication March 08, 2016; date of current version May 13, 2016. This work was supported by the National Natural Science Foundation of China under Grant 61210006, Grant 61402034, and Grant 61272262, by the Beijing Natural Science Foundation under Grant 4154082, by the Fundamental Research Funds for the Central Universities under Grant 2015JBM032, by the International Cooperative Program of Shanxi Province under Grant 2015081015, by the Scientific and Technological project of Shanxi Province under Grant 2015031003-2, by the Shanxi Scholarship Council of China under Grant 2014-056, and by the Program for New Century Excellent Talent in Universities under Grant NCET-12-1037. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Cha Zhang. *(Corresponding author: Anhong Wang.)*

J. Jin, Y. Zhao, and C. Lin are with the Institute of Information Science, Beijing Jiao Tong University, Beijing 100044, China, and also with the Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China (e-mail: jianjin@bjtu.edu.cn; yzhao@bjtu.edu.cn; cylin@bjtu.edu.cn).

A. Wang is with the School of Electronic Information Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China (e-mail: wah_ty@163.com).

B. Zeng is with the Institute of Image Processing, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: eezeng@uestc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2539825

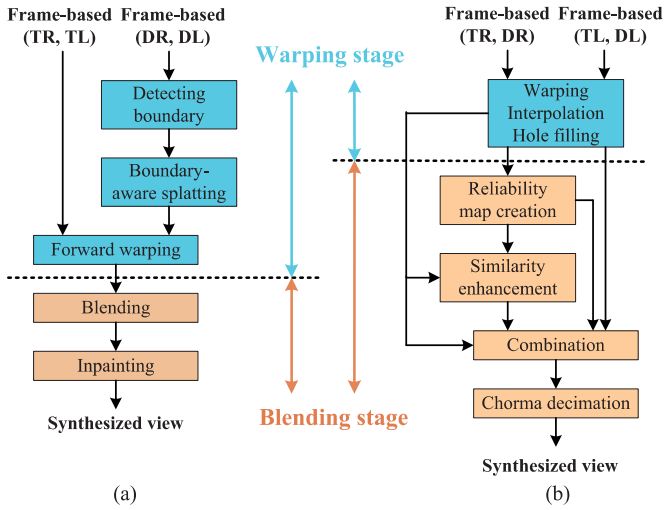

Fig. 1. Two main stages in VSRS3.5 and VSRS-1D-fast under the 1-D parallel model, where TR, TL, DR, and DL are "texture right," "texture left," "depth right," and "depth left," respectively.

the most widely used version of VSRS) and VSRS-1D-fast [14] (regarded as the state-of-the-art) as the benchmark for comparison in view rendering [15].

All methods mentioned above are frame-based synthesis methods. This means that some unnecessary information in certain regions of a frame, which contributes very little to the view synthesis, will still be used in the warping process. Inspired by this observation, a novel region-aware 3-D warping scheme is proposed in this paper. In this scheme, we consider the characteristics of different regions in the original views when warping them into the virtual view and then perform a 3-D warping selectively so that the process can be much more efficient without degrading the quality of synthesized views. Experimental results show that our proposed scheme can achieve significant computation savings with negligible quality degradation compared to the existing algorithms.

The remaining parts of this paper are organized as follows. We review the framework of VSRS3.5 and VSRS-1D-fast in Section II. Then, we analyze the characteristics of different regions in Section III and propose the region-aware 3-D warping scheme in Section IV. Experiments are conducted in Section V to evaluate our region-aware method. Finally, some conclusions and discussions are presented in Section VI.

II. VSRS3.5 and VSRS-1D-Fast: A Brief Review

VSRS3.5 (assuming the "1-D model") and VSRS-1D-fast are quite similar to each other, and both contain two main stages, as shown in Fig. 1: 1) the warping stage, in which all pixels in the original views are warped to the virtual view based on the depth information, and 2) the blending stage, in which the warped views are blended together.

At the warping stage, pixels from the original views are transformed into the virtual view. In VSRS3.5, this stage is further divided into two steps: 1) boundary detecting and boundary-aware splatting are first carried out in the depth map and 2) forward warping is performed. During the forward warping step, a 3-D warping function [16] is used. However, mapping competition, which means that different pixels in the original views are mapped to the same point in the virtual view, is unavoidable. To solve this problem, the so-called z-buffer is usually adopted in VSRS3.5 [17], where the front-most pixel is selected for the virtual view to maintain the logical occlusion order.

By contrast, the warping stage of VSRS-1D-fast is quite simple: Once the depth information is utilized to map the original view to the virtual view, the corresponding texture information will follow. During this stage, three steps, i.e., warping, interpolation, and hole filling, are carried out together. To simplify this stage, the 3-D warping function is not used. Instead, a separable look-up table-based technique is used. In addition, to overcome the mapping competition, the 3-D warping of different original views is performed from different directions in VSRS-1D-fast; e.g., the left original view is warped from left to right, while the right one is warped from the opposite direction [11]. Mapping competition often occurs in regions that belong to the background and can be seen from (one of) the original views, but are occluded by the foreground and thus cannot be seen from the virtual view. Therefore, these regions are not needed for the virtual view. If we can identify all regions that are not required to be warped, certain or many computations would be saved. In addition, if the mapping competition can be prevented, we can further save the computation consumed for the z-buffer. Another issue in the 3-D warping process is the visible range. In the 1-D parallel model, there are some regions that are visible in the original views, but are out of the visible range of the virtual view. These regions are not needed for the virtual view either and can thus be excluded from 3-D warping to save computation.

The blending stage merges two warped views into a single virtual view. In VSRS3.5, this stage contains two steps: 1) blending two warped views into the virtual view and 2) inpainting the virtual view [18]. In VSRS-1D-fast, this stage contains four steps: 1) creating reliability maps, pixels of the warped views (generated in the warping stage) are given different values according to their reliabilities (e.g., disocclusion holes are unreliable with value 0). Then, two reliability maps are created; 2) enhancing similarity, the most reliability pixels in the two warped views will be used to generate two histograms, respectively. Then, the enhancement processing is carried out through adapting one histogram to another; 3) combining the warped views. According to the reliability maps, the warped views are combined; 4) the chroma of the combined results is decimated. Although their blending stages are different in terms of the details, both share two similar configurations: 1) blending two warped views by a linear weighting function and 2) choosing one warped view as the dominant view and filling the holes by the other warped view. They both have their advantages and disadvantages; see [11], [19] and [20] for detailed discussions.

After 3-D warping, there are three types of holes: 1) disocclusion holes, 2) boundary holes, and 3) cracks. These three different types of holes have been discussed in [21]. Disocclusion holes are regions that are occluded by the foreground in the original views but become visible in the virtual view. These holes are commonly generated by sharp depth discontinuities between the foreground and the background in the original

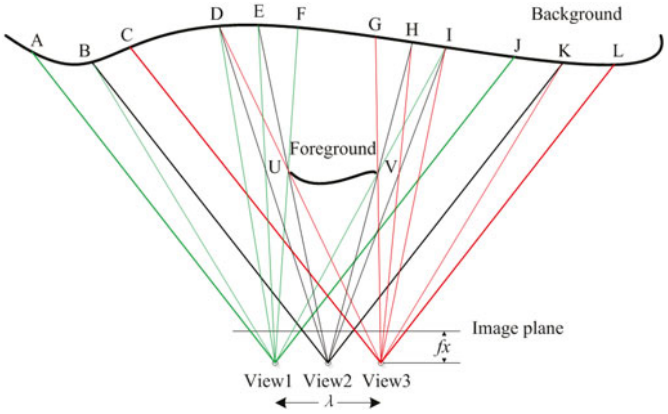

Fig. 2. Illustration of different regions in 3-D warping.

views so that their sizes are usually large (more than two-pixel width), which are the main holes in the virtual view. The dis-occlusion holes in one warped view are generally filled using the information from the other warped view. Boundary holes are the regions that are out of the visible range in an original view but within the visible range of the virtual view. Because the range of each view is limited, when an original view is used as the dominant view to synthesize the virtual view, it produces holes at the picture boundary. Boundary holes can also be filled using the information from the other original view. Cracks are defined as holes with a size no larger than 2-pixel width. They are caused by small depth discontinuities in the original views or rounding errors in the 3-D warping process. Because cracks have high correlation with the neighboring pixels, an interpolation process can be used to fill this type of holes.

Generally speaking, the blending stage should mainly fill the dis-occlusion holes and boundary holes using some regional information in the other original view. However, because the specific regions may not be able to be located accurately, a frame-based 3-D warping is used in both VSRS3.5 and VSRS-1D-fast, meaning that all pixels in the original frames are warped, which causes repeated regional pixels (that are visible in both of the two original views) being warped. If these specific regions can be located, all repeated 3-D warping can be avoided and the blending stage can thus be saved.

III. Analysis of Regional Characteristics

In the discussion above, we notice that certain or even many computations can be saved for some regions. However, the key problem is how to identify such regions. To this end, we present a detailed analysis of the characteristics of different regions in this section.

A vertical view of a scene and the cameras describing a foreground object in front of a background plane are shown in Fig. 2. View1 and View3 stand for two original views, and View2 is the virtual view that will be synthesized from View1 and View3. Region *UV* represents a foreground, whereas points *A*, *B*, ..., *K*, and *L* are all in the background region. Specifically,

1) *A* and *J* are the two boundary points covered by View1;
2) *C* and *L* are the two boundary points covered by View3;
3) *B* and *K* are the two boundary points that will be covered by View2;
4) *D* and *F* are the corresponding points of *U* in the background for View3 and View1, respectively;
5) *E* is the corresponding point of *U* in the background for View2;
6) *G* and *I* are the corresponding points of *V* in the background for View3 and View1, respectively;
7) *H* is the corresponding point of *V* in the background for View2.

For simplicity, we choose View2 at the midpoint between View1 and View3. In addition, the optic axes of the views are parallel to each other. Among the views, there is only *X*-axis directional disparity, which is known as the 1-D parallel arrangement, the same as that used in VSRS-1D-fast. The length of the baseline distance between View1 and View3 is λ, and f_x is the focal length of the camera. According to the regions in the virtual view, regions in the image planes of View1 and View3 are classified into the following types.

1) *Boundary Non-Effective Region (BO-NER):* When synthesizing View2 by View1, we may find that Region *AB* is out of the visible range of the virtual view. Thus, this region has no effect on the virtual view. We refer to it as a BO-NER. Similarly, Region *KL* in View3 is also a BO-NER. In the frame-based synthesis algorithms such as VSRS3.5 and VSRS-1D-fast, the pixels of a BO-NER are still warped to the virtual view, which requires computation but brings no benefit to the synthesized view.
2) *Boundary Single Effective Region (BO-SER):* Region *BC* can only be seen from one view (i.e., View1) when we synthesize View2. We refer to it as a BO-SER. In other words, there will be holes (boundary holes) if we use View3 to synthesize View2. Similarly, Region *JK* is also a BO-SER for View3. Generally, blending uses the BO-SER of one original view to fill the holes generated by the other original view. Therefore, the information in this type of region is the key to filling the boundary holes.
3) *Double Effective Region (DER):* Regions *CD*, *UV*, and *IJ* can be observed by both original views, and their information is repeatedly used during rendering. Regions of this type are referred to as DERs. Although the depth in most of these regions is continuous, there could be depth discontinuities, which would cause small cracks in the warped view. Given that the cracks are small and are strongly related to their neighboring pixels, an interpolation using the neighboring pixels would be used to fill these cracks rather than using the other original view's pixels.
4) *Background Single Effective Region (BA-SER):* Region *DE* belongs to the background and is a single effective region in View1, but it appears at the boundary between the foreground and the background. Such a region is referred to as a BA-SER. Similarly, Region *HI* is a BA-SER for View3. The characteristic of such regions is similar to that of the BO-SER; i.e., the information in these regions is the key to filling the dis-occlusion holes, which are the main holes in the virtual view.

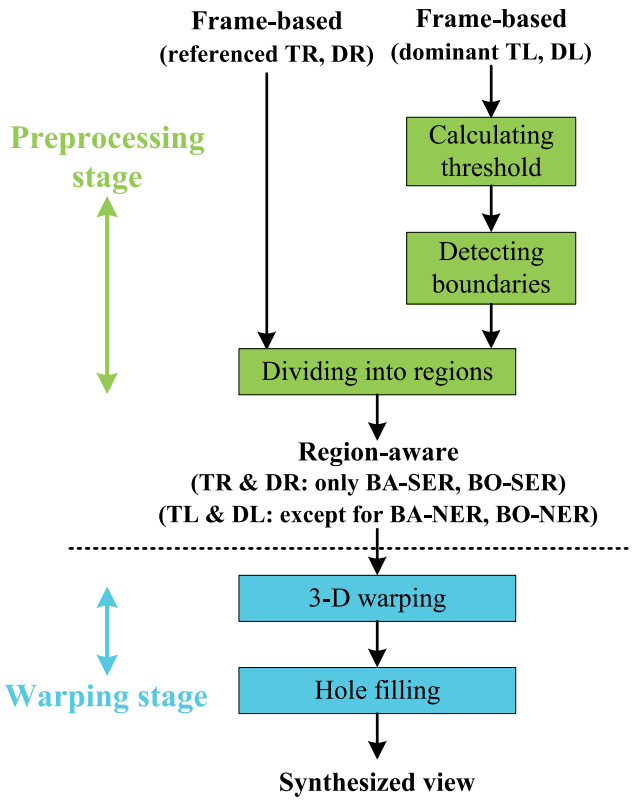

Fig. 3. Flowchart of our proposed region-aware 3-D warping.

5) *Background Non-Effective Region (BA-NER):* Region *EF* is the area in which the mapping competition occurs, as background *EF* and a part of foreground *UV* might be mapped to the same area in the virtual view during the 3-D warping process. Occlusion will occur in this case. Currently, there are two measures to address such a situation. One is to use the z-buffer as in VSRS3.5. The other is to use the special 3-D warping method such as that in VSRS-1D-fast: To the left original view, the 3-D warping direction is from left to right on each line so that the foreground *UV* will cover *EF*, which keeps the logical occlusion order; to the right original view, the 3-D warping order will reverse. We refer to this occluded region as a BA-NER. Pixels of such areas are still mapped in VSRS3.5 and VSRS-1D-fast. According to this definition, *GH* is also a BA-NER for View3.

IV. Region-Aware 3-D Warping

Based on the analysis presented in Section III, we can apply different processing to different regions to achieve computation savings. In this section, we present our region-aware 3-D warping with the detailed block diagram shown in Fig. 3.

To start, we select one of the original views arbitrarily, e.g., the left view (View1), as the dominant view and the other as the reference view. Although the information is available from both views for many regions, because the texture may have slightly different values in different views, it is advantageous to use the information from the dominant view as much as possible to make the texture in the virtual view more consistent. Because the relationships among these views are constructed by the spatial geometry information, we start from the depth map of the dominant view to do the region division.

In the first step of our proposed approach, we calculate the threshold for the depth discontinuity to determine the boundaries of interest for performing the region division. In the second step, we detect the boundaries and perform the region partitioning. In the third step, we perform the region-aware 3-D warping. All regions except BO-NER and BA-NER in the dominant view will be warped to the virtual view. In addition, the effective regions including BO-SER and BA-SER in the reference view will be warped to the virtual view. Finally, an interpolation is carried out for hole filling dis-occlusion.

A. Calculating the Threshold $\Delta Z_{\max}$ to Detect Disocclusion Holes (Occlusion Regions)

As previously mentioned, our purpose is to find the larger BO-NERs (occlusion regions) in the dominant view to discard and locate the BO-SERs in the reference view to fill holes. The smaller occlusions (whose sizes are the same as those of cracks) and cracks are sufficiently small so that they can be easily addressed.

We calculate a watershed depth discontinuity threshold $\Delta Z_{\max}$ to distinguish the dis-occlusion holes (or larger occlusion regions) from the cracks (or smaller occlusion regions). If the depth difference between two neighboring pixels is larger than $\Delta Z_{\max}$, there will be a dis-occlusion hole (or larger occlusion). In this paper, we set the threshold $\Delta Z_{\max}$ as follows:

$$\Delta Z_{\max} = \frac{510}{\lambda \cdot f_x \cdot \left(\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}}\right)} \tag{1}$$

where Z_{near} and Z_{far} are the depth range of the physical scene. λ and f_x are the baseline and focal length, respectively. The derivation for determining this threshold is shown in Appendix for conciseness.

B. Detecting the Boundaries of Interest in the Depth Image

We calculate the depth difference ΔZ between two neighboring pixels in the depth map as discussed in [22]. Based on this depth difference, we can detect the boundaries of interest using $\Delta Z_{\max}$ as the threshold, where $d(x, y)$ stands for the depth value of the pixel at (x, y)

$$F(x,y) = \begin{cases} 1, & \Delta Z_L = d(x,y) - d(x-1,y) \geq \Delta Z_{\max} \\ -1, & \Delta Z_R = d(x+1,y) - d(x,y) \leq -\Delta Z_{\max} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In (2), $F(x, y)$ is the boundary function. If $F(x, y)$ equals 1, it indicates that there is a left boundary between $(x-1, y)$ and (x, y), where (x, y) belongs to the foreground and $(x-1, y)$ belongs to the background. In this case, occlusion will occur. If $F(x, y)$ equals -1, this indicates that there is a right boundary. In this case, dis-occlusion will occur. Otherwise, there is no boundary appearing between these two neighboring pixels.

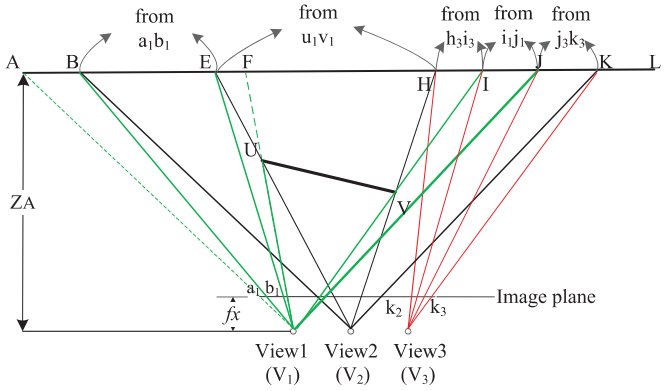

Fig. 4. Region-aware rendering.

C. Region-Aware 3-D Warping for a Flat Background

In Section III, the characteristics of different region are analyzed. Here, we will calculate the regions sizes and perform the 3-D warping for different regions selectively. For clarity of the presentation, we first consider the simple example shown in Fig. 4, where a foreground object is in front of a flat background plane in parallel to the image plane. A more general case will be discussed in Section IV-D.

The points in the 3-D real world are denoted as *A*, *B*, ... ; the corresponding points in the depth image of View1 are denoted as a_1, b_1, ... , the same as a_2, b_2, ... in View2. Note that because point *A* is out of the visible range of View2, "a_2" does not actually exist.

As shown in Fig. 4, to generate View2, we need to warp the regions from the corresponding regions in the image planes of View1 and View3. We generate View2 starting from the left using the information in View1. Because Region *AB* is out of View2's visible range, we could skip Region "a_1b_1" warping. Let $\Phi_{a_1b_1}$ denote the size of Region "a_1b_1"; similarly, Φ_{AB} denotes the size of Region *AB*, etc. For Region *AB*, because the flat background plane is in parallel to the image plane, we have

$$\frac{\Phi_{a_1b_1}}{\Phi_{\mathrm{AB}}} = \frac{f_x}{Z_B}. \tag{3}$$

View2 is at the midpoint between View1 and View3, and Φ_{AB} is λ/2. Z_B is the physical depth of point *B*. Considering that the background is a flat plane and Z_B is the same as Z_A, we can calculate $\Phi_{a_1b_1}$ as

$$\Phi_{a_1b_1} = \frac{\Phi_{\mathrm{AB}} \cdot f_x}{Z_B} = \frac{\lambda \cdot f_x}{2Z_B} = \frac{\lambda \cdot f_x}{2Z_A} = \beta_A. \tag{4}$$

Denote β_X as the disparity at point *X* in View1 with respect to View2. Then, β_A is the disparity at point *A* in View1 with respect to View2, which can be calculated from the depth value d_{a_1} at "a_1" by

$$\beta_A = \frac{\lambda}{2} \cdot f_x \cdot \left(\frac{d_{a_1}}{255} \cdot \left(\frac{1}{Z_{\mathrm{near}}} - \frac{1}{Z_{\mathrm{far}}} \right) + \frac{1}{Z_{\mathrm{far}}} \right). \tag{5}$$

Hence, if we obtain β_A, we can calculate $\Phi_{a_1b_1}$. To obtain β_A in a time-saving manner, a look-up table-based technique inherited from VSRS-1D-fast is used here. Given a depth value d_{a_1} of point "a_1" in the depth image, the corresponding disparity β_A can be determined by looking up the table stored in advance.

Next, we need to calculate $\Phi_{b_1e_1}$ in View1 so that we can render Region "b_2e_2" in View2 from Region "b_1e_1" in View1. Because *U* appears as a left boundary point in View1, we can locate "u_1" in View1 easily. Because the triangle *UEF* is similar to the triangle UV_2V_1, we obtain

$$\Phi_{EF} = \frac{\lambda}{2} \cdot \frac{Z_F - Z_U}{Z_U}. \tag{6}$$

Meanwhile, we know that

$$\frac{\Phi_{e_1f_1}}{\Phi_{EF}} = \frac{f_x}{Z_F}. \tag{7}$$

Combining (6) and (7), we can generate

$$\Phi_{e_1f_1} = \beta_U - \beta_F. \tag{8}$$

From the point "u_1" (which is the point in View1 corresponding to the points *U* and *F*) and $\Phi_{e_1f_1}$, we can locate the point "e_1" in View1 and then use Region "b_1e_1" in View1 to render Region "b_2e_2" in View2. To render Region "e_2h_2" in View2 from Region "u_1v_1" in View1, because *V* is a right boundary, we can locate Region "u_1v_1" in View1 and warp Region "u_1v_1" into Region "e_2h_2" easily.

Because Region *HI* is occluded from View1, we need to render Region "h_2i_2" in View2 from "h_3i_3" in View3. To do this, we first need to find Φ_{HI} and locate the point "h_3" in View 3. Because the triangle *VHI* is similar to the triangle VV_2V_1 and $\Phi_{V_1V_2} = \lambda/2$

$$\Phi_{HI} = \frac{\lambda}{2} \cdot \frac{Z_I - Z_V}{Z_V}. \tag{9}$$

Thus

$$\Phi_{h_3i_3} = \beta_V - \beta_I. \tag{10}$$

Now, we can warp "i_1" to "i_3" and locate the point "h_3" by subtracting $\Phi_{h_3i_3}$. After that, we can warp Region "h_3i_3" in View3 to "h_2i_2" in View2.

Next, we need to render Region "i_2j_2" from the information of "i_1j_1" because View1 is the dominant view (although the information is also available from View3 in this case). It should be noted that "i_1j_1" is next to"u_1v_1" in View1 and the point "j_1" is the end of the image plane of View1. Thus, Region "u_1j_1" can be processed as a whole in View1, and we need not to confirm the size of Region "i_1j_1."

Finally, the information of Region "j_2k_2" from Region "j_3k_3" in View3 is located, because the information is not available from View1. Before that, we need to find the point "j_3" in View3. To locate the point "j_3," we just need to warp the point "j_1" to "j_3." Because $Z_J = Z_K$, we obtain

$$\Phi_{j_3k_3} = \Phi_{j_3r_3} - \Phi_{k_3r_3} = \frac{\lambda \cdot f_x}{Z_J} - \frac{\lambda \cdot f_x}{2Z_K} = \frac{\lambda \cdot f_x}{2Z_J} = \beta_J. \tag{11}$$

In practical situations, holes will be created when parts of the image are occluded from both View1 and View3 but are visible from View2. In this case, regardless of the algorithms (VSRS-1D-fast, VSRS3.5, or ours) that are used to perform the

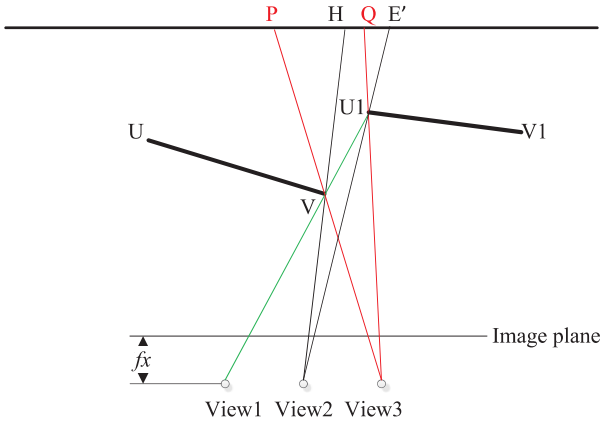

Fig. 5. Holes in 3-D warping.

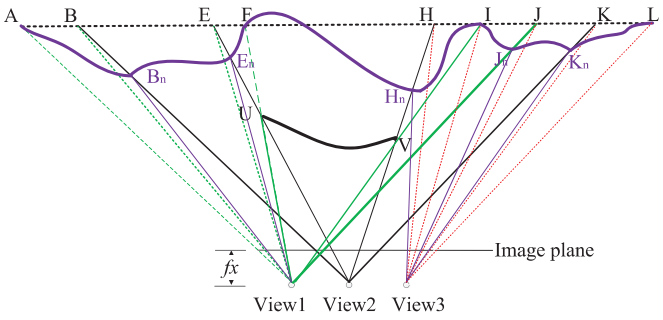

Fig. 6. Background is non-flat, as the bold purple line shows.

3-D warping, holes will be created. An example of a situation involving holes is shown in Fig. 5: *HE′* is occluded in View1 but visible from View2, whereas *QE′* is occluded in View3 but visible from View2. Thus, *QE′* is visible from View2 but occluded in both View1 and View3, becoming holes. This situation occurs when the right boundary point of a foreground object is close to the left boundary of another foreground object and *Q* falls between *H* and *E′*. Because the point *H* in Fig. 5 plays a similar role as the point *H* in Fig. 4 and the point *E′* in Fig. 5 is similar to the point *E* in Fig. 4, these points can be identified in the process as discussed above. Thus, this situation can be detected. When this situation occurs, after the right boundary point, the process discussed above can still be used, but the region between *QE′* will become holes and should be inpainted from the background. It should be noted that these holes are not specific to our proposed method. It should also be noted that one foreground object could partially overlap with another foreground object. In this case, a left boundary could be followed by another left boundary. The proposed approach can handle this situation correctly because the process is boundary based. In this case, the foreground object behind will becomes the background of the other foreground object.

D. Region-Aware 3-D Warping for General Background

The above discussions are based on the assumption that the background is flat. However, in practice, backgrounds are usually non-flat and have depth changes as shown in Fig. 6. To address this case, we further propose a more general method to make the region calculation more reasonable. Note that the rendering process is similar to that of the flat background explained in Section IV-C. Here, to make sure that the region size is as accurate as possible, an iterative algorithm is used.

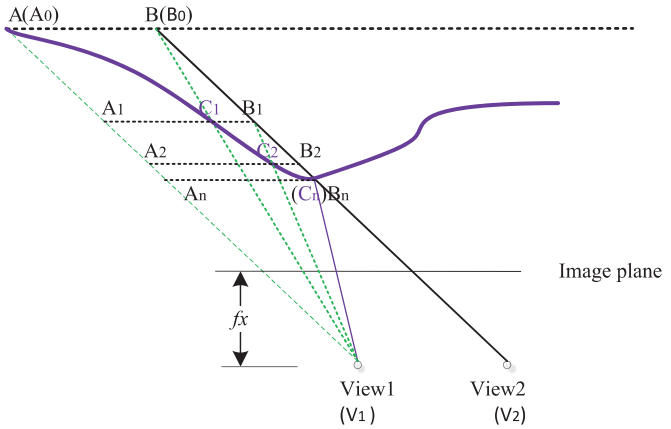

Fig. 7. This figure is a part of Fig. 6 for convenient representation. Connect *B* and View1 with a straight line crossing the background at C_1. Points A_1, B_1, and C_1 are on the same line. *AB* and A_1B_1 are parallel lines.

When the background is not flat, the size of each region will change; e.g., a BO-NER Region *AB* will evolve from A_0B_0 to A_nB_n, where *n* stands for the nth iteration. In View1's image plane, the corresponding region size $\Phi_{a_1 b_1}$ changes from $\Phi_{a_{1,0}b_{1,0}}$ to $\Phi_{a_{1,n}b_{1,n}}$. Our purpose is to obtain $\Phi_{a_{1,n}b_{1,n}}$. To this end, we should know B_n's depth value and the corresponding disparity. However, B_n's depth value is hardly known. To address this, we propose an algorithm to approach point B_n from point $B(B_0)$ iteratively. The details are as follows.

Denote A_i as a point with the same position as point *A* in View1 in the ith iteration; B_i has distance $\lambda/2$ from A_i in the horizontal direction as shown in Fig. 7. Points A_i, B_i and C_i have the same physical depth distance and are represented by $a_{1,i}$, $b_{1,i}$, $c_{1,i}$, respectively, in View1.

1) The Initial Iteration Starts From $i = 0$: Because *B*(B_0)'s depth value can be confirmed, (point *B*'s depth value is equal to that of point *A*), and the region size $\Phi_{a_{1,0}b_{1,0}}$ can be calculated by (4), $\Phi_{a_{1,0}b_{1,0}} = \beta_B = \beta_A$. Locate point C_1 in View1's depth map using $\Phi_{a_{1,0}b_{1,0}}$ (the coordinate of point C_1 equals $\Phi_{a_{1,0}b_{1,0}}$. Then, point $\Phi_{a_{1,0}b_{1,0}}$'s depth value can be obtained by checking View1's depth image. In this way, we can obtain point B_1's depth value, which is equal to that of point C_1.

2) In the Next Iteration, Where $i = 1$: Region A_1B_1's size in View1 can be confirmed using B_1's depth value, namely, $\Phi_{a_{1,1}b_{1,1}} = \beta_{B_1}$. Locate point C_2 in View1's depth map using $\Phi_{a_{1,1}b_{1,1}}$. Then, point C_2's depth value can be obtained by checking View1's depth image, and consequently, we can obtain point B_2's depth value, which is equal to that of point C_2.

3) In the ith Iteration, Where $i \geq 2$: Region A_iB_i's size in View1 can be confirmed using B_i's depth value, namely, $\Phi_{a_{1,i}b_{1,i}} = \beta_{B_i}$ Locate point C_i in View1's depth map using $\Phi_{a_{1,i}b_{1,i}}$, Then, point B_{i+1}'s depth value can be obtained, which equals to that of point B_{i+1}.

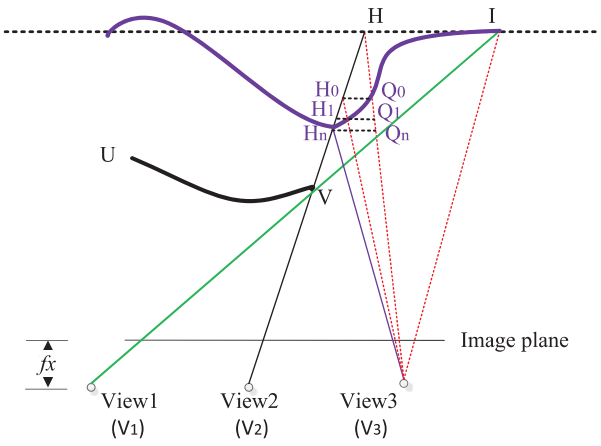

Fig. 8. This figure is a part of Fig. 6 for convenient representation. Connect point H and View3 with a straight line crossing the background at point Q_0, where H_0Q_0 and the image plane are parallel.

The iteration will stop when it meets either of the following two conditions: 1) the absolute value of the depth difference between point B_i and B_{i+1} is smaller than $\Delta Z_{\max}/2$, which suggests that this depth difference will lead to a region change of no more than one sampling precision and consequently C_i converges to B_i, and 2) the maximum number of iteration is no more than 5.

After the iteration stops, we can obtain the accurate size of Region BO-NER

$$\Phi_{a_{1,n}b_{1,n}} = \beta_{B_n}. \tag{12}$$

We can use the same procedure to find the other regions as follows:

For BA-NER

$$\Phi_{e_{1,n}f_{1,n}} = \beta_U - \beta_{E_n}. \tag{13}$$

For BO-SER in View3

$$\Phi_{k_{3,n}l_{3,n}} = \frac{\Phi_{KL}\cdot f_x}{Z_{K_n}} = \frac{\lambda\cdot f_x}{2Z_{K_n}} = \beta_{K_n} \tag{14}$$

$$\Phi_{j_{3,n}l_{3,n}} = \frac{\Phi_{JL}\cdot f_x}{Z_{J_n}} = \frac{\lambda\cdot f_x}{Z_{J_n}} = 2\beta_{J_n} \tag{15}$$

and

$$\Phi_{j_{3,n}k_{3,n}} = \Phi_{j_{3,n}l_{3,n}} - \Phi_{k_{3,n}l_{3,n}} = 2\beta_{J_n} - \beta_{K_n}. \tag{16}$$

Finally, BA-SER will change from Region *HI* to Region H_nI in View3, so the depth value of point H_n is needed to calculate the size of region H_nI. Because H_nI consists of Region H_nQ_n and Region HI as shown in Fig. 8, the size of Region HI in View3, i.e., $\Phi_{h_3i_3}$ could be calculated by (10). Then, $\Phi_{h_{3,n}q_{3,n}}$ could be calculated easily because we can obtain H_n's depth value and the corresponding disparity by the iteration. Specifically, denote Q_i as a point with the same position as point H in View3 in the ith iteration; then, H_i is the point with the same physical depth distance to point Q_i. Because $\Phi_{h_3i_3}$ is known, point Q_0's physical depth distance could be determined by $Z_{Q_0} = Z_{H_0}$. So, we can obtain

$$\frac{Z_H - Z_{H_0}}{Z_I} = \frac{H_0Q_0}{V_2V_3} = \frac{H_0Q_0}{\lambda/2} \tag{17}$$

$$\frac{H_0Q_0}{\Phi_{h_{3,0}q_{3,0}}} = \frac{Z_{H_0}}{f_x}. \tag{18}$$

Combining (17) and (18), we obtain

$$\Phi_{h_{3,0}q_{3,0}} = \frac{\lambda\cdot f_x}{2}\cdot\left(\frac{1}{Z_{H_0}} - \frac{1}{Z_I}\right) = \beta_{H_0} - \beta_I. \tag{19}$$

In the iteration, the initial value of $\Phi_{h_{3,i}q_{3,i}}$ is set to $\Phi_{h_{3,0}q_{3,0}}$, and the iteration step is almost the same as the aforementioned method except that point H_i's depth value and the corresponding disparity act as the iteration variable

$$\Phi_{h_{3,n}q_{3,n}} = \beta_{H_n} - \beta_I. \tag{20}$$

Finally, Region H_nI's size can be calculated as

$$\Phi_{h_{3,n}i_3} = \Phi_{h_{3,n}q_{3,n}} + \Phi_{h_3i_3} = \beta_{H_n} + \beta_V - 2\beta_I. \tag{21}$$

It is worth noting that, during the iteration stage, the computation focuses on the disparity difference and the separable look-up table-based technique [11] can be reused. In other words, both VSRS-1D-fast and our method share the same table in terms of the look-up table-based technique.

V. Experimental Results

In this section, we present some detailed simulation results for our proposed region-aware 3-D warping algorithm. Our methods performance is compared against VSRS3.5 and VSRS-1D-fast, which are considered as the conventional algorithm and the state-of-the-art algorithm for view synthesis. The performance metrics used here include 1) the time complexity for view synthesis, 2) the objective PSNR of the synthesized view, 3) the mean structural similarity index metric (SSIM) [23] for the synthesized views, and 4) visual results and the statistics on the ratio of the partitioned regions. In addition, some experimental results are also exhibited here to evaluate the depth noise effects and the changing baseline effects on these methods. The multi-view sequences used for this evaluation are the first 100 frames of the Nagoya University's *Kendo* (use views 1 and 3 to render view 2) and *Balloons* (use views 1 and 3 to render view 2) [24], GIST's *Café* (use views 2 and 4 to render view 3) [25], HHI's *BookArrival* (use views 8 and 10 to render view 9) [26], and Philips' *Mobile* (using views 3 and 5 to render view 4) [27]. The size of *Kendo*, *Balloons*, and *BookArrival* is 1024 × 768 pixels, while *Café* and *Mobile*'s sizes are 1920 × 1080 pixels and 720 × 540 pixels, respectively. All these sequences are uncompressed, which are used in the whole Section V, unless other specified. For VSRS3.5, we choose the 1-D parallel mode. The precision is set to three different types: integer/full-pel, half-pel, and quarter-pel. In addition, all 1-D parallel options are set to the default values. For VSRS-1D-fast, we also choose three different precisions, and the blend model is set to "holes from right." The other options are all set to the default values. Our proposed method,VSRS3.5, and VSRS-1D-fast are

TABLE I
RUN TIME COMPARISON AMONG DIFFERENT ALGORITHMS

		Preprocessing stage time (s)			Warping stage time (s)			Blending stage time (s)			Total time (s)		
Sequence	Precision	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours
Kendo	Full	0	0	0.003	0.101	0.036	0.010	0.068	0.011	0	0.169	0.046	0.013
	Half	0	0	0.003	0.111	0.034	0.010	0.147	0.010	0	0.258	0.045	0.012
	Quarter	0	0	0.003	0.221	0.035	0.010	0.410	0.011	0	0.631	0.046	0.013
Balloons	Full	0	0	0.003	0.101	0.034	0.010	0.067	0.011	0	0.168	0.045	0.013
	Half	0	0	0.002	0.110	0.034	0.009	0.145	0.011	0	0.255	0.045	0.012
	Quarter	0	0	0.003	0.216	0.035	0.010	0.367	0.011	0	0.583	0.046	0.013
Café	Full	0	0	0.007	0.257	0.091	0.025	0.175	0.029	0	0.432	0.120	0.031
	Half	0	0	0.007	0.292	0.092	0.024	0.409	0.029	0	0.702	0.120	0.032
	Quarter	0	0	0.006	0.593	0.095	0.027	1.086	0.029	0	1.679	0.124	0.033
BookArrival	Full	0	0	0.003	0.102	0.036	0.010	0.065	0.011	0	0.167	0.047	0.013
	Half	0	0	0.003	0.113	0.035	0.010	0.157	0.011	0	0.270	0.046	0.012
	Quarter	0	0	0.003	0.222	0.036	0.010	0.417	0.012	0	0.639	0.047	0.013
Mobile	Full	0	0	0.001	0.049	0.017	0.004	0.028	0.005	0	0.077	0.022	0.006
	Half	0	0	0.001	0.055	0.016	0.004	0.078	0.005	0	0.133	0.021	0.005
	Quarter	0	0	0.001	0.110	0.017	0.005	0.204	0.005	0	0.314	0.022	0.006

TABLE II
COMPARISON OF PSNR AMONG DIFFERENT ALGORITHMS

Sequence	Precision	VSRS3.5 (dB)	1-D-fast (dB)	Ours (dB)
Kendo	Full	39.02	38.75	38.80
	Half	39.23	39.41	39.43
	Quarter	39.32	39.42	39.44
Balloons	Full	38.20	38.16	38.23
	Half	38.68	38.75	38.82
	Quarter	38.82	38.75	38.83
Café	Full	36.41	35.72	35.68
	Half	36.19	35.77	35.71
	Quarter	35.80	35.78	35.70
BookArrival	Full	37.61	37.71	37.63
	Half	38.44	38.52	38.45
	Quarter	38.26	38.61	38.54
Mobile	Full	43.02	43.49	44.44
	Half	43.49	43.56	44.27
	Quarter	43.90	43.79	44.29
Average		39.09	39.08	39.22

TABLE III
SSIM COMPARISON AMONG DIFFERENT ALGORITHMS

Sequence	Precision	VSRS3.5	1D-fast	Ours
Kendo	Full	0.986	0.982	0.983
	Half	0.986	0.984	0.984
	Quarter	0.986	0.984	0.984
Balloons	Full	0.982	0.981	0.981
	Half	0.983	0.982	0.982
	Quarter	0.984	0.982	0.982
Café	Full	0.976	0.972	0.971
	Half	0.976	0.972	0.972
	Quarter	0.975	0.972	0.972
BookArrival	Full	0.974	0.973	0.973
	Half	0.976	0.974	0.974
	Quarter	0.976	0.975	0.974
Mobile	Full	0.995	0.995	0.996
	Half	0.990	0.996	0.996
	Quarter	0.991	0.996	0.996
Average		0.982	0.981	0.981

run on a workstation with an Intel(R) Core(TM) i7-4790 CPU at 3.60 GHz and 8.00 GB of DDR3 RAM.

A. Time Complexity

We compare the run times among these three schemes. As shown in Table I, we can see that our approach is approximately 13.3 times faster than VSRS3.5 and 3.7 times faster than VSRS-1D-fast in full-pel precision, approximately 22.2 times faster than VSRS3.5 and 3.8 times faster than VSRS-1D-fast in half-pel precision, and approximately 49.3 times faster than VSRS3.5 and 3.7 times faster than VSRS-1D-fast in quarter-pel precision.

We also list in Table I the step-wise execution time comparison of the three methods. First, the preprocessing stage only exists in our method, and its goal is to calculate the size of different regions. Second, in the warping stage of VSRS3.5, due to the use of the frame-based forward warping, the process is quite time consuming. In addition, the boundary-aware splatting and the z-buffer rules adopted in this stage also contribute to the complexity. In the warping stage of VSRS-1D-fast, although a separable look-up table-based 3-D warping technique speeds up the forward warping efficiently, both the determination of the three cases (warping, interpolation and hole filling) and the corresponding processing add additional computation. For our method, because only the necessary regions are warped and the mapping competition can be avoided, it is the most computationally efficient. Finally, for the blending stage, both VSRS3.5 and VSRS-1D-fast consume run time. For our method, because the blending has been merged into the preprocessing and warping stages, we do not need to perform the blending.

B. Objective Quality Evaluation

To evaluate the quality of the synthesized views objectively, we compare their PSNR. The results are given in Table II, where 2(1, 3) represents that View2 is the virtual view that is synthesized using View1 and View3, etc. The PSNR results correspond to the average PSNR over the first 100 frames. It

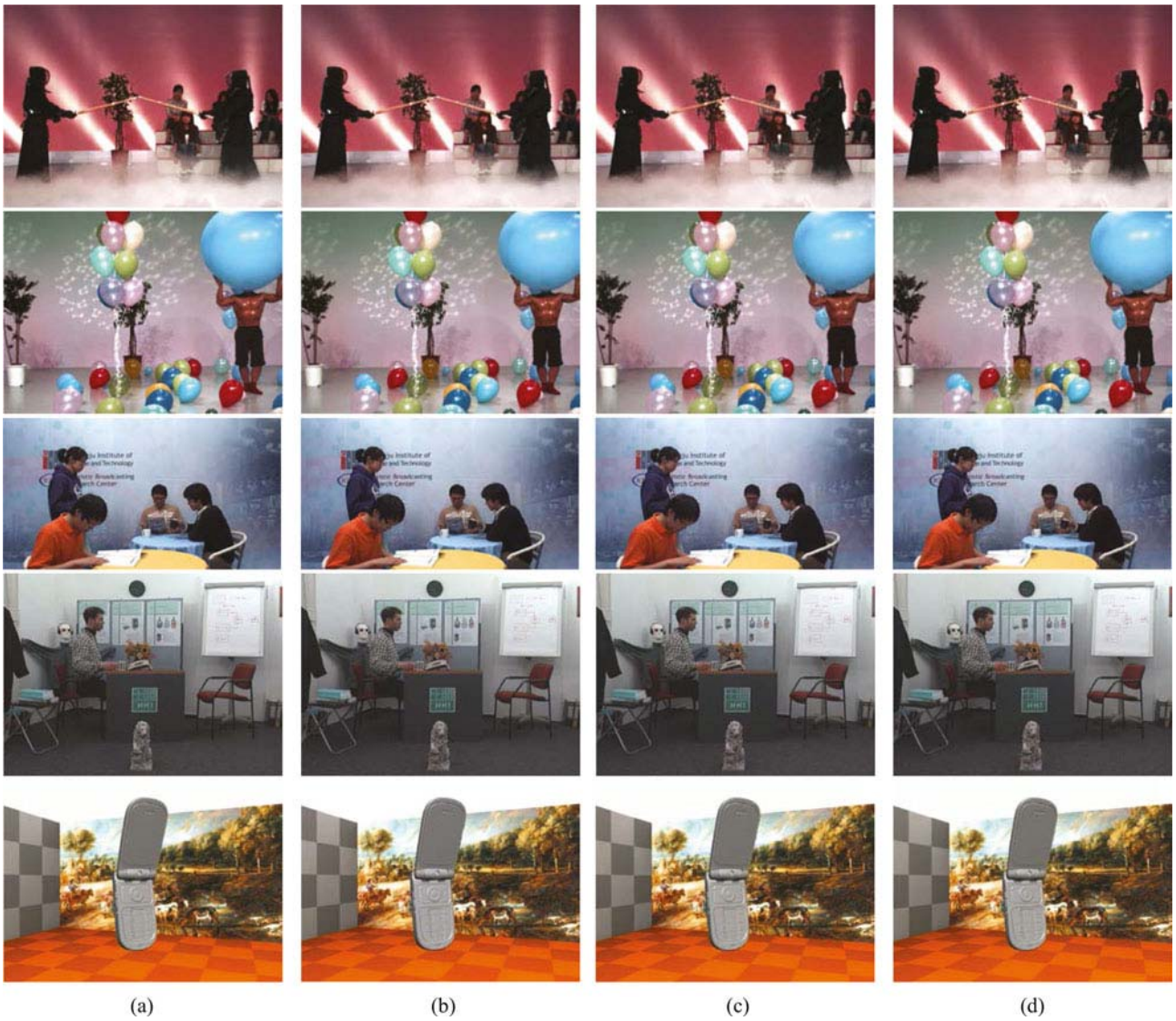

Fig. 9. This figure is to show the subjective evaluation of the synthesized view. (a) Original target view. (b)–(d) Generated by VSRS3.5, VSRS-1D-fast, and our method, respectively.

can be seen from these results that our scheme has achieved an objective quality that is very comparable to that of VSRS3.5 and VSRS-1D-fast. It should be noted that VSRS3.5 uses boundary splatting and depth-based inpainting, whereas VSRS-1D-fast uses a reliability-based weighting scheme for blending. These are more complex but can improve the quality of the virtual view to a certain extent. However, for our method, the cracks are filled using the neighboring pixels information instead of the pixel from the other reference views, which can also bring some benefits in some sequences.

TABLE IV
PARAMETERS OF CAMERAS SETUP AND ASSOCIATED DEPTH RANGES OF TEST SEQUENCES

	Parameters and depth ranges			
Sequence	f_x	λ	Z_{near}	Z_{far}
Kendo	2241.25607	10.0	448.251214	11206.280350
Balloons	2241.25607	10.0	448.251214	11206.280350
Café	1796.47620	130.0	1297.455033	3892.365100
BookArrival	1399.46666	1.159238	23.175928	54.077165
Mobile	1000	1.0	22.959650	53.572517

C. SSIM and Subjective Evaluation

During the 3-D warping process, pixel position changes are inevitable, which could affect the PSNR comparisons. However, the pixel position changes do not always lead to disturbing artifacts. Thus, we also compute the SSIM for the synthesized views with the first 100 frames. During the SSIM tests, all parameters are set to default values. The results are shown in Table III, from which it can be seen that our algorithm is still very comparable to VSRS3.5 and VSRS-1D-fast. It is important to note that the results over all test sequences are greater than 0.970, which indicates that the synthesized views match structurally the

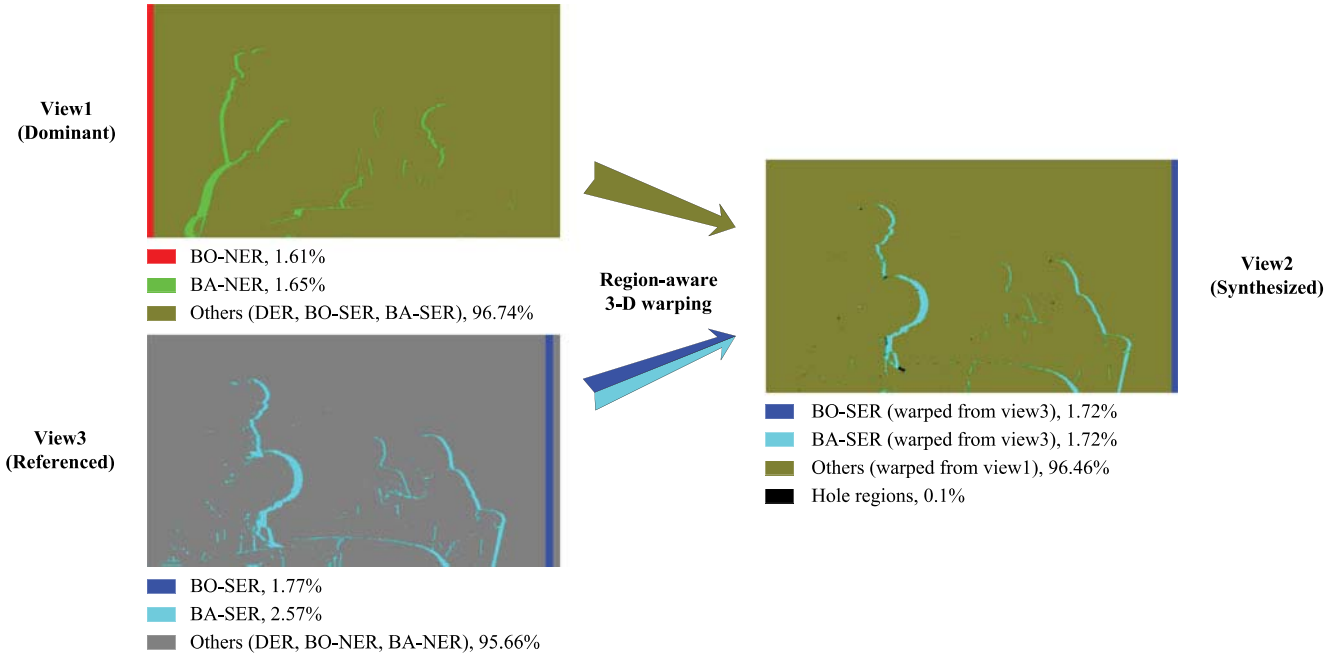

Fig. 10. Statistics on the ratio of the regions during the warping stage in our method for the first frame of *Café*.

original views very well and thus will not pose any problem during subjective viewing tests. To support this statement, we present some synthesized views (using VSRS3.5, VSRS-1D-fast, and our proposed algorithm) in Fig. 9. The original target view is in part (a), while parts (b), (c), and (d) are the synthesized views using VSRS3.5, VSRS-1D-fast, and our scheme, respectively. All these pictures are the first frames of the sequences with quarter-pel precision. We can see that our approach offers the same subjective/visual quality as the others.

D. Statistics on the Ratios of the Partitioned Regions

Because the performance of our proposed method depends on the ratio of the various regions obtained after the division, an experiment and the corresponding analysis are shown in the following. Besides, the parameters of camera setup and the associated depth ranges of test sequences are also given out in this part as shown in Table IV.[1]

As shown in Fig. 10, the ratio of the regions during the warping stage in our method is given based on the 1st frame of *Café*, where both View1 and View3 are divided into several regions. To be more convincing, we present the average region-ratio results over 100 frames in these five sequences in Fig. 11. Note that the bottom row of this figure presents the average ratios over five sequences, e.g., the average ratios of BO-NER, BA-NER, and others (DER, BO-SER, BA-SER) in the dominant View1 are 1.08%, 0.73%, and 98.19%, respectively; whereas the average ratios of BO-SER, BA-SER, and others (DER, BO-NER, BA-NER) in the reference View3 are 1.89%, 1.86%, and 96.26%, respectively.

[1] f_x is the focal length of camera, λ is the length of baseline, Z_{near}, and Z_{far} are the lower and upper limits of physical depth range, respectively.

		Region (ratio)			
Sequence	View1	BO-NER	BA-NER	Others	
	View2	Others	BO-SER	BA-SER	HOLE
	View3	BO-SER	BA-SER	Others	
Kendo	View1	0.0094	0.0021	0.9885	
	View2	0.9804	0.0123	0.0049	0.0024
	View3	0.0166	0.0157	0.9677	
Balloons	View1	0.0075	0.0065	0.986	
	View2	0.9722	0.015	0.0094	0.0034
	View3	0.0198	0.0208	0.9595	
Café	View1	0.0161	0.0187	0.9652	
	View2	0.9611	0.0167	0.0179	0.0043
	View3	0.0177	0.0292	0.9531	
BookArrival	View1	0.0075	0.0065	0.986	
	View2	0.9722	0.015	0.0094	0.0034
	View3	0.0198	0.0208	0.9595	
Mobile	View1	0.0135	0.0025	0.984	
	View2	0.9773	0.0167	0.0049	0.001
	View3	0.0204	0.0064	0.9733	
Average	**View1**	**0.0108**	**0.0073**	**0.9819**	
	View2	**0.9726**	**0.0151**	**0.0093**	**0.0029**
	View3	**0.0189**	**0.0186**	**0.9626**	

Fig. 11. Illustration of the average region-ratio results over 100 frames.

To prevent the inaccurate boundary calculation caused by the non-convergence, the lengths of BO-NER and BA-NER are actually set 2 pixels smaller than the calculated values in our algorithm. Consequently, a small portion of the non-effective regions will still be warped and then occluded by the foreground, which reduces the ratio of others (DER, BO-SER, BA-SER) from 98.19% in View1 to 97.26% in View2. Conversely, the lengths of BO-SER and BA-SER are set 2 pixels larger than the calculated value, which means that more pixels in View3 will

TABLE V
VIRTUAL VIEW QUALITY (PSNR) COMPARISON AMONG DIFFERENT ALGORITHMS IN TERMS OF DEPTH NOISE EFFECTS

		PSNR (dB): without compression			PSNR (dB): QP = 40			ΔPSNR (dB)		
Sequence	Precision	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours
Kendo	Full	39.02	38.75	38.80	38.82	38.31	38.30	−0.20	−0.44	−0.50
	Half	39.23	39.41	39.43	39.05	38.63	38.62	−0.18	−0.78	−0.81
	Quarter	39.32	39.42	39.44	39.10	38.68	38.66	−0.22	−0.74	−0.78
Balloons	Full	38.20	38.16	38.23	37.94	37.57	37.56	−0.26	−0.59	−0.67
	Half	38.68	38.75	38.82	38.43	37.85	37.83	−0.25	−0.90	−0.99
	Quarter	38.82	38.75	38.83	38.51	37.91	37.89	−0.31	−0.84	−0.94
Café	Full	36.41	35.72	35.68	36.19	35.59	35.51	−0.22	−0.13	−0.17
	Half	36.19	35.77	35.71	36.03	35.62	35.53	−0.16	−0.15	−0.18
	Quarter	35.80	35.78	35.70	35.70	35.65	35.51	−0.10	−0.13	−0.19
Average		37.96	37.83	37.85	37.75	37.31	37.27	−0.21	−0.52	−0.58

TABLE VI
EXECUTION EFFICIENCY (TIME) COMPARISON AMONG DIFFERENT ALGORITHMS IN TERMS OF DEPTH NOISE EFFECTS

		Time (s): without compression			Time (s): QP = 40			ΔTime (s)		
Sequence	Precision	VSRS3.5	1D-fast	Ours	VSRS3.5	1-D-fast	Ours	VSRS3.5	1-D-fast	Ours
Kendo	Full	0.169	0.046	0.013	0.171	0.045	0.013	0.002	−0.001	0
	Half	0.258	0.045	0.012	0.266	0.044	0.012	0.008	−0.001	0
	Quarter	0.631	0.046	0.013	0.643	0.045	0.013	0.012	−0.001	0
Balloons	Full	0.168	0.045	0.013	0.169	0.046	0.013	0.001	0.001	0
	Half	0.255	0.045	0.012	0.264	0.045	0.012	0.009	0	0
	Quarter	0.583	0.046	0.013	0.614	0.046	0.013	0.031	0	0
Café	Full	0.432	0.120	0.031	0.431	0.122	0.032	−0.001	0.002	0.001
	Half	0.702	0.120	0.032	0.700	0.119	0.032	−0.002	−0.001	0
	Quarter	1.679	0.124	0.033	1.635	0.123	0.033	−0.044	−0.001	0
Average		0.542	0.071	0.019	0.544	0.071	0.019	0.002	0	0

be warped to View2 and finally occluded by View1's warped pixels. As a result, the ratios of BO-SER and BA-SER in View3 (1.89% and 1.86%, respectively) reduce to 1.51% and 0.93%, respectively. Note that the pixels from BO-SER and BA-SER in View3 are warped first, and then, the pixels from DER, BO-SER, and BA-SER in View1 will be warped to cover the pixels from View3.

Finally, it is important to note that, on average, only approximately one frame's (0.9819 + 0.0189 + 0.0186 = 1.0194) have been warped to the virtual view in total in our warping stage, while, two frames' data is warped in VSRS3.5 and VSRS-1D-fast. Obviously, this greatly contributes to time saving.

The performance of our method can achieve significant computation saving in the virtual view synthesis process without sacrificing synthesis quality. However, the preprocessing stage of our method is based on the depth images, and it might become sensitive to depth noise, especially to that caused by depth quantization during the transmission stage. Hence the effects of depth noise, caused by depth quantization, is tested in Section V-E.

E. Effects of Depth Noise

In this section, we use 3D-HEVC [28] to compress the original depth sequences so as to simulate the depth noise caused by depth quantization. We set QP equal to 40, and the depth quantization is quite large. Then, the compressed depth sequences and the original texture sequences (without compression) are used to synthesize the virtual view with these three different algorithms. Still, the first 100 frames are tested in this part. For conciseness, only the first three sequences *Kendo*, *Balloons*, and *Café* are tested in this part. The performances of PSNR and run time are listed in Tables V and VI, respectively. It is unsurprising to find that, although the depth quantization is quite large, our method still has advantages in computation saving. It still maintains comparable results in view synthesis quality compared with VSRS-1D-fast. One other thing to note is that VSRS3.5 contains a depth preprocessing stage [29], which can depress the effects of depth noise to some degree and lead to good performance in the case of depth quantization noise.

F. Effects of Changing Baseline

To evaluate the changing baseline effects on the performance of our method, the *Mobile* sequences are tested, as only *Mobile* sequences contain more than six different texture sequences and associated depth sequences, which provide us with three types of baselines. In this test, the first 100 frames of each sequence and quarter-pel precision of synthesis are used. The experimental results are shown in Table VII, and we find that different

TABLE VII
PERFORMANCE (PSNR AND TIME) COMPARISON OF THREE DIFFERENT ALGORITHMS IN TERMS OF CHANGING BASELINE EFFECTS

Views	Metrics	VSRS3.5	1-D-fast	Ours
5.5(3, 8)	PSNR (dB)	42.82	42.60	42.26
	Time (s)	0.303	0.022	0.006
5.5(4, 7)	PSNR (dB)	44.92	45.52	45.12
	Time (s)	0.306	0.022	0.006
5.5(5, 6)	PSNR (dB)	43.20	42.54	42.58
	Time (s)	0.306	0.022	0.006
Average	PSNR (dB)	43.65	43.55	43.32
	Time (s)	0.305	0.022	0.006

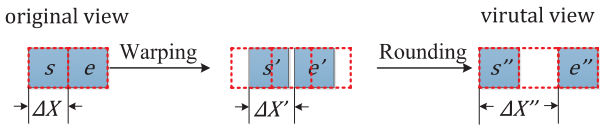

Fig. 12. Illustration of 3-D warping process.

baseline distances between two original views indeed affect the synthesized view's quality, and sometimes quite substantially. Interestingly, as the baseline distance changes, the quality of the synthesized view will be improved or declined in all three methods (VSRS3.5, VSRS-1D-fast, and ours), while the run time remains nearly constant. That is because the frame-based warping is carried out in VSRS3.5 and VSRS-1D-fast, the warping maintains two views' data and thus will not be greatly affected by the changed baseline distance. For our method, although a larger baseline distance will increase the ratios of BO-NER and BA-NER in View1 and BO-SER and BA-SER in View3, the total warping area is unchanged. Thus, our method still maintains the lowest run time.

VI. CONCLUSION AND DISCUSSIONS

A region-aware 3-D warping for DIBR has been proposed in this paper to achieve significant computation saving in the virtual view synthesis process. To this end, we have first performed region partitioning based on the depth map, considering that boundaries in the depth map contain more spatial characteristics than those in the texture image. Then, pixels in different regions are selectively warped. Extensive simulation results have been presented to verify the effectiveness of the proposed algorithm. In particular, our algorithm is demonstrated to be much faster than the well-known VSRS3.5 and VSRS-1D-fast while maintaining similar quality. In addition, in view of the performance regarding the effects of depth noises and baseline changes, our method is still comparable with VSRS-1D-fast. Note that our method currently considers the 1-D parallel model. It is worth noting that the idea of "region-aware 3-D warping" can be extended to the general model. The main challenge is that there will be more directional disparities in the general model, which causes some difficulty in the calculation of the region sizes and consequently the region partitioning. The corresponding study will be the main focus in our future work.

APPENDIX A
DERIVATION OF THRESHOLD $\Delta Z_{\max}$

For simplicity, we only present the derivation for the dis-occlusion case, as the same applies to the occlusion case.

Referring to Fig. 12, we denote two neighboring points in the original view as s and e, while their coordinates are x_s and x_e, respectively. Let ΔX be the interval's size between these two points. Because points s and e neighbor to each other, ΔX equals one sample size. Next, suppose that these two points are warped to s' and e', their new coordinates are x'_s and x'_e, and their interval is denoted as $\Delta X'$. Finally, two warped points are rounded to points s'' and e'' with the coordinates x''_s and x''_e, while $\Delta X''$ is the final interval in the virtual view. Meanwhile, let ΔZ be the depth difference between point s (with depth value d_s) and point e (with depth value d_e).

With the above notations, we obtain

$$\begin{aligned}\Delta X' = x'_s - x'_e &= \frac{\lambda}{2}\cdot f_x \cdot \left(\frac{d_s}{255}\cdot\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right)+\frac{1}{Z_{\text{far}}}\right)\\ &\quad -\frac{\lambda}{2}\cdot f_x \cdot \left(\frac{d_e}{255}\cdot\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right)+\frac{1}{Z_{\text{far}}}\right)+1\\ &= \frac{\lambda}{2}\cdot f_x \cdot \left(\frac{\Delta Z}{255}\cdot\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right)+\frac{1}{Z_{\text{far}}}\right)+1 \qquad (22)\end{aligned}$$

where constants Z_{near} and Z_{far} are the camera's parameters that represent the depth limit, and $\lambda/2$ and f_x are the baseline and focal length, respectively.

After rounding, (22) becomes

$$\begin{aligned}\Delta X'' &= x''_s - x''_e\\ &= round\left(\frac{\lambda}{2}\cdot f_x \cdot \left(\frac{d_s}{255}\cdot\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right)+\frac{1}{Z_{\text{far}}}\right)\right.\\ &\quad \left. -round\left(\frac{\lambda}{2}\cdot f_x \cdot \left(\frac{d_e}{255}\cdot\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right)+\frac{1}{Z_{\text{far}}}\right)\right)+1. \qquad (23)\end{aligned}$$

Comparing (22) and (23), we find the following.

1) If $\Delta X' = 0$, overlap will occur and $\Delta X'' = 0$.
2) If $0 < \Delta X' < 1$, one case is that point e'' overlaps with point s'' and $\Delta X'' = 0$, while another case is that these two points still maintain the adjacent position with $\Delta X'' = 1$.
3) If $\Delta X' = 1$, these two points must be adjacent and $\Delta X'' = 1$.
4) If $1 < \Delta X' < 2$, one case is that these two points still maintain the adjacent position with $\Delta X'' = 1$, while another one is that there will be a crack between them and $\Delta X'' = 2$.
5) If $\Delta X' = 2$, there must be a crack and $\Delta X'' = 2$.
6) If $2 < \Delta X' < 3$, one case is that there will be a crack between these two points with $\Delta X'' = 2$, while another one is that there will be a dis-occlusion between them and $\Delta X'' = 3$.
7) If $\Delta X' = 3$, there must be a dis-occlusion and $\Delta X'' = 3$.

In summary, to distinguish dis-occlusions from cracks, we set $\Delta X' = 2$ and then calculate $\Delta Z_{\max}$ through (22) as the threshold so that all the dis-occlusion can be found.

ACKNOWLEDGMENT

The authors would like to thank Prof. M.-T. Sun of Washington University for his valuable suggestions and kind help.

REFERENCES

[1] C. Fehn, "A 3D-TV approach using depth-image-based rendering (DIBR)," in *Proc. Vis., Imag. Image Process.*, Sep. 2003, pp. 482–487.
[2] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2007, pp. I-201–I-204.
[3] M. M. Oliveira, G. Bishop, and D. McAllister, "Relief texture mapping," in *Proc. SIGGRAPH*, 2000, pp. 359–368.
[4] D. Min, D. Kim, and K. Sohn, "Virtual view rendering system for 3DTV," in *Proc. 3DTV Conf.*, May 2008, pp. 249–252.
[5] J. J. Lei *et al.*, "Depth sensation enhancement for multiple virtual view rendering," *IEEE Trans. Multimedia*, vol. 17, no. 4, pp. 457–469, Apr. 2015.
[6] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *Proc. SPIE, Appl. Digital Image Process.*, Aug. 2009, vol. 7443, pp. 74430T-1–74430T-11.
[7] Y. R. Horng, Y. C. Tseng, and T.-S. Chang, "VLSI architecture for real-time HD1080p view synthesis engine," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 9, pp. 1329–1340, Sep. 2011.
[8] P. K. Tsung, P. C. Lin, L. F. Ding, S. Y. Chien, and L. G. Chen, "Single iteration view interpolation for multiview video applications," in *Proc. 3DTV Conf.*, May 2009, pp. 1–4.
[9] K. H. Chen, "Reducing computation redundancy for high-efficiency view synthesis," in *Proc. IEEE Int. Symp. VLSI Des., Automat. Test*, Apr. 2013, pp. 1–4.
[10] K. R. Vijayanagar, J. Kim, Y. Lee, and J. B. Kim, "Efficient view synthesis for multi-view video plus depth," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 2197–2201.
[11] *3D-HEVC Test Model 1*, doc. JCT3V-A1005, ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Jul. 2012.
[12] *Analysis of View Synthesis Methods (VSRS 1D fast and VSRS3.5)*, doc. JCT3V-B0124, ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Oct. 2012.
[13] M. Tanimoto, T. Fujii, and K. Suzuki, *View Synthesis Algorithm in View Synthesis Reference Software 3.5 (VSRS3. 5)*, ISO/IEC JTC1/SC29/WG11 (MPEG), doc. M16090, 2009. [Online]. Available: http://wg11.sc29.org/svn/repos/MPEG-4/test/trunk/3D/view_synthesis/VSRS
[14] *Test Model 8 of 3D-HEVC and MV-HEVC*, ITU-T SG 16 WP3 and ISO/IEC JTC 1/SC 29/WG 11, doc. JCT3V-H1003, Apr. 2014. [Online]. Available: https://hevc.hhi.fraunhofer.de/trac/3d-hevc/browser/3DVCSoftware
[15] S. Ma, S. Q. Wang, and W. Gao, "Low complexity adaptive view synthesis optimization in HEVC based 3D video coding," *IEEE Trans. Multimedia.*, vol. 16, no. 1, pp. 266–271, Jan. 2014.
[16] Y. Mori, N. Fukushima, T. Fujii, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," *Signal Process., Image Commun.*, vol. 24, no. 1, pp. 65–72, 2009.
[17] N. Greene, M. Kass, and G. Miller, "Hierarchical z-buffer visibility," in *Proc. SIGGRAPH*, 1993, pp. 231–240.
[18] N.-N. Patrick *et al.*, "Depth image-based rendering with advanced texture synthesis for 3-D video,"*IEEE Trans. Multimedia.*, vol. 13, no. 3, pp. 453–465, Jun. 2011.
[19] Y. Zhao, C. Zhu, Z. Chen, D. Tian, and L. Yu, "Depth no-synthesis-error model for view synthesis in 3-D video," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2221–2228, Aug. 2011.
[20] Y. Zhao, C. Zhu, Z. Chen, D. Tian, and L. Yu, "Boundary artifact reduction in view synthesis of 3D-video: From perspective of texture-depth alignment," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 510–522, Jun. 2011.
[21] K. Y. Chen, P. K. Tsung, P. C. Lin, H. J. Yang, and L. G. Chen, "Hybrid motion/depth-oriented inpainting for virtual view synthesis in multiview applications," in *Proc. 3DTV Conf.*, Jun. 2010, pp. 1–4.
[22] X. Lu, F. Wei, and F. Chen, "Foreground-object-protected depth map smoothing for DIBR," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Jul. 2012, pp. 339–343.
[23] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[24] "3DV sequences of Nagoya University," Nagoya Univ., Nagoya, Japan, Mar. 2008. [Online]. Available: http://www.tanimoto.nuee.nagoya-u.ac.jp/mpeg/mpeg-ftv.html
[25] "3DV sequences of ETRI and GIST," Electron. Telecommun. Res. Institute Gwangju Inst. Sci. Technol., Daejon, Korea, Apr. 2008. [Online]. Available: ftp://203.253.130.48
[26] "3DV sequences of HHI," Fraunhofer Heinrich Hertz Inst., Berlin, Germany, Sep. 2013, [Online]. Available: ftp://ftp.hhi.de/HHIMPEG3DV
[27] F. Bruls, R. K. Gunnewiek, and P. Walle, *Philips response to new call for 3DV test material: Arrive book and mobile*, ISO/IECJTC1/SC29/WG11 doc. M16420, Apr. 2009.
[28] G. Tech, K. Wegner, Y. Chen, and S. Yea, *3D-HEVC Test Model 1*, JCT3V-A1005, Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 1st Meeting, Stockholm, Sweden, Jul. 2012.
[29] P.-J. Lee and Effendi, "Nongeometric distortion smoothing approach for depth map preprocessing," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 246–254, Apr. 2011.

Jian Jin received the B.S. and M.E. degrees in economics and management and electronic engineering from the Taiyuan University of Science and Technology, Shanxi, China, in 2011 and 2014, respectively, and is currently working toward the Ph.D. degree in computer science from Beijing Jiaotong University, Beijing, China.

His research interests include image/video compression, 3-D video processing, and 3-D computer vision.

Anhong Wang was born in Shanxi Province, China, in 1972. She received the B.E. and M.E. degrees from the Taiyuan University of Science and Technology (TYUST), Taiyuan, China, in 1994 and 2002, respectively, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University, Beijing, China, in 2009.

She became an Associate Professor in 2005 and a Professor in 2009, both with TYUST. She is currently the Director of the Institute of Digital Media and Communication, TYUST. She has authored or coauthored more than 70 papers. She is currently leading two national research projects with the National Science Foundation of China. Her research interest includes image/video coding, compressed sensing, and secret image sharing.

Yao Zhao (M'06–SM'12) received the B.S. degree in radio engineering from Fuzhou University, Fuzhou, China, in 1989, the M.E. degree in radio engineering from Southeast University, Nanjing, China, in 1992, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University (BJTU), Beijing, China, in 1996.

He became an Associate Professor in 1998 and a Professor in 2001, both with BJTU. From 2001 to 2002, he was a Senior Research Fellow with the Information and Communication Theory Group, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands. He is currently the Director of the Institute of Information Science, BJTU. He is currently leading several national research projects from the 973 Program, 863 Program, and the National Science Foundation of China. His current research interests include image/video coding, digital watermarking and forensics, and video analysis and understanding.

He serves on the Editorial Board of several international journals, including as an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS, an Associate Editor of the IEEE SIGNAL PROCESSING LETTERS, the Area Editor of *Signal Processing: Image Communication*, and an Associate Editor of *Circuits, System, and Signal Processing*. He was named a Distinguished Young Scholar by the National Science Foundation of China in 2010, and was elected as a Chang Jiang Scholar of the Ministry of Education of China in 2013.

Prof. Zhao is an IET Fellow.

Chunyu Lin was born in Liaoning Province, China. He received the Ph.D. degree from Beijing Jiaotong University, Beijing, China, in 2011.

From 2009 to 2010, he was a Visiting Researcher with the ICT Group, Delft University of Technology, Delft, The Netherlands. From 2011 to 2012, he was a Postdoctoral Researcher with the Multimedia Laboratory, Ghent University, Ghent, Belgium. He is currently an Associate Professor with Beijing Jiaotong University. His current research interests include image/video compression and robust transmission, 2D-to-3D conversion, and 3-D video processing.

Bing Zeng (F'91–SM'13–F'16) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1991.

He was a Postdoctoral Fellow with the University of Toronto, Toronto, ON, Canada, from September 1991 to July 1992, and a Researcher with Concordia University, Bronxville, NY, USA, from August 1992 to January 1993. He then joined the Hong Kong University of Science and Technology (HKUST), Hong Kong, China. Through China's "1000-Talent-Scheme," he returned to UESTC in the summer of 2013, where he currently leads the Institute of Image Processing. He has received more than 20 research grants, filed 8 international patents, and authored or coauthored more than 150 papers, published in journals such as the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He has been SCI-cited more than 870 times (Google cited more than 1950 times) and currently stands at the seventh position among all papers published in the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY

Mr. Zeng served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY for eight years, and was the recipient of the Best Associate Editor Award in 2012. He is currently on the Editorial Board of the Journal of Visual Communication and Image Representation and is the General Co-Cchair of the IEEE VCIP-2016, to be held in Chengdu, China, in November 2016. He was the recipient of the 2011 IEEE CSVT Transactions Best Paper Award, as well as the Best Paper Award from China-Com in 2009, 2010, and 2012. He was the first recipient of a Second Class Natural Science Award from the Chinese Ministry of Education in 2014.