A Parallel Depth-aided Exemplar-based Inpainting for Real-time View Synthesis on GPU

Zheng Tian, Cheng Xu
College of Information Science and Engineering
Hunan University
Changsha, China
tianzheng@hnu.edu.cn, cheng_xu@yeah.net

Xiaoyun Deng
R&D, STMicroelectronics Asia Pacific Pte Ltd

Singapore
xiaoyun.deng@st.com

***Abstract*—Synthesizing new images from given image pair and their corresponding depth maps is an essential function for many 3D video applications. Exemplar-based inpainting methods have been proposed in recent years to be used to restore newly synthesized images by strategically filling the missing pixels which don't have any references due to occlusion. Due to the prioritized filling process, the inpainting methods usually result in high computational complexity and can hardly reach real-time performance. In this paper, a parallel depth-aided inpainting method is proposed to address the efficiency issue of this kind of high performance algorithms. In order to reduce the computation, the proposed method searches for background pixels in a restricted search range on the reference images for effective context filling. Then a partially parallel strategy is proposed to speedup the inpainting process while maintaining its high restoration accuracy. Finally the method is implemented with CUDA on NVidia graphic card GTS450. The experiment results showed that the proposed method could produce the best on par results and is suitable for real-time multi-view image synthesis.**

Keywords- View Synthesis; Inpainting; Search Range; Partially Parallel; CUDA

I. INTRODUCTION

View synthesis is a process of generating virtual viewpoint images according to multiple real images which are captured from different viewpoints. It is a key technique in many 3D video applications such as free-viewpoint television, multi-view video compression and eye contact correction for people communicating using webcam.

Depth-Image-Based Rendering (DIBR) [1] is currently the most popular technique for view synthesis due to the easy availability of the depth information. In general, DIBR is composed of three main steps: pixel mapping, pixel merging, and hole-filling. Pixel mapping step maps pixels in the reference image to the target image by a 3D warping formula [1]. However, due to occlusion and discontinuities of depth map, some pixels on the target image cannot find their corresponding ones from the reference image and holes appear. The merging step blends the multiple target images together to partly eliminate the holes, but large hole regions may still exist especially due to dramatic change of the viewing angle. Hole-filling step restores the merged target image by filling the missing pixels and makes the final picture visually appealing.

In the above three steps of DIBR, the mapping and merging step are pixel-based process based on equations and there is little data dependency among pixels, so these two steps are straight-forward and could be easily speeded up by parallel computing. The third step for filling the missing pixels, however, may require smart algorithms of certain complexity to fulfill the performance requirements. Hole-filling methods such as horizontal interpolation [2] and multidirectional extrapolation [3] for example, are effective in small cracks, but may wrongly fuse the foreground and background in large holes and create obvious artifacts. In order to make the filling more visually appealing, a new category of algorithms based on image inpainting has been proposed. These algorithms [4-7] are based on the original exemplar-based inpainting method proposed by A. Criminisi et al [8], which propagates both the texture and structures from the neighboring reference pixels to effectively fill the missing pixels in structure-prioritized order. They further added the depth information obtained in the view synthesis application for confining the search range to only background pixels in the synthesized image. This category of methods produces good quality results, but is expensive to implement due to the sequential process for the prioritized filling and the exhaustive searching. Thus they are not really suitable for real-time applications.

In this paper, we propose a parallel depth-aided exemplar-based inpainting method that is efficient and can produce good subjective results in the synthesized images or video. Differing from the existing sequential inpainting methods, the presented solution searches for the best matching pixels in a pre-calculated window of the reference images for the most effective search and reduced computational cost. Moreover, a partially parallel strategy is proposed to accelerate the sequential inpainting process, so it can be used for real-time or near real-time video applications. Finally, CUDA technique is adopted to implement the whole process on GTS450.

The rest of the paper is organized as follows. Section II describes the main idea of the proposed depth-aided exemplar-based inpainting method, the parallel optimization and the GPU implementation. Section III shows the experimental results. Finally, a brief conclusion is given in section IV.

II. PROPOSED METHOD

In the process of view synthesis, there are two kinds of holes, the 'cracks' and the 'cavities', namely. The 'cracks' are of relatively small size on the surface of an object, which is usually caused by insufficient precision of depth values that is lost during the pixel mapping process. This kind of 'cracks' could be easily filled by pixel interpolation due to their small size. The 'cavities', unlike 'cracks', always happen on the background pixels between two foreground objects due to occlusion. The size of 'cavities' could be very large depending on the virtual viewing position in view synthesis. In this paper, we focus on methods for filling the 'cavities'.

A. Defining Search Range

In exemplar-based inpainting algorithms [8], the 'inpainting' problem is described as how to fill the target region Ω with the information from the source region Φ. In the view synthesis problem, the available source region includes the two or more reference images instead of only the source region Φ of the target image.

The state-of-art depth-aided inpainting methods for view synthesis always search the whole target image and require iterative filling, which is very time-consuming. On the other hand, the simplified hole-filling methods only search on the neighboring pixels on the target image based on the assumption that the color does not change in adjacent pixels, which is often not true in the view synthesis application. For example, in figure 1(b), there are only foreground pixels but no background ones around the hole regions after the initial stages of view synthesis. In another approach, reference images are used for the searching [9], which could avoid searching on foreground pixels with existing depth information. But the challenge of searching on reference images is how to pre-define an effective search range to avoid redundant computational cost while maximizing the probability of having relevant context included in the range.

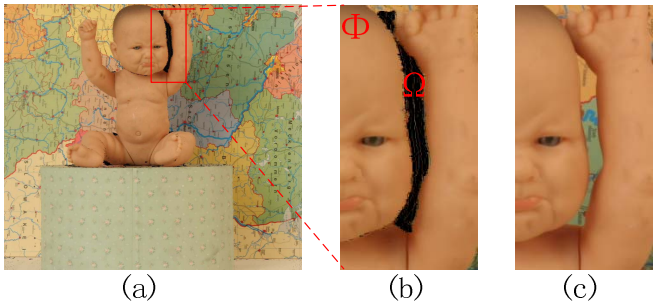

Figure 1. Illustration of target region and reference region: (a) target image after pixel mapping and merging. (b) hole region on target image. (c) source region on one of the reference images.

To address these problems, we propose a method to define a reduced but effective search range in the reference images. The reduced window size shall significantly reduce the computational cost and could provide more complete structure and texture information than the target image. An example search window that is desired in the reference image is indicated in figure 1(c). We could further limit the search on background pixels based on the provided depth information.

To define an effective search range, we start by analyzing the cause of holes or 'cavities'. As shown in figure 2, the common part of the occluded area on the left and right view forms the hole region in the synthesized image. In order to fill the hole with the most relevant background pixels, a search in Region 1 (highlighted in blue) and 2 (in green) is desired.

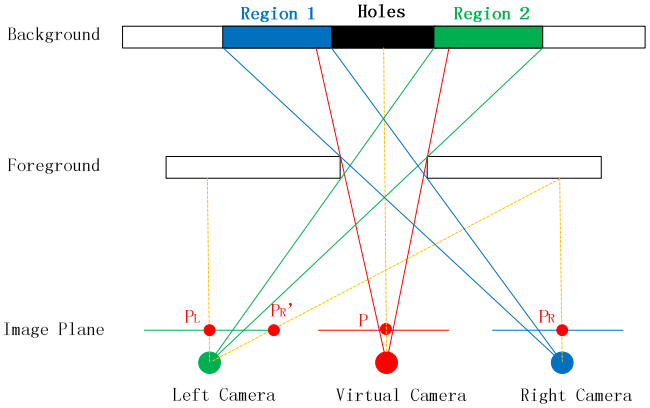

Figure 2. Generation of holes caused by occlusion and illustration of feasible search range

To encompass Region 1 and 2 as shown in figure 2, let's say pixel P is a hole pixel that we want to fill, and we define P_L and P_R as the co-located pixels in the left and right reference image, respectively, such that P_L and P_R have the same pixel coordinates with P. $P_{R'}$ is the corresponding pixel of P_R on the left reference image. Figure 3 illustrates the relationship among these points in an intuitive way.

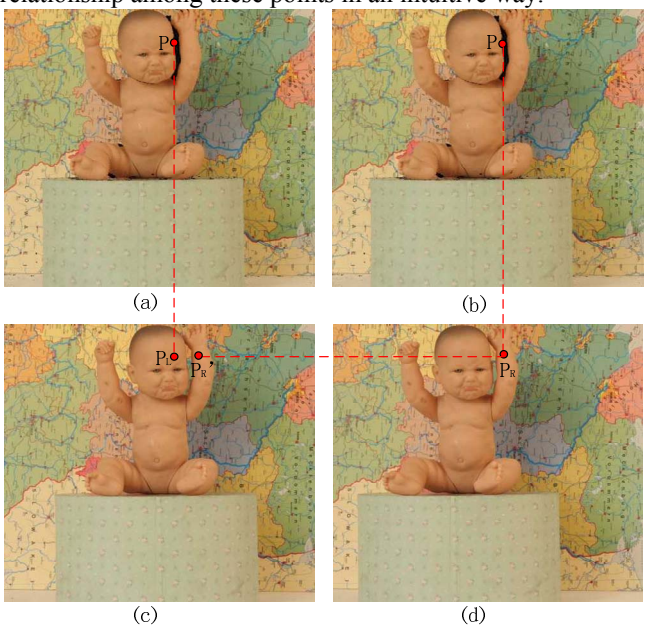

Figure 3. Example of search range determination: (a) and (b) the same target image. (c) left reference image. (d) right reference image

Assuming the reference images are stereo left and right images, and the synthesized view is merely by horizontal translation, the x-coordinates of these 4 pixels should satisfy the following equations, wherein d_{P_R} is the disparity value of point P_R.

$$x_P = x_{P_L} = x_{P_R} \quad (1)$$

$$x_{P_R'} = x_{P_R} + d_{P_R} \quad (2)$$

We can see that Region 2 will be contained in the region between P_L and $P_{R'}$ when the hole is caused by the foreground and background relationship depicted in figure 2. The corresponding search range on the right reference image to contain Region 1 could be determined in a similar way.

B. *Partially Parallel Inpainting*

With the search range defined, the next step is to find the best matching pixel to fill the missing parts of the image. In the original exemplar-based inpainting algorithm [8], a 9x9 support window was built for each pixel to calculate the sum of squared differences (SSD) between the support window of the target pixel and that of the reference pixels. Then the hole pixel h is filled with the reference pixel p in the search range R with the minimal matching cost as in equation (3). We proposed to use a similar matching algorithm for the best in class performance but optimized the method to achieve a good balance between performance and efficiency.

$$h = \arg\min_{p \in R} MC(h, p) \quad (3)$$

There are two main constrains that we should consider when we optimize the inpainting process. Firstly, in the matching cost calculation, we should make sure that there is certain number of source pixels in the support window of target side. Secondly, as in exemplar-based inpainting, the filling process should be in sequential and be prioritized so that the structural information could be correctly propagated to the hole region. With these two requirements bearing in mind, we partially parallelized the method for GPU acceleration with little degradation in performance.

We first divide the target image into many non-overlapping target windows of size 8x9 for example. Inside each target window, we parallelize the matching cost calculation but fill the missing part column-by-column in order to meet the requirement to prioritize the filling process. Among the target windows, parallel filling process is allowed except for those windows without any existing reference pixels. Target windows near the boundaries of holes are hence filled first in parallel, and those far away from the reference pixels will be reserved for the next loop. This ensures that the requirement to have reference pixels in the support window is also met in order to propagate the structural information.

In such strategy, the worst case of processing time happens when the hole to be filled is of maximally possible size. In fact, given a scenario with known foreground and background depth Z_f and Z_b respectively, and the camera parameters such as the baseline T between the two cameras, one could derive that the worst case happens when the viewing position is in the middle of the two cameras, as shown in figure 4. According to the geometrical relationship, the hole size could be calculated using equation (4), and the gap between the two foreground objects G is of a value as expressed in equation (5). As the depth could be represented by the disparity value in the reference images with equation (6), one could further derive the maximally possible hole size H_{max} to be as equation (7), where D_f and D_b is the disparity value of the foreground and background object, respectively. Since in our implementation each target window contains 8x9 pixels, the maximum literation numbers in the worst case could by approximately expressed in equation (8). For all the test images in this paper, the number of iterations didn't exceed 3.

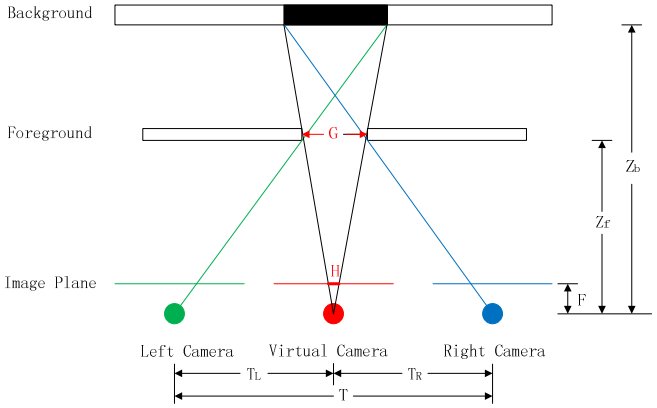

Figure 4. The case with the maximum hole size

$$H = \frac{F \bullet G}{Z_f} \quad (4)$$

$$G = \frac{T}{2} \bullet \frac{(Z_b - Z_f)}{Z_b} \quad (5)$$

$$Z = \frac{F \bullet T}{D} \quad (6)$$

$$H_{\max} = \frac{1}{2} \bullet (D_f - D_b) \quad (7)$$

$$Iter_{\max} = \frac{1}{16} \bullet (D_{\max} - D_{\min}) \quad (8)$$

C. *Detailed CUDA Implementation*

For each target window, we use a 16x16 CUDA thread block and the on-chip shared memory to speed up the matching cost calculation according to the architecture of the GPU. As shown in figure 5, three shared memory array A, B and C are created for each target window to cache the relevant data which are frequently accessed. A is used to store the target window of size 8x9 with the neighboring support pixels, resulting in a window of 16x9. B is of size 9x64 to store the reference pixels in the search range. The temporary matching cost between each column of A and B can be calculated in parallel and fill the array C, which is of size 64x16. For instance, the matching cost between column i in A and column j in B is stored in $C(j,i)$. Each thread needs

to calculate 4 values in C. The advantage of having the shared memory C is to avoid repetitive computation for the intermediate results, and hence greatly saves computational time. Once C is ready, we then need to sum 9 elements in C corresponding to 9 columns of support pixels to get the matching cost for one target column, as shown in equation (9). The best matching column j_{best} for each target column i is thus obtained as in equation (10). The intermediate results in row i of C is then updated at one shot to prepare for the filling of the next column i+1.

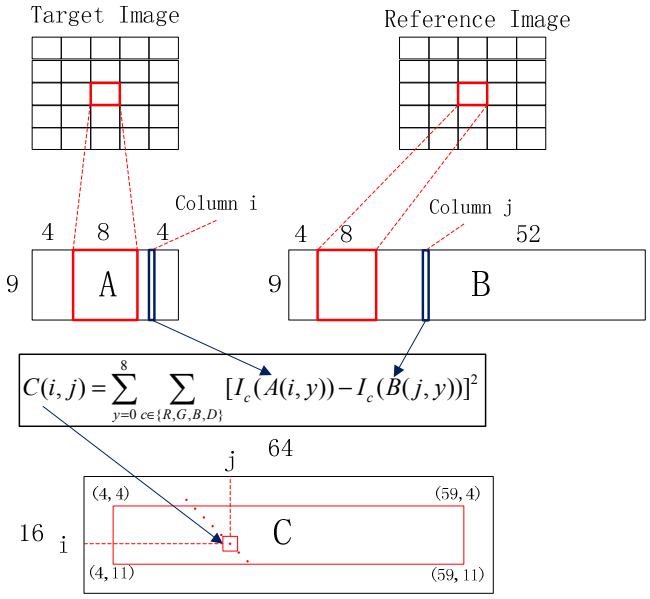

Figure 5. Illustration of shared memory optimization

$$MC(i,j)=\sum_{n=-4}^{4}C(j+n,i+n),\ for\ i\in[4,11],\ j\in[4,59] \quad (9)$$

$$j_{best}=\arg\min_{j\in[i,59]}MC(i,j),\ for\ i\in[4,11] \quad (10)$$

The following pseudo code illustrates the main procedures described above.

```
dim3 D_b = dim3(16, 16);    //define block dimensions
dim3 D_g = dim3( ceil(width/8), ceil(height/9) ); //block image
while(  there are still holes ) {
    // many blocks at one shot
    cuda_kernel_hole_filling<<<D_g,D_b>>>();
}
cuda_kernel_hole_filling {
    load_shared_mem(A,B);                    // parallel
    MC(i,j) = calc_matching_cost(A,B,C);     // parallel
    for( i=4:11 ) {              // sequential filling
        search j_best for i;     // sequential searching
        update row i of C;       // parallel
    }
}
```

III. EXPERIMENTS

We compared the proposed method with three other image restoration methods, they are: improved parallel horizontal interpolation based on literature [1], original exemplar-based inpainting, whose source code could be downloaded from [10], and a simplified depth-aided inpainting based on literature [4]. The testing images with corresponding depth maps are provided from the Middlebury stereo dataset [11]. All the algorithms are evaluated on a personal computer equipped with AMD Quad-Core 2.8GHz CPU with 8GB RAM, and NVidia GeForce GTS450 GPU with 1GB RAM. All these methods are implemented with CUDA, except for the original inpainting method which uses MATLAB.

We synthesized the images in the middle position of two real cameras, and some results are shown in figure 6. The original left reference image and enlarged ground truth region are given in figure 6(a) and 6(b). Large holes may appear in the synthesized image between the two foreground objects, as shown in figure 6(c). The depth-aided interpolation method in figure 6(d) produces noticeable artifacts such as wrong filled color and artificial line patterns in the regions, especially when the color in both sides of the holes is different. Meanwhile, the original inpainting method sometimes propagates wrong texture information into the holes as in figure 6(e), because it doesn't utilize the depth information and the foreground pixels are allowed to fill the holes. The simplified depth-aided inpainting in figure 6(f) is better than the former two methods but is still not perfect perhaps due to the limitation of searching only on the target image, especially when there is complex texture on the background. Comparatively, the proposed method as in figure 6(g) produces very good results with the texture and structure well propagated into the hole regions.

TABLE I. EXECUTION TIME OF RELATED METHODS / MS

	Baby	Lampshade	Flowerpots
Resolution	620x555	650x555	656x555
Hole Portion	0.167%	1.17%	4.15%
Interpolation	15.8	16.2	16.5
Original Inpainting	123672	381758	924263
Depth-aided Inpainting	8190	11045	28143
Proposed	42.5	65.2	67.5

Table 1 shows the execution time of the proposed inpainting method compared to the other sequential processing methods. The results are the average value after running the respective method 20 times. The results show that the proposed method can reach real-time or near real-time performance. It is understandable that the original inpainting running on MATLAB is the most time-consuming, while the rest are implemented in the same platform and constitute a more fair comparison. It is observed that the proposed method has the computation time comparable to or at the same scale as the simplest interpolation method, while it is able to preserve the structural information and provide much better hole filling results. Compared to the depth-aided

inpainting, the proposed method could speed up near 200 times, thanks to the proposed strategy for optimal utilization of parallelism and reduction of computation as explained in the previous sections.

IV. CONCLUSION

This paper discusses a new depth-aided exemplar-based parallel inpainting method, which can be applied to a real-time view synthesis system. Firstly we limit the search range in the reference image for an efficient search for the best matching results. Then a partially parallel strategy is presented to speed up the inpainting process, which is implemented with CUDA. Experimental results showed that the proposed method is greatly more efficient than the state-of-art image restoration methods without compromise in performance.

Currently, this method is not fit for synthesizing the virtual views outside of the two cameras, where the virtual viewing position is more left than the left camera or more right than the right camera. Our future work is to extend the method to support this scenario.

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for the Central Universities (No.531107040421) and Hunan Provincial Innovation Foundation for Postgraduate (CX2011B138). The authors are particularly grateful to the support from the R&D Centre of STMicroelectronics Asia Pacific Pte Ltd.

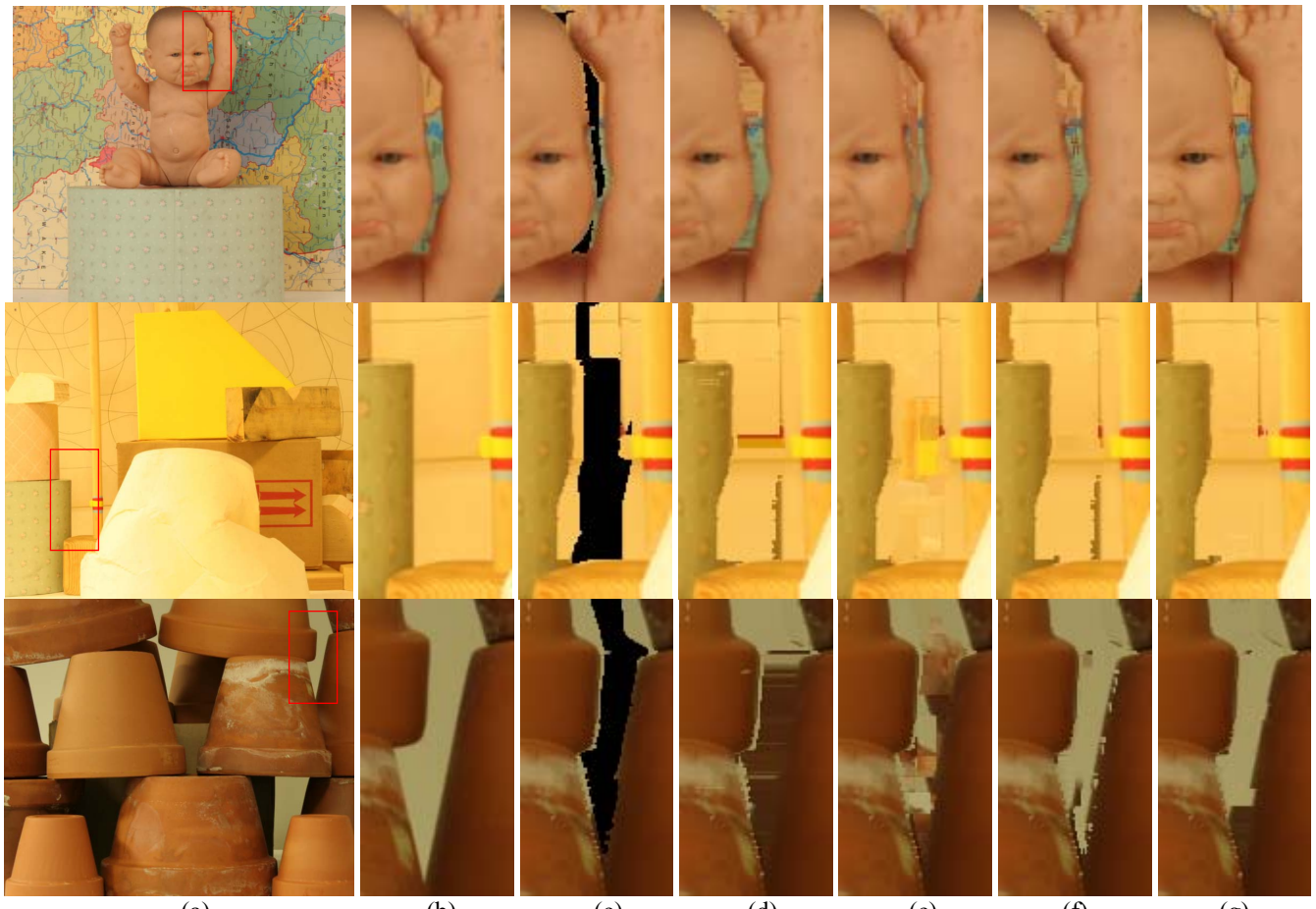

Figure 6. the subjective hole-filling results of (from top to bottom) 'Baby', 'Lampshade', and 'Flowerpots' sequences: (a) left reference image, (b) ground truth, (c) hole regions, (d) result of interpolation, (e) result of original inpainting, (f) result of depth-aided inpainting, (g) result of proposed method.

REFERENCES

[1] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "View Generation with 3D Warping Using Depth Information for FTV," Signal Processing: Image Communication, vol. 24, no.1-2, pp.65-72, Jan. 2009.

[2] C. Vazquez, W. J. Tam and F. Speranza, "Sterescopic Imaging: Filling Disoccluded Areas in Depth Image-Based Rendering," Proc. Of SPIE, Vol. 6392, 2006.

[3] S.H. Zhang, X.Y. Xu, and Y.S. Zhu, "A new multidirectional extrapolation hole-filling method for Depth-Image-Based Rendering," Proc. of 18th IEEE International Conference on Image Processing, pp.2589-2592, 2011.

[4] K. Luo, D.X. Li, Y.M. Feng, and M. Zhang, "Depth-Aided Inpainting for Disocclusion Restoration of Multi-View Images Using Depth-Image-Based Rendering," Jounal of Zhejiang University Science A, vol.10, pp.1738-1749, Dec. 2009.

[5] Y.S. Kim, S. Lee, O. Choi, and J.D.K. Kim, "Bi-layer inpainting for novel view synthesis," Proc. of 18th IEEE International Conference on Image Processing, pp.1089-1092, 2011.

[6] S. Choi, B. Ham, and K. Sohn, "Hole filling with random walks using occlusion constraints in view synthesis," Proc. of 18th IEEE International Conference on Image Processing, pp.1965-1968, 2011.

[7] C.M. Cheng, S.J. Lin, and S.H. Lai, "Spatio-Temporally Consistent Novel View Synthesis Algorithm From Video-Plus-Depth Sequences for Autostereoscopic Displays," IEEE Transactions on Broadcasting, Vol. 57, No. 2, pp. 523-532, June 2011.

[8] A. Criminisi, P. Pérez, and K. Toyama, "Object removal by exemplar-based inpainting," Proc. of Int. Conf. Comput. Vis. Pattern Recog., vol. 2, no. 2, pp. 721–728, Jun 2003.

[9] S. H. Bae, A. Berestov, F. Baqai, et al. Method and Apparatus for Multiview Image Generation Using Depth Map Information. US Patent, US 2012/0162193 A1, Jun. 28, 2012

[10] http://www.cc.gatech.edu/~sooraj/inpainting/

[11] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1-8, June 2007.