Depth Image Based Rendering with Inverse Mapping

Muhammad Shahid Farid, Maurizio Lucenteforte, Marco Grangetto

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, ITALY
lastname@di.unito.it

***Abstract*—Three-dimensional video has gained much attention during the last decade due its vast applications in cinema, television, animation and virtual reality. The design of intermediate view synthesis algorithms that are efficient both in terms of computational complexity and visual quality is a paramount goal in the fields of 3D free view point television and displays. This papers focuses on the design of a low complexity view synthesis algorithm that produces better quality of the virtual image. A novel view synthesis technique to create a virtual view from two video sequences with corresponding depths is proposed. The technique employs low complexity integer pixel precision warping and a novel approach for hole filling based on inverse mapping. The proposed technique is tested over a number of video sequences and compared with existing state of the art methods, yielding excellent results both in terms of signal to noise ratio and visual quality.**

***Index Terms*—3D-TV, Depth image based rendering, View synthesis, 3D warping**

I. Introduction

The quest for novel and efficient 3D video representations and coding techniques in area of 3D television (3DTV) has recently revitalized the research in the area of intermediate view synthesis. In fact, emerging 3D display technologies, such as autostereoscopic displays, require the availability of several images corresponding to different point of views. It turns out that the possibility to synthesize intermediate views from a limited set of original video sequences has gained much attention in the last years. View synthesis has the potential to greatly reduce the amount of video data to be transmitted and can be integrated in the common hybrid predictive coding approaches to capture the inter-view redundancy.

The concept of creating a virtual view from two or more views has been coined in the computer vision literature almost four decades ago [1] using different terms like view synthesis [2], view morphing [3] and image metamorphosis [4]. A number of techniques to create a virtual view from two or more views have been proposed. They may be classified into three broad categories [5]. Any view can be generated if the scene is represented with its 3D structure, which is usually not reasonable for natural video shots. The second class, known as *depth image based rendering* (DIBR), uses a number of views with implicit geometry like depth of the scene to create intermediate views [6], [7]. In the third category, only images are used to create a new view without depth or geometry information like [8]. The view synthesis technique proposed in this paper falls in the second category: in particular, we consider the common problem to estimate an intermediate view in between two original views plus the corresponding depth maps. All DIBR techniques comprise two main algorithmic steps; first, the intermediate virtual view is generated by warping the original pixels to the corresponding intermediate positions, based on depths and camera parameters. After that, two warped views are blended to create a single virtual view. Because of geometrical occlusions, depth imprecision and/or lossy coding of depth there may still be some pixels of the virtual view which are uninitialized, referred to as *holes*. In DIBR, the second step consists in estimating such missing pixels usually exploiting *image inpainting* techniques.

Many DIBR techniques have been proposed so far. An extensive review can be found in [9], [10]. Minh Do et. al. [11] proposed an intensity propagation algorithm to generate a virtual view from two views with their depths. Their algorithm works in three steps. In first step, without considering the occlusions all pixels of a view are warped to the new positions. In second step, the occluded pixels are identified with the help of their depth. In the final step, the occluded pixels' intensities are interpolated using a cubic spline function. To compute the intensities in occluded regions, texture based approach is proposed in [12]. Texture based synthesis normally generates better visual quality in case of large size occlusions but can be very expensive in terms of computational costs making it unfeasible in real time applications. Hofsetz et. al. [13] presented a technique for view synthesis when the depth maps are not accurate. The regions with inaccurate depths are termed as uncertain regions. Their approach is to determine these regions and then apply 3D ellipsoidal Gaussian kernels to render the virtual view. Schmeing and Jiang [14] proposed an image based rendering technique that first estimate the background and uses the segmentation results to fill the occluded regions. This technique yields accurate results in the particular case of scenes with only two depth layers. A number of techniques have been developed to recover synthesis holes. A hole filling algorithm that uses pixels' depths and intensities to compute the occluded regions has been presented in [15]. Another technique to compute the occluded regions by registering the two views has been presented in [16]. In [17] foreground and background weights are used to interpolate the missing pixel.

MMSP'13, Sept. 30 - Oct. 2, 2013, Pula (Sardinia), Italy.

Finally, [18], [19], [20] exploits inpainting techniques. Most of the mentioned techniques have led to the development of *View Synthesis Reference Software* (VSRS) [21] by the 3DV MPEG group and that will be used as a benchmark for our experiments.

This major contribution of this paper is a novel low complexity *View Synthesis with Inverse Mapping* (VSIM), which yields competitive results as compared to state of the art VSRS. As opposed to more complex solutions, VSIM achieves excellent results applying warping with integer pixel precision and recovering synthesis holes using a novel inverse-mapping function.

The rest of the paper is organized as follows. In Sect. II the proposed algorithm for virtual view generation is presented. In Sect. III experimental results and comparisons with existing techniques are shown and in Sect. IV our conclusions are drawn.

II. Proposed View Synthesis with Inverse Mapping (VSIM) technique

The proposed VSIM algorithm works in two steps like other view synthesis algorithm. It takes two views with depths (left view and right view) as input and computes the intermediate virtual view. Each input view is warped to the intermediate position and the two resultant virtual views are merged together to get a single virtual view. While warping the original views to intermediate positions, a mapping function is defined against each warped view. The missing pixels are then computed with the help of the mapping functions. The proposed algorithm assumes the usual horizontal shift camera setup. The following subsections describe the algorithm.

A. Pixel warping

Let vL and vR be left and right original views of size $m \times n$, and dL, dR the respective depth maps. Let fL and fR be the camera focal lengths, and bL, bR the position of the two cameras on the base line, i.e. their disparity is equal $b = bR - bL$. Usually, the depth maps are provided as 256 levels images where values 0 and 255 represents the farthest and nearest depths, respectively. The true depth values dL' (dR') are recovered from the encoded depth map dL (dR) using the following equation:

$$dL' = \frac{1}{\frac{dL}{255}\left(\frac{1}{zNear} - \frac{1}{zFar}\right) + \frac{1}{zFar}} \tag{1}$$

where $zNear$ and $zFar$ are the nearest and farthest depths in the scene.

Since the cameras are parallel with baseline b, the position of the virtual camera for intermediate view is $bL + \frac{b}{2} (= bR - \frac{b}{2})$ and original image pixels can be warped to intermediate view position by applying horizontal shifts that depend on the their depth values. If we let (u, v) represent the coordinates of a given pixel in the view (vL), then the coordinates of the same pixel in the left virtual view (vL') turns to be $(u', v') = (u, v')$ with:

$$v' = v - \frac{fL \times b}{dL'(u, v)} \tag{2}$$

The value $\frac{fL \times b}{dL'(u,v)}$ is usually termed *column shift*. Column shift will be subtracted from the pixel column value to find its position in the virtual view because the considered pixel in the left view will move leftward in the coordinate system of the virtual view. Similarly, in case of right virtual view (vR'), the column shift would be added to the pixel column value.

Using Eq. 2, the warped positions of each pixel can be easily computed. During warping more than one pixels from the original view may map to the same position in the virtual view. In this case only the foremost pixel (the one with the largest depth) will be considered. It is also possible that some locations in the virtual view remain empty, i.e. synthesis holes: these positions represent pixels that look occluded from the original view point or they can be caused by warping errors due to depth estimation or quantization errors.

Once the left virtual view vL' and right virtual view vR' have been computed applying horizontal warping, a single intermediate virtual view vM is obtained by merging the two images according to the following equation:

$$vM(u, v) = \begin{cases} \frac{vL'(u,v) + vR'(u,v)}{2}, & \text{if } hL(u, v) = hR(u, v) = 0 \\ vL'(u, v), & \text{if } hL(u, v) = 0 \wedge hR(u, v) = 1 \\ vR'(u, v), & \text{if } hL(u, v) = 1 \wedge hR(u, v) = 0 \\ 0, & otherwise \end{cases} \tag{3}$$

where hL (hR) is a binary map identifying the holes in the left (right) virtual view. According to previous equation only those pixels that are missing in both the virtual views will appear as holes in the merged view vM. A binary map $holesV$ is used to identify the location of these holes. Fig. 1 shows the whole warping process schematically.

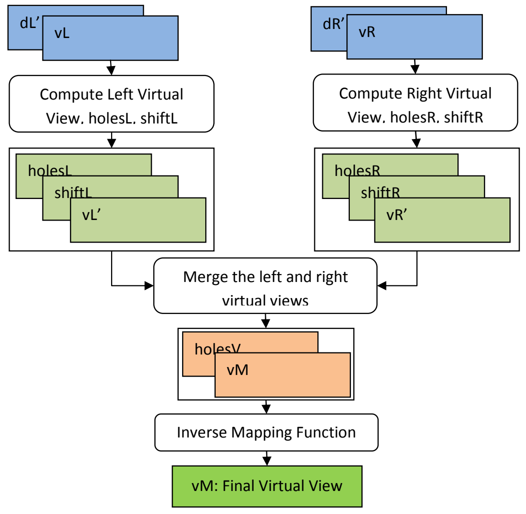

dL'
vL
dR'
vR
Compute Left Virtual View. holesL. shiftL
Compute Right Virtual View. holesR. shiftR
holesL
shiftL
vL'
holesR
shiftR
vR'
Merge the left and right virtual views
holesV
vM
Inverse Mapping Function
vM: Final Virtual View

Fig. 1: Schematic diagram of the proposed technique.

B. Holes filling through inverse mapping

Not all the pixels of the virtual view can be recovered for three main reasons that are recalled in the following.

1) *Holes due to inaccurate depth:* the depth maps may contain some errors as they are usually estimated and then quantized to 256 levels. Because of inaccurate depth values erroneous shifts may be applied to the pixel locations. Some warping techniques proposed in literature try to limit this effect by smoothing the depth maps before warping.
2) *Cracks:* the warping process does not warp the objects or surfaces as whole and this may introduce holes in the surface (we name them cracks). These cracks can be recovered by the proposed inverse mapping function. The black lines on the right hand side in Fig. 2 are example of cracks.
3) *Disocclusion:* third case is represented by disocclusions that appear when an object in the foreground uncover a portion of background that is not visible from the original view points. Such unknown background regions clearly appear as holes in the virtual view. The green areas on the right side of Fig. 2 are examples of disocclusion.

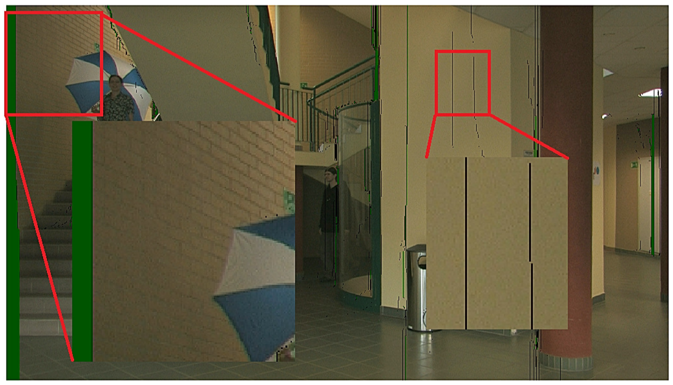


Fig. 2: Green area in enlarged rectangle on the right side of the image is the new region introduced in the scene whereas the black lines in the enlarged region on the left side of the image are cracks.

In the DIBR literature holes of the virtual view are recovered by different techniques, e.g. averaging the neighboring pixels, applying non-linear filters like median filer or by using more complex techniques like texture based inpainting. In this paper, we fill the holes by re-mapping their locations in the original view based on the column-shifts of the neighborhood. Using such information holes can be mapped backward to one of the original views so as to identify the missing pixel values. We name this technique *inverse mapping*.

An inverse mapping function can be defined for both the left and the right virtual view. The function takes a missing hole position as input and returns its approximated position in the respective original left or right view.To define this function we maintain a table that records the column shift of each pixel with respect to the virtual view. Let $shiftL$ be such a table for left virtual view: if $vL(u, v)$ is shifted to position (u, v') in vL', then $shiftL(u, v') = v$. Similarly, $shiftR$ is used for right virtual view.

The first step in inverse mapping is to determine the original locations of holes in $holesV$ by trying to interpolate the missing values in the table $shiftL$ ($shiftR$). To this end, VSIM applies a median filter to $shiftL$ and $shiftR$. We verified experimentally that the median filter yields better results than average or weighted average filters. In particular, we have observed that the median filters with kernel size 3×3 and 5×5 produce the best results. The locations recovered by applying median filter to $shiftL$ and $shiftR$ are used to determine the pixels of the original views that can be copied to fill holes of the virtual view. Due to the limited size of the median filter some holes cannot be recovered with this mechanism. We observed that iteratively increasing the size of the filter to recover all holes generally yields poor shift interpolation results with the creation of artifacts in the virtual view. We have found that the few remaining holes can be recovered by simply assuming that their depth is the same as the co-located pixels in the original views. The detailed description of inverse mapping function is shown in Algorithm 1.

Algorithm 1 Inverse Mapping Function

```
Require: $vL, vR, vM, holesV, shiftL, shiftR, zInvL,$
    $zInvR, f, b$
Ensure: $vM$: Final virtual view after filling the holes
 1: $shiftL' \leftarrow medianfilter(shiftL)$
 2: $shiftR' \leftarrow medianfilter(shiftR)$
 3: for $(u, v) \in holesV$ do
 4:   $v' = shiftL'(u, v)$
 5:   if $v' \leq m$ and $v' > 0$ then
 6:     $vM(u, v) \leftarrow vL(u, v')$
 7:   else
 8:     $v' = shiftR'(u, v)$
 9:     if $v' <= m$ and $v' > 0$ then
10:       $vM(u, v) \leftarrow vR(u, v')$
11:     else
12:       $v' = round(v + zInvL(u, v) \times f \times b)$
13:       if $v' <= m$ and $v' > 0$ then
14:         $vM(u, v) \leftarrow vL(u, v')$
15:       else
16:         $v' = round(v - zInvR(u, v) \times f \times b)$
17:         $vM(u, v) \leftarrow vR(u, v')$
18:       end if
19:     end if
20:   end if
21: end for
    {$zInvL$ and $zInvR$ are $\frac{1}{dL'}$ and $\frac{1}{dR'}$ as we are multiply-
    ing it with focal length f and camera base line difference
    b. Compare it with Equation 2}
22: return $vM$: Final intermediate virtual view
```

III. Experimental Evaluation

The proposed VSIM technique has been tested over a number of standard test sequences. VSIM implementation takes as input two views and their depth maps in YUV(4:2:0) format and estimate the intermediate view in the same format. The warping phase works using integer pixel precision, i.e. by rounding to nearest integer all shifted column positions. The chroma components U, V are warped in their native (down-sampled) resolutions as well. The proposed inverse mapping procedure is then used to fill the holes, independently on each of the 3 components.

First of all the VSIM technique has been compared with MPEG VSRS reference software using both integer and quarter pixel precisions. Tab. I show the experimental settings reporting sequence name, rendered view index, video resolution and total number of rendered frames (NF), along with the average Luma PSNR yielded by VSIM and VSRS with integer and quarter pixel precision (see VSRS$_1$ and (VSRS$_4$ columns respectively). It can be observed that VSIM achieves a significant gain over VSRS with integer pixel precision and exhibits quite similar performance compared to VSRS with quarter pixel precision. In other words, VSIM favorably competes with the reference software without requiring either up-sampling or interpolation of the warped views. In order to better appreciate the effect of the proposed inverse mapping procedure an additional set of experiments have been worked out by replacing it with a simple 7 averaging filter (see VSIM$_a$ column in Tab. I). It can be observed that inverse mapping yields a noticeable improvement with respect to pixel interpolation by averaging.

Fig. 3, Fig. 4 and Fig. 5 show the Luma PSNR versus frame number obtained by VSRS and VSIM on Undo_Dancer, Kendo and Cafe sequences, respectively. It can be noted that VSIM yields a major improvement especially in the case of the computer generated Undo_Dancer sequence.

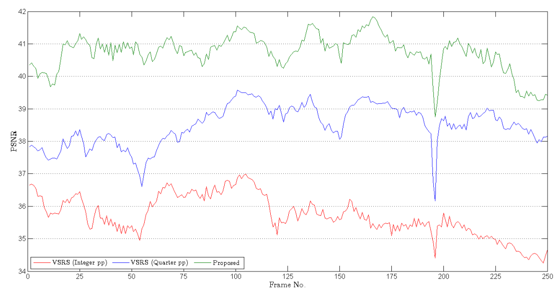


Fig. 3: Undo_Dancer sequence: PSNR comparison of the proposed technique with that of VSRS.

The experiments described above show that the VSIM yields very competitive results in terms of objective PSNR metric. The most important feature of VSIM is represented by the novel hole filling procedure based on interpolation of pixel coordinates through inverse mapping. On the contrary most existing view synthesis techniques use inpainting algorithms to fill the holes; such algorithms fill the holes by either looking for similar texture areas or using the color information in the pixel neighborhood. Such approaches are effective in recovering holes within smooth areas (with limited depth variations) but may fail in region with sharp transitions between foreground and background objects. In this latter situation inverse mapping can be more effective since it avoids smoothing or averaging pixel values in the texture domain but try to recover column shifts towards the corresponding pixels of the original view. Fig. 6 shows a particular frame of the Undo_Dancer sequence where the background in between two pillars (see the zoomed details) is rendered correctly by VSIM whereas VSRS averages foreground and background pixels. The proposed technique on the other hand, maps the missing locations to the respective left or right views and fills the hole without averaging pixels with completely different depths. Other examples of visual details that are rendered better by VSIM than VSRS are shown in Fig. 7 and Fig. 8.

Finally, it is worth analyzing the computational cost of

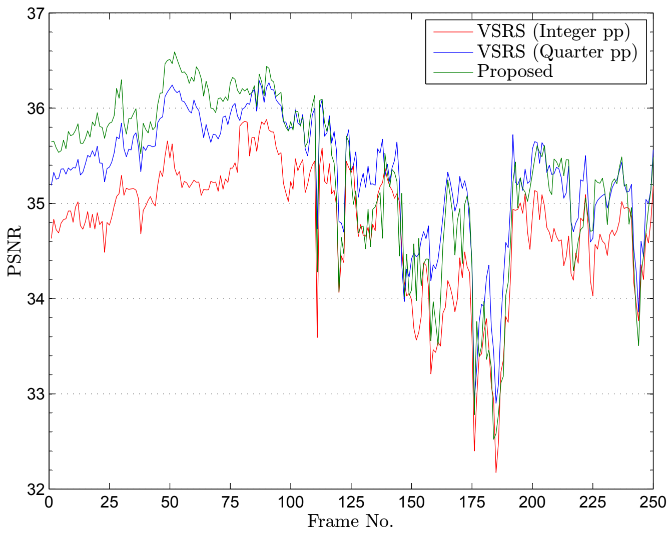


Fig. 4: Kendo sequence: PSNR comparison of the proposed technique with that of VSRS

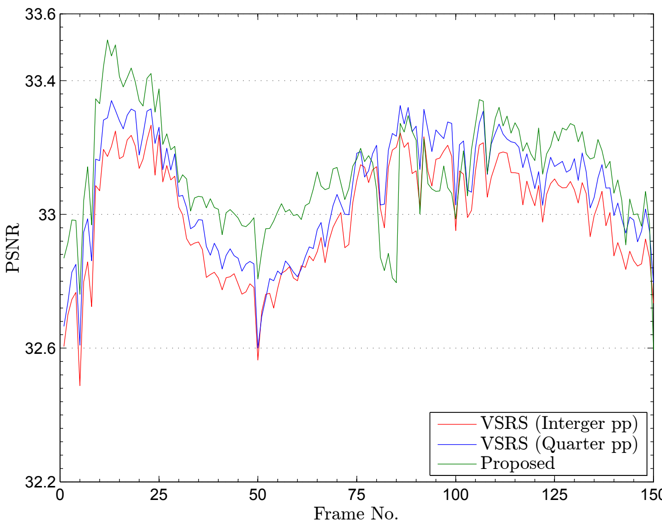
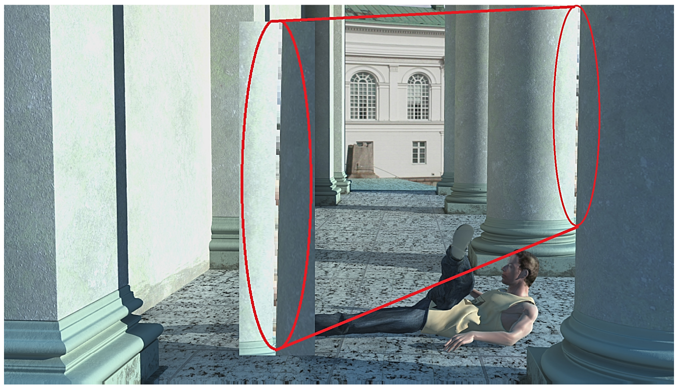


Fig. 5: Cafe sequence: PSNR comparison of the proposed technique with that of VSRS

TABLE I: Experimental settings and results showing sequence name, rendered view index, video resolution and total number of frames (NF) and average Luma PSNR obtained with VSRS integer precision ($VSRS_1$ and quarter pixel precision ($VSRS_4$), VSIM with hole filling by averaging ($VSIM_a$) and proposed VSIM hole filling (VSIM).

Sequence	Views	Size	NF	PSNR				Gain of VSIM over		
				$VSRS_1$	$VSRS_4$	$VSIM_a$	VSIM	$VSRS_1$	$VSRS_4$	$VSIM_a$
Undo_Dancer	$1,3 \to 2$	1920×1088	250	35.7117	38.4899	40.5387	40.7348	5.0231	2.2448	0.1961
Balloons	$1,5 \to 3$	1024×768	250	34.0799	34.5399	33.2804	34.5000	0.4201	-0.0399	1.2196
Poznan_Hall2	$5,7 \to 6$	1920×1088	200	35.6328	36.4990	36.0198	36.3705	0.7377	-0.1285	0.3507
Kendo	$1,5 \to 3$	1024×768	250	34.8026	35.3193	34.3606	35.2419	0.4393	-0.0774	0.8813
Cafe	$2,4 \to 3$	1920×1080	150	32.9975	33.0673	32.7299	33.1265	0.1290	0.0592	0.3966


(a) VSRS

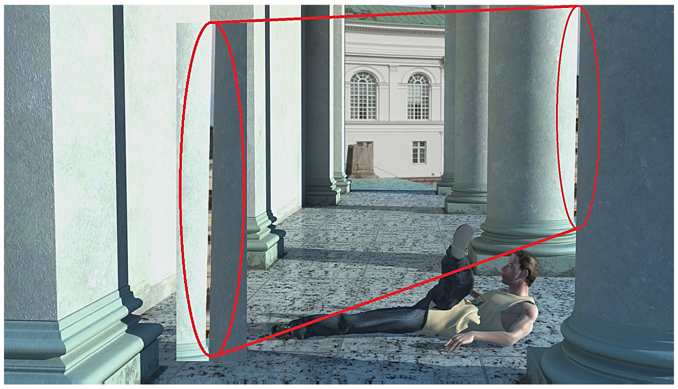
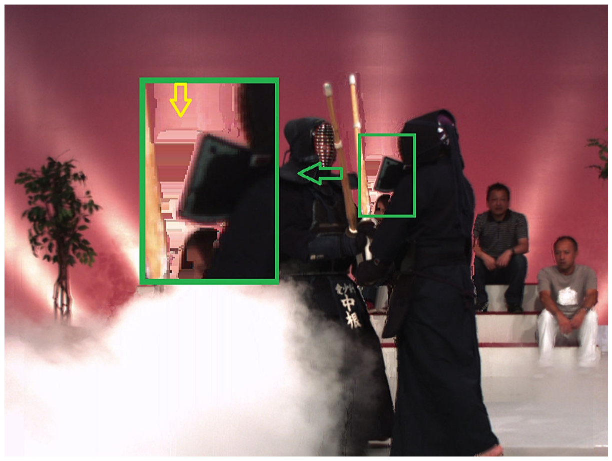

(b) VSIM

Fig. 6: Rendering example for Undo_Dancer sequence: VSRS (a), VSIM (b).

the proposed VSIM with respect to VSRS. VSIM does not require fractional pixel precision and therefore it requires approximately the same computational cost of VSRS with integer pixel precision, that yields significantly lower performance as reported in Tab. I. Moreover, the inverse mapping hole filling algorithm is based on simple median filtering of shift information that is already available after warping and therefore it does not represent an issue in terms of memory and computation. In this paper we cannot fairly compare execution times of the two softwares because VSIM has been implemented using MATLAB whereas VSRS in written in C language. An efficient implementation of VSIM in C language is part of our ongoing work.


(a) VSRS

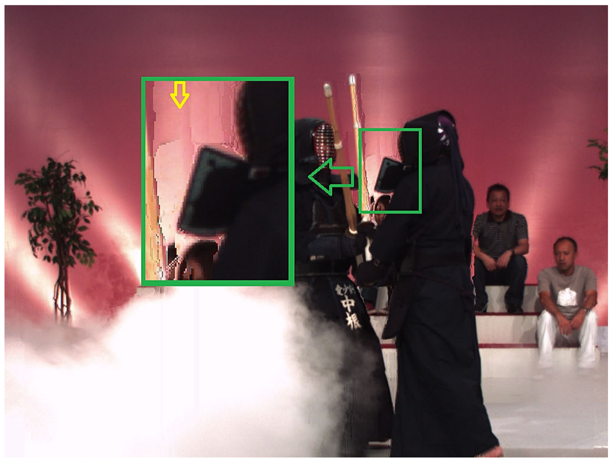
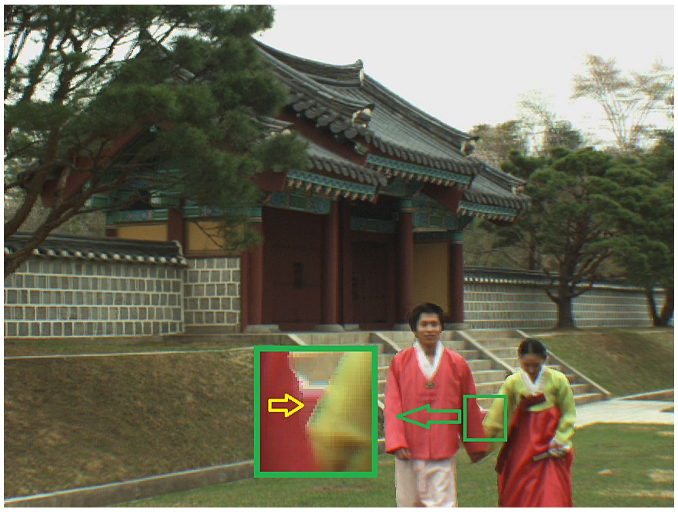

(b) VSIM

Fig. 7: Rendering example for Kendo sequence: VSRS (a), VSIM (b).

IV. Conclusions

In this paper we have presented VSIM, a novel depth image based rendering algorithm that conjugates limited computational complexity and high quality view synthesis results. These conflicting goals have been achieved by using only integer pixel warping on the one hand, and improving the visual quality of the hole filling algorithm on the other hand. Hole filling is based on the novel idea of inverse mapping that


(a) VSRS

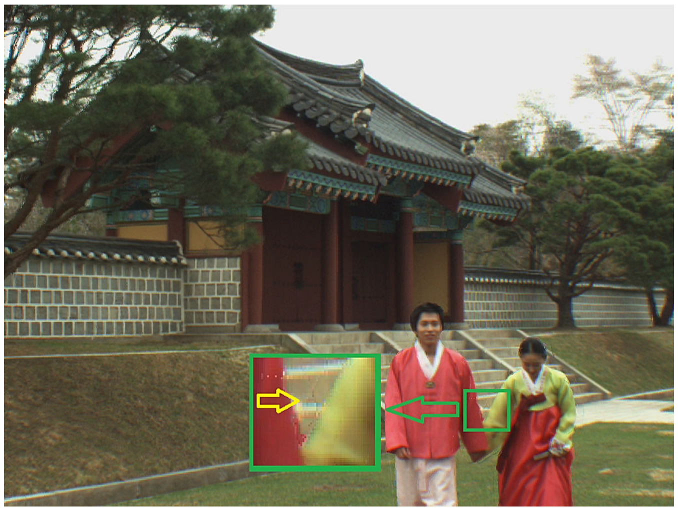

(b) VSIM

Fig. 8: Rendering example for LoveBirds sequence: VSRS (a), VSIM (b).

consists in retrieving the missing pixels from the original views rather than interpolating them from the surrounding neighborhood. The presented experimental evaluation shows that VSIM favorably compares with VSRS in terms of objective and subjective results.

REFERENCES

[1] H. C. Longuet-Higgins, "Readings in computer vision: issues, problems, principles, and paradigms," chapter A computer algorithm for reconstructing a scene from two projections, pp. 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[2] Shenchang Eric Chen and Lance Williams, "View interpolation for image synthesis," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1993, SIGGRAPH '93, pp. 279–288, ACM.

[3] Steven M. Seitz and Charles R. Dyer, "View morphing," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1996, SIGGRAPH '96, pp. 21–30, ACM.

[4] George Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1st edition, 1990.

[5] Heung-Yeung Shum and Sing Bing Kang, "A review of image-based rendering techniques," in *IEEE/SPIE Visual Communications and Image Processing (VCIP)*, 2000.

[6] Christoph Fehn, "A 3d-tv approach using depth-image-based rendering (dibr)," in *Proc. of VIIP*, 2003, vol. 3.

[7] Christoph 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, 93 (May 21, 2004)*, April 2004.

[8] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen, "The lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1996, SIGGRAPH '96, pp. 43–54, ACM.

[9] C. Zhang, "A survey on image-based renderingrepresentation, sampling and compression," *Signal Processing: Image Communication*, vol. 19, no. 1, pp. 1–28, Jan. 2004.

[10] Heung-Yeung Shum, Sing Bing Kang, and Shing-Chow Chan, "Survey of image-based representations and compression techniques," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 11, pp. 1020–1037, 2003.

[11] Ha T. Nguyen and Minh N. Do, "Image-based rendering with depth information using the propagation algorithm," in *in Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc*, 2005, pp. 589–592.

[12] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand, "Depth image-based rendering with advanced texture synthesis for 3-d video," *Trans. Multi.*, vol. 13, no. 3, pp. 453–465, June 2011.

[13] Christian Hofsetz, Kim Ng, George Chen, Peter McGuinness, Nelson Max, and Yang Liu, "Image-based rendering of range data with estimated depth uncertainty," *IEEE Comput. Graph. Appl.*, vol. 24, no. 4, pp. 34–42, July 2004.

[14] M. Schmeing and Xiaoyi Jiang, "Depth image based rendering: A faithful approach for the disocclusion problem," in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, June, pp. 1–4.

[15] Kwan-Jung Oh, Sehoon Yea, and Yo-Sung Ho, "Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-d video," in *Picture Coding Symposium, 2009. PCS 2009*, May, pp. 1–4.

[16] M. Koppel, Xi Wang, D. Doshkov, T. Wiegand, and P. Ndjiki-Nya, "Consistent spatio-temporal filling of disocclusions in the multiview-video-plus-depth format," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*, Sept., pp. 25–30.

[17] V. Paradiso, M. Lucenteforte, and M. Grangetto, "A novel interpolation method for 3d view synthesis," in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2012*, Oct., pp. 1–4.

[18] M.S. Farid, H. Khan, and A. Mahmood, "Image inpainting based on pyramids," in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, Oct., pp. 711–715.

[19] M.S. Farid and H. Khan, "Image inpainting using dynamic weighted kernels," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, July, vol. 8, pp. 252–255.

[20] 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, 2011.

[21] ISO/IEC JTC1/SC29/WG11 (MPEG), "View synthesis reference software (vsrs) 3.5," Mar. 2010.