EFFICIENT VIEW SYNTHESIS SCHEME WITH RAY CASTING AND PULL-PUSH TECHNIQUES

Ku-Chu Wei[1]*, Yung-Lin Huang*[2]*, Shao-Yi Chien*[1]

Media IC & System Lab
[1]Graduate Institute of Electronics Engineering and Department of Electrical Engineering
[2]Graduate Institute of Networking and Multimedia
National Taiwan University
1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan, R.O.C.
{kcwei, cary}@media.ee.ntu.edu.tw, sychien@cc.ee.ntu.edu.tw

ABSTRACT

View synthesis, composed of depth-image-based rendering followed by hole-filling, is a crucial technology for 3D TV and free-viewpoint TV. To realize view synthesis in practical systems, the efficiency of view synthesis must be considered to achieve good a trade-off between the image quality and the computational complexity. We propose a efficient view synthesis scheme that, when compared to state-of-the-art backward warping, requires only half of the runtime with comparable quality. Specifically, the proposed scheme uses ray casting and pull-push processing to render in one pass, which can be regarded as applying 3D filters in the depth-image-based rendering. Moreover, the proposed scheme can benefit the hole-filling process to further improve the efficiency of view synthesis.

Index Terms— View synthesis, depth-image-based rendering, free-viewpoint TV, ray casting, pull-push algorithm

1. INTRODUCTION

With the huge improvement of 3D TV, the representation of 3D videos has brought lots of researches, seeking for better compression efficiency with the independence to display technology. Among various representations, video plus depth (V+D) can fulfill the requirements with the aid of depth-image-based rendering (DIBR) [1]. The receiver side of 3D systems can flexibly render virtual views to suit different types of display devices. The rendered virtual views for 3D TV are constrained to a narrow baseline, often specialized into 1D cases. Stepping forward, free-viewpoint TV (FTV) allows more flexibility by breaking the limitation of camera configurations. Although previous works specialized for 1D view synthesis cannot be directly employed in FTV systems, DIBR itself can be applied to general FTV applications. With the MVD (multi-view plus depth) representation, an extension of V+D representation, the virtual views can be synthesized by DIBR given the depth maps of every reference views. However, unlike 2D warping, holes appear due to physical occlusions in 3D warping. A mature image processing technique, inpainting [2], can fill such regions and therefore implemented in most current view synthesis systems [3][4].

In this paper, we focus on how to improve the efficiency of the implementation of DIBR rather than the hole-filling process. However, we will also demonstrate that by improving the DIBR implementation properly, the hole-filling process can be benefitted, resulting in an overall enhancement for view synthesis.

2. PREVIOUS WORKS

Two classical DIBR implementations, forward warping and backward warping, are briefly introduced in this section.

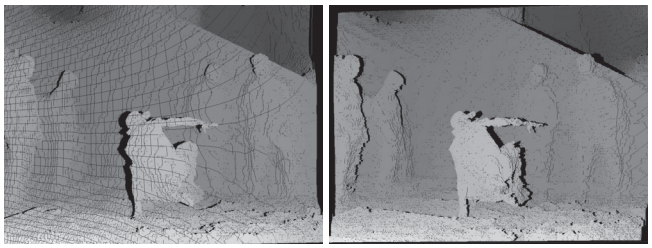
2.1. One-pass Forward Warping

One-pass forward warping is the basic implementation of DIBR, where both the structure and texture information are mapped to the virtual view simultaneously. First, 3D warping is employed to transform pixels from input reference views to the virtual view. Assume $\mathbf{A}_r$ and $\left[\mathbf{R}_r|\mathbf{t}_r\right]$ are the intrinsic and extrinsic parameters of the reference view respectively. Given a pixel position (x_r, y_r) in the image coordinate of the reference view and the corresponding depth value z_r, the world coordinate of the 3D point $\mathbf{P}_w = (x_w, y_w, z_w)^{\mathrm{T}}$ is calculated by

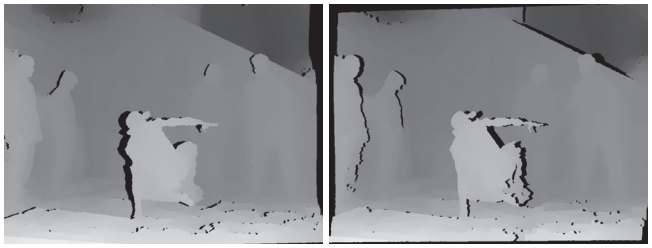
$$\mathbf{P}_w = \mathbf{R}_r^{-1}\left(z_r\mathbf{A}_r^{-1}\mathbf{p}_r - \mathbf{t}_r\right), \tag{1}$$

where $\mathbf{p}_r = (x_r, y_r, 1)^{\mathrm{T}}$. Assume $\mathbf{A}_v$ and $\left[\mathbf{R}_v|\mathbf{t}_v\right]$ are the intrinsic and extrinsic parameters of the virtual view respectively. The mapped position in the image coordinate of the virtual view (x_{rv}, y_{rv}) with depth z_{rv} can be further derived by

$$z_{rv}\mathbf{p}_{rv} = \mathbf{A}_v\left(\mathbf{R}_v\mathbf{P}_w + \mathbf{t}_v\right), \tag{2}$$



(a) Left view(unfiltered) (b) Right view(unfiltered)



(c) Left view(filtered) (d) Right view(filtered)

Fig. 1. Depth filtering in backward warping. Top rows are the two warped partial depth maps, while the bottom rows are the depth maps filtered by 5x5 median filters. The cracks are removed after filtering.

where $\mathbf{p}_{rv} = (x_{rv}, y_{rv}, 1)^{\mathrm{T}}$.

A visibility test (z test) is subsequently employed to determine if the pixel should be removed or kept. A depth buffer $\mathbf{z}_v$ is used to record the nearest depth value from the virtual viewpoint, and the visibility is therefore tested by

$$\mathbf{z}_v(x_v, y_v) = \min\{z_{rv} | (x_{rv}, y_{rv}) \equiv (x_v, y_v)\}. \quad (3)$$

Once the depth buffer $\mathbf{z}_v(x_{rv}, y_{rv})$ has been updated, the corresponding color value $\mathbf{c}_v(x_{rv}, y_{rv})$ is updated simultaneously. That is, the texture and structure information update synchronously. However, two main problems appear for this approach. First, due to the discrete sampling, noticeable cracks appear inevitably when synthesizing a novel-view image, as shown in Fig. 1(a)(b). Second, different white balance and exposure settings among the reference cameras can result in a poor rendered image as shown in Fig. 6(a). Generally, it is difficult to directly post-process on the rendered virtual color image.

2.2. Backward Warping

To conquer the aforementioned problems, backward warping [5][6] uses inverse mapping with pre-filtered reference images. The depth mapping and the color mapping are separated in two passes. In the first pass, the reference depth map $\mathbf{D}_{r,k}$ of each view k is warped to the virtual view to generate a partial virtual depth map $\mathbf{D}_{v,k}$, just identical as forward warping does, as shown in Fig. 1(a)(b). Then the partial virtual depth map $\mathbf{D}_{v,k}$ is filtered by edge-preserving filters, such as bilateral filters or median filters, to eliminate the cracks without destroying the structure, as shown in Fig. 1(c)(d). The color values are determined by backward mapping from the filtered partial virtual depth maps in the second pass. Due to the digital sampling, the back-projected positions are usually non-integer values. Pre-filtering the reference images by bilinear interpolation (or bicubic interpolation) is consequently performed to eliminate texture aliasing. Sophisticated backward warping methods would further blend color values from all visible reference views to generate a much better virtual view, with an extra visibility test on input reference views.

Note that only depth maps are filtered by the edge-preserving filters because they consist of structure information only, while color images consist of the combination of structure and texture information. It is relatively safe to interpolate depth values than interpolate color values. Although good rendered results can be derived from backward warping, the required two-pass processing and extra storage (partial depth maps and partial color images) lead to higher cost for practical implementation. This paper aims to propose an efficient one-pass view synthesis scheme with good performance.

3. PROPOSED VIEW SYNTHESIS SCHEME

The key that backward warping can achieve better results is filtering the depth maps and the color images separately. Depth maps are filtered by edge-preserving filters in the first pass, whereas color images are filtered by bilinear interpolation in the second pass. To break the two-pass scheme, we employ ray casting and a pull-push algorithm instead. The overall flowchart of the proposed scheme is shown in Fig. 2. Ray casting creates 3D filters in the world space, whereas pull-push algorithm filters the virtual color buffer $\mathbf{c}_v$ with the aid of the depth buffer $\mathbf{z}_v$ and the weight buffer $\mathbf{w}_v$. The detail of each block will be explained in the following paragraphs.

3.1. Ray Casting

Generally more than one point are mapped to the same pixel for view synthesis. The conventional one-pass forward warping generates objectionable artifacts due to not blending the color information. Different from the 2D filters used in the backward warping, points are directly filtered in the 3D world coordinate, as shown in Fig. 3, which is more reasonable than the 2D filters used in the backward warping. A 2D pixel corresponds to a frustum in the 3D space under perspective projection, and all points in the same frustum would be blended.

In practice, a modified visibility test (soft-z test) is performed as the 3D filters. Potentially visible points that map to the same pixel are all blended. However, a direct implementation of the soft-z test needs two passes: the first pass determines the information of the depth buffer $\mathbf{z}_v$ by z test, and the second pass verifies whether the mapped point (x, y, z)

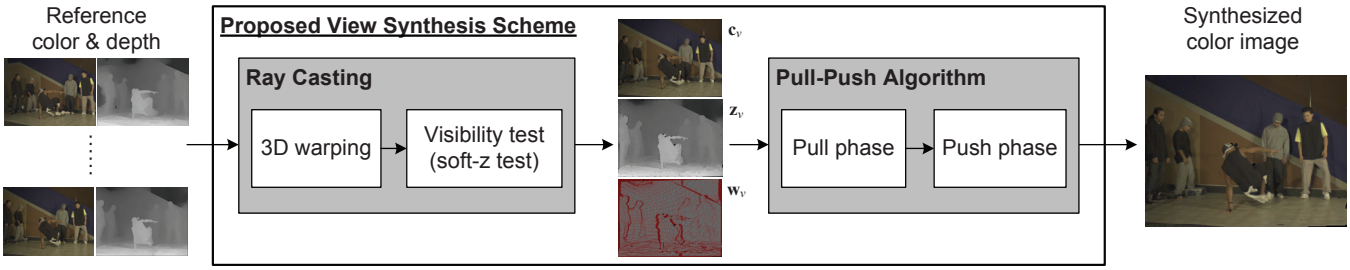


Fig. 2. Flowchart of proposed view synthesis scheme.

Algorithm 1 One-pass soft-z test

Require: depth threshold Δz
  color buffer $\mathbf{c}_v \leftarrow 0$
  depth buffer $\mathbf{z}_v \leftarrow \infty$
  weight buffer $\mathbf{w}_v \leftarrow 0$
  for each warped point (x, y, z) with color $\mathbf{c}$ **do**
    if $z < \mathbf{z}_v(x, y) - \Delta z$ **then**
      $\mathbf{z}_v(x, y) \leftarrow z$
      $\mathbf{c}_v(x, y) \leftarrow \mathbf{c}$
      $\mathbf{w}_v(x, y) \leftarrow 1$
    else if $\mathbf{z}_v(x, y) - \Delta z \le z \le \mathbf{z}_v(x, y) + \Delta z$ **then**
      $\mathbf{z}_v(x, y) \leftarrow z$
      $\mathbf{c}_v(x, y) \leftarrow \frac{\mathbf{c}+\mathbf{w}_v(x,y)\times\mathbf{c}_v(x,y)}{\mathbf{w}_v(x,y)+1}$
      $\mathbf{w}_v(x, y) \leftarrow \mathbf{w}_v(x, y) + 1$
    end if
  end for

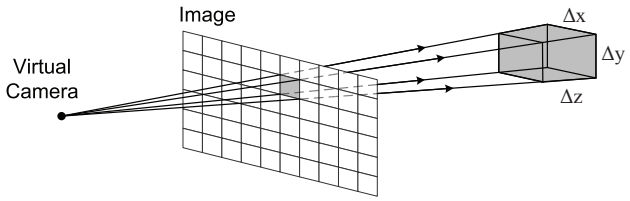


Fig. 3. Illustration of ray casting.

is potentially visible by checking if $z < \mathbf{z}_v(x, y) + \Delta z$. To improve the efficiency of ray casting, we further modify the soft-z test to be performed in one pass by using dynamic programming, as listed in Algorithm 1. The one-pass soft-z test requires an extra weight buffer $\mathbf{w}_v$. However, the execution time is halved with minor quality loss compared to the original two-pass soft-z test.

3.2. Pull-push Algorithm

After ray casting, the virtual image still has lots of cracks due to digital sampling. To eliminate the cracks, edge-preserving filters can be employed in backward warping (see Sec. 2.2). However, it is hard to determine the proper kernel size of the filters for sequences with different resolutions. Therefore the pull-push algorithm [7], which interpolates data from scattered points by two phases, is adopted in the proposed scheme. The pull phase (first phase) generates a series of downsampled images, and the push phase (second phase) uses the downsampled images to reconstruct the images, as illustrated in Fig. 4. Different from the setting of conventional pull-push algorithms, we have not only the color buffer $\mathbf{c}_v$, but also the depth buffer $\mathbf{z}_v$ and the weight buffer $\mathbf{w}_v$. Hence the algorithm is further modified to enhance the performance in the view synthesis scheme.

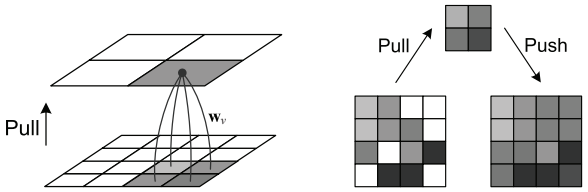


Fig. 4. Illustration of the pull-push algorithm.

3.2.1. *Pull Phase*

Noticeable noise for real-world sequences appears due to incorrectly estimated depth values. To reduce the effect from such noise, the pyramid of the depth buffer $\mathbf{z}_v$ is constructed first by the following equation:

$$\mathbf{z}_{v,1}(x, y) = \frac{\sum_{i=2x}^{2x+1}\sum_{j=2y}^{2y+1}\mathbf{w}_v(i, j)\mathbf{z}_v(i, j)}{\sum_{i=2x}^{2x+1}\sum_{j=2y}^{2y+1}\mathbf{w}_v(i, j)}. \quad (4)$$

The downsampled depth buffer $\mathbf{z}_{v,1}$ is then used to check if there exists odd visible depth values in $\mathbf{z}_v$ by

$$\mathbf{z}_v(x, y) - \mathbf{z}_{v,1}(x/2, y/2) > \Delta z. \quad (5)$$

For each 2x2 window, if only one depth value exceeds the threshold Δz, the depth value has high probability of being wrong. Those pixels are marked as $\mathbf{u}(x, y) = 0$, and the other pixels are marked as $\mathbf{u}(x, y) = 1$. With the extra marked information, the pyramid of the color buffer $\mathbf{c}_v$ is then constructed by

$$\mathbf{c}_{v,1}(x, y) = \frac{\sum_{i=2x}^{2x+1}\sum_{j=2y}^{2y+1}\mathbf{u}(i, j)\mathbf{w}_v(i, j)\mathbf{c}_v(i, j)}{\sum_{i=2x}^{2x+1}\sum_{j=2y}^{2y+1}\mathbf{u}(i, j)\mathbf{w}_v(i, j)}. \quad (6)$$

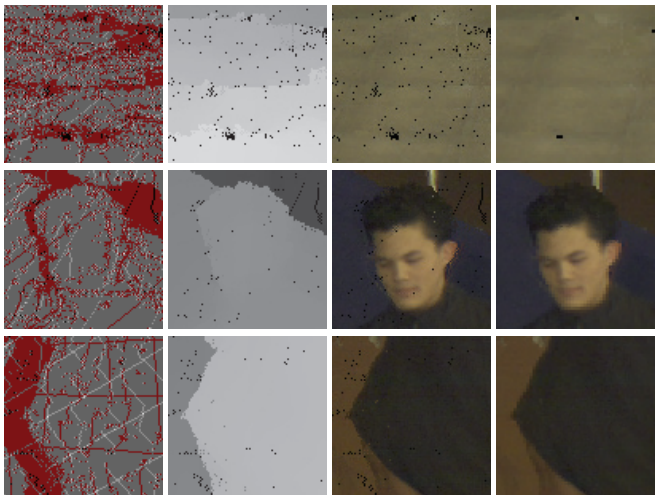


Fig. 5. Results of the pull-push algorithm. The first column visualizes the weight buffer $\mathbf{w}_v$. Brighter pixels represent higher weight, and red pixels denotes those who contributed from only one point. The second and the third columns are the depth buffer $\mathbf{z}_v$ and the color buffer $\mathbf{c}_v$ before the pull-push processing. The last column shows the pull-push processed color buffer $\mathbf{c}_v$.

As a result, the color value $\mathbf{c}_v(x, y)$ would not be averaged if the pixel is marked as uncertainty, $\mathbf{u}(x, y) = 0$, which makes the view synthesis being more resistant to the sparse noise.

3.2.2. Push Phase

The pixel values are then refilled in the push phase under the following three cases: pixels without color information, or $\mathbf{w}_v(x, y) = 0$, because those are where the cracks appear; pixels contributed from one point only, or $\mathbf{w}_v(x, y) = 1$, because the pixels have high possibility to be quite different from adjacent pixels; pixels with large depth error, or $\mathbf{u}(x, y) = 0$, because the large depth difference compared to adjacent pixels indicates the depth of the pixel has high probability to be wrong. The results of the pull-push algorithm are shown in Fig. 5. It can be seen that cracks are removed and error pixels are corrected by the pull-push algorithm. The remaining black regions are indeed the occluded regions, which are filled by inpainting.

4. EXPERIMENTAL RESULTS

To compare the view synthesis efficiency of the proposed algorithm, we also implement both one-pass forward warping and backward warping as mentioned in Section 2. As for the edge-preserving filter in the backward warping, 3x3 median filters are used to smooth the depth maps faster, which is identical to the setting in the MPEG view synthesis reference software (VSRS) [8]. The hole-filling are finished by the Navier-Stokes method [2], which has been implemented in many open source libraries. Note that the depth threshold Δz influences the image quality largely, and it is set to be half of the z-direction quantization interval. That is,

$$\Delta z = \frac{1}{2} \times \frac{z_{far} - z_{near}}{255}, \tag{7}$$

which is sequence-independent.

Last, all the programs are run on a PC with i7-2600 CPU and 4GB memory, and the OS is Win7 64-bit.

Table 1. PSNR and the execution time of rendering view 6 from view 5 and view 7 for sequence Breakdancers.

Method	PSNR (dB)	Time (sec.) (w/o inpainting)	Time (sec.) (w/ inpainting)
Forward	31.8038	**0.573**	**0.815**
Backward	**33.7382**	2.891	2.935
Proposed	**33.5162**	**0.656**	**0.676**

4.1. MSR 3D video

The sequence Breakdancers produced by Interactive Visual Group at Microsoft Research [9] is used for the experiments, and the sequence contains 100 frames from 8 different views. View 5 and view 7 are selected as the input reference views, and the camera parameter of the virtual view is set identical to view 6 for all view synthesis algorithms. Fig. 6 shows the rendered results by three approaches: forward warping, backward warping, and the proposed scheme. Note that the white balance setting of view 5 and view 7 is slightly different for this sequence. Discontinuous color values on the background wall are visible for the result of forward warping, while both backward warping and the proposed scheme eliminate such artifacts. The PSNR evaluation and the execution time are listed in Table 1. It can be seen that the proposed scheme performs similarly as backward warping does; both outperforms forward warping, whereas the proposed scheme only requires less than one quarter time of backward warping. Efficient view synthesis is achieved by the proposed scheme.

The visibility test in the proposed scheme is soft-z test, which is more complicated than the z test in the forward warping is. Thus, the execution time of the proposed approach is expected to be longer than that of forward warping. However, the execution time including inpainting of the proposed scheme is surprisingly even shorter than that of forward warping. Since the pull-push algorithm is employed after ray casting, most cracks as well as the small occlusion regions are eliminated. The regions needing inpainting become much smaller than that of forward warping, even smaller than that of backward warping does, and the execution time reflects the facts consequently.

Table 2. PSNR and the execution time of rendering view 4 from different input reference views for sequence Breakdancers. The input reference views: Near - view 3 & view 5, Medium - view 2 & view 6, Far - view 1 & view 7. It can be seen that the execution time (with inpainting) of the proposed scheme is even shorter that that of one-pass forward warping.

Method	PSNR (dB)			Time (w/o inpainting)(sec.)			Time (w/ inpainting)(sec.)		
	Near	Medium	Far	Near	Medium	Far	Near	Medium	Far
Forward	32.2530	31.3266	29.9668	0.559	0.561	0.552	0.758	1.685	2.646
Backward	33.9458	33.1014	31.6309	2.779	2.553	2.437	2.838	2.721	2.880
Proposed	33.5339	33.1245	31.6121	0.668	0.667	0.674	0.707	0.733	0.861

Another experiment is further performed to observe the effect of the baseline distance between the reference views. The camera parameter of the rendered virtual view is set identical to view 4. For input reference views, view 3 and view 5 are used as small baseline, view 2 and view 6 are used as medium baseline, and large baseline uses view 1 and view 7. The PSNR evaluation and the execution time from the three different baseline distances are listed in Table 2. Similar to the results of the previous experiment, the PSNR of the proposed scheme performs almost the same as that of backward warping. Moreover, the proposed scheme even outperforms backward warping at the medium baseline. The execution time without inpainting of forward warping and that of the proposed scheme are almost identical among different baseline distances. However, when taking the inpainting time into consideration, forward warping takes almost the same time that backward warping requires, while the proposed scheme takes only one third time that forward warping requires. As the baseline distance is getting larger, the inpainting time becomes dominant rather than DIBR. With the proposed pull-push algorithm, subsequent inpainting can be benefitted, resulting in an overall efficiency enhancement.

4.2. Computer-generated (CG) sequence

We use a computer-generated (CG) sequence to further evaluate how the baseline distance influences the inpainting time, by freely controlling the baseline. Notice that the depth maps are perfect, allowing us to analyze the causality precisely. The CG model Sponza produced by Crytek is used in this paper. The baseline distance is set into 10 levels, and the baseline distance is proportional to the level; that is, the baseline distance of level 7 is exactly $7/4$ times than that of level 4.

The inpainting time of each level is shown in Fig. 7. As the baseline distance is getting larger, the required inpainting time increases correspondingly. The forward warping requires the longest inpainting time among the three methods, while the proposed scheme requires the shortest time. The reduction ratio on inpainting is roughly constant with the baseline distance. The backward warping requires about 75% time of forward warping, and the proposed scheme only needs about 60% time.

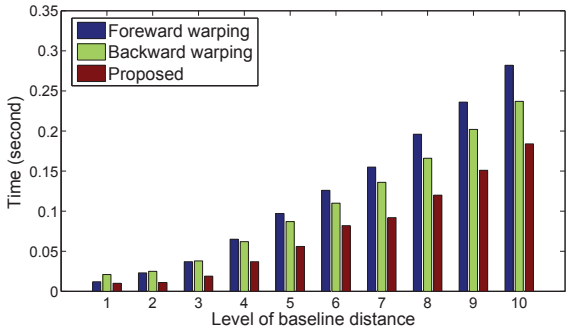
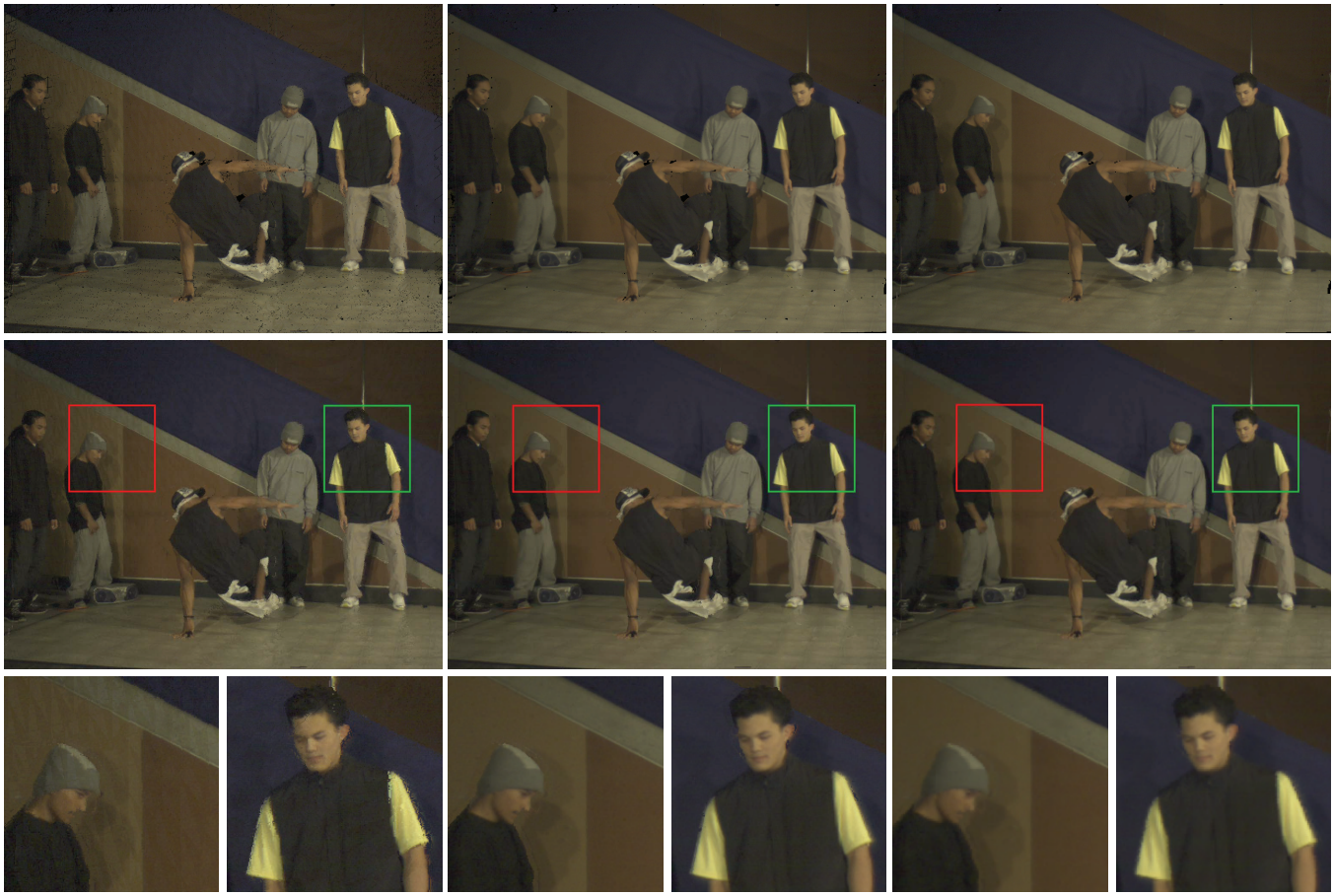


Fig. 7. Comparison of the inpainting time with different baseline distance for sequence Sponza.

5. CONCLUSION

We propose an efficient view synthesis scheme with ray casting and pull-push algorithm. Only one pass is required to synthesize novel views, resulting in a great trade-off between the resource cost and the quality. The required time is slightly more than that of forward warping, while similar PSNR is achieved compared to that of backward warping. Moreover, with the proposed pull-push algorithm, both cracks and small occlusion regions can be eliminate simultaneously. The subsequent inpainting time is therefore reduced, making the proposed method even faster than forward warping. Note that as the number of input views increases, the overhead of backward warping increases largely, while our method has no such problems. With the proposed efficient view synthesis scheme, the development of FTV can be benefited.

In the current implementation, the depth threshold Δz in both ray casting and pull-push algorithm is set as a constant (sequence-independent). However, Δz should adapt according to the properties of the input sequence, such as the distribution of depth maps. A possible future direction is to analyze the optimal determination of Δz, which may be sequence-dependent or location-dependent. Moreover, we observe that the pull-push algorithm is highly suitable for the view synthesis application. Although only two-level pyramids are used in this paper, it is possible to further extend the existing pull-push algorithm by adopting multi-level pyramids.



(a) Forward Warping (b) Backward Warping (c) Proposed

Fig. 6. Comparison of the rendered results for sequence Breakdancers. The top row are the results without hole-filling, and the second row are the results after hole-filling. The details of selected patches are shown in the bottom row.

6. REFERENCES

[1] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *Proc. SPIE 5291, Stereoscopic Displays and Virtual Reality Systems XI*, 2004.

[2] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Computer Vision and Pattern Recognition (CVPR)*, 2001.

[3] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-D video," in *Picture Coding Symposium*, May 2009.

[4] I. Daribo and H. Saito, "A novel inpainting-based layered depth video for 3DTV," *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 533 – 541, June 2011.

[5] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," *Journal of Image Communication*, vol. 24, no. 1-2, pp. 65–72, January 2009.

[6] D. Berjon, A. Hornung, F. Moran, and A. Smolic, "Evaluation of backward mapping DIBR for FVV applications," in *Multimedia and Expo (ICME), IEEE International Conference on*, July 2011.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *SIGGRAPH*, New York, NY, USA, 1996.

[8] ISO/IEC JTC1/SC29/WG11, MPEG, *View Synthesis Software Manual*, September 2009, release 3.5.

[9] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, Aug. 2004.