# GPU Parallel Programming for Real-time Stereoscopic Video Generation

In-Yong Shin and Yo-Sung Ho
Gwangju institute of Science and Technology (GIST)
261 Cheomdan-gwagiro, Buk-gu, Gwangju, 500-712, South Korea
{siy0808, hoyo}@gist.ac.kr

**Abstract** - ***In this paper, we propose a fast depth image based rendering method to generate a virtual view image in real-time using a graphic processor unit (GPU) for a 3D broadcasting system. Simple and efficient hole filling method is performed to reduce complexity and prevent hole filling error. Also, we designed a vertical parallel structure for a forward mapping process to take advantage of the single instruction multiple threads structure of GPU. Additionally, we utilize high speed GPU memories to boost computation speed. As a result, we can generate virtual view images 15 times faster than a CPU-based processing.***

## I. Introduction

Recently, various researches have been on 3D broadcasting system as increasing interest in 3D multimedia service. 3D broadcasting system provides realistic multimedia service that offers a 3D effect based on binocular depth cue.

Fundamentally, in order to offer clients 3D experience, real camera images captured from more than two viewpoints should be given to a 3D display device. The most representative way of 3D broadcasting system is stereo video broadcasting system that gives only two viewpoints. Stereo viewing is a common technique to increase visual realism or enhance user interaction with 3D scenes. Two views of a scene are needed, one for the left eye, one for the right. Thus, the stereo video system requires double transmission bandwidth compare to traditional broadcasting system. There are several solutions for this problem. Multi-view video coding (MVC) can reduce amount of redundant data by using inter-view statistical dependencies. Since all cameras capture the same scene from different viewpoints. There is another way of solving this problem. Send one color video for the left eye and corresponding depth map at a transmitting station. Then, virtual viewpoint video for the right eye can be generated by using depth image based rendering (DIBR) in the receiving part. This method can reduce amount of transmission data due to simplicity of depth map. A great part of depth map consists of smooth areas. Additionally, it is single channel image. Conversely, DIBR part has high computation complexity. Thus, it is hard to execute in real-time at the receiving part.

In this paper, we focused on second method which use DIBR algorithm. In order to implement massively large computational algorithm in real-time, algorithms based on graphics processing units (GPUs) have recently attracted a lot of interest in several fields. So, we propose a fast stereo video generation technique by using GPU. Compute Unified Device Architecture (CUDA) is a parallel computing architecture developed by NVIDIA cooperation. CUDA is the general purpose computing engine in NVIDIA GPUs that is accessible to software developers through industry standard programming languages.

This paper organized as follows. In Section II, the stereo video broadcasting system is explained. In Section III, our proposed method is explained. In Section IV, the experimental results are given. The conclusion is presented in Section V.

## II. Stereo Video Broadcasting System

In general, stereo video display device needs two viewpoint video sequences of the same scene. Thus, we need to send these two sequences through a transmission channel. However, if we send two video sequences, this doubles the bandwidth increase in Fig. 1 as shown below.

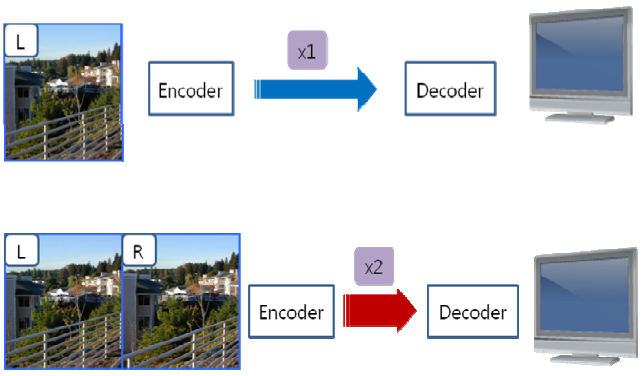


**Figure 1:** Bandwidth Comparison of mono and stereo video.

Some methods are proposed to reduce this bandwidth. In this paper we use one of these methods which utilize a depth map. There are three parts of this method. First, preparing left viewpoint images and corresponding depth maps. A depth map provides a clue to make right viewpoint image.

Additionally, the depth map can be compressed with high compression ratio due to its simple texture. Second, encoding left viewpoint images and corresponding depth map. After then, transmit its compressed data. Last, decoding received data and making right viewpoint image by using depth map. In the last part, we use DIBR algorithm which can make right viewpoint image what we want based on depth information of left color image. By this means, as shown in Fig. 2, we can achieve about 1.2 times bandwidth comparing mono video broadcasting bandwidth while still generating stereo video in the receiving part. However, DIBR algorithm is very heavy at a real-time point of view. Thus, it is important to implement DIBR algorithm in real-time.

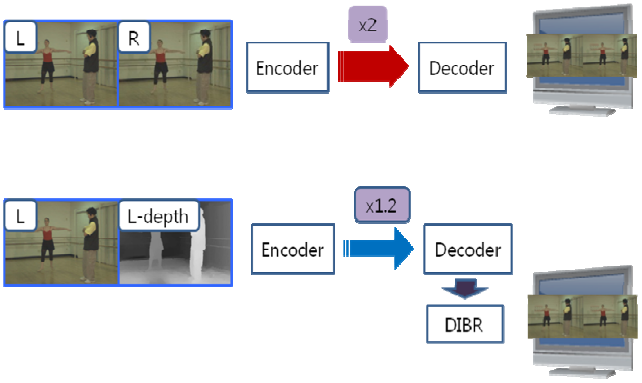


**Figure 2:** Bandwidth Comparison of traditional method and application of transmitting left viewpoint image and corresponding depth map.

### A. Procedures of the DIBR Method

For the right viewpoint image generation based on a depth map, we use depth image based warping algorithm as shown in Fig. 3 [3]. DIBR is the process of synthesizing virtual views of a scene from color image and corresponding depth map. Conceptually, this view generation can be understood as the following process. At first, the original image points are re-projected into the 3D world, utilizing the relevant depth data. After then, these 3D space points are projected into the image plane of a virtual viewpoint camera, which is located at the wanted viewing position. The consecutive operation of re-projection and subsequent projection is generally called 3D image warping in the computer graphics. However, This approach requires drastic computational costs. Thus, instead of using projection and re-projection homography matrix is commonly used to transform one plane into another one [4]. In the end of DIBR algorithm filling holes which are appeared due to disocclusion region. It is important part of DIBR algorithm, because it determines quality of virtual viewpoint image. So, we proposed a simple and efficient hole filling algorithm to achieve parallelized fast implementation.

### B. Homography Matrix

In order to alleviate complexity of projection and re-projection, a simple view transformation described by homography matrix is used to transform one plane into another one instead of using projection and re-projection. Each depth plane in left view has to acquire its own homography matrix. Generally, depth map has 256 possible levels of depth which we have to calculate 256 homography matrices before warping process.

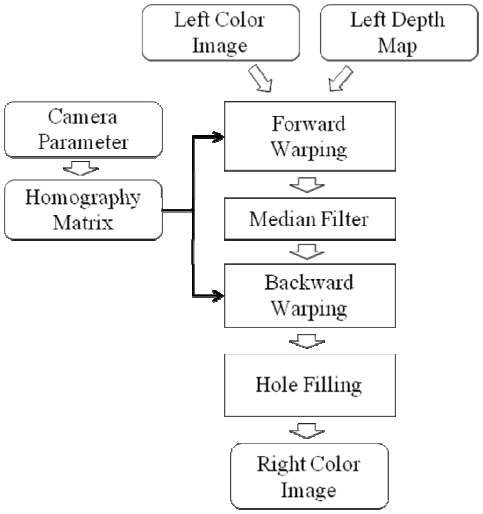


**Figure 3:** Block diagram of depth image based rendering algorithm.

### C. Warping Process

If we warp a left color image toward right viewpoint, there will be numerous small holes on the warped color image because of spatially sampled image data. There are three steps to prevent this problem. First, forward depth map warping is conducted. Second, depth image is filtered by using median filter. Third, backward color image warping is performed. After then, we can get an image as shown in Fig. 4 (b). There are some hole regions that are absent in left view, such as occluded regions, in the result of warping process.

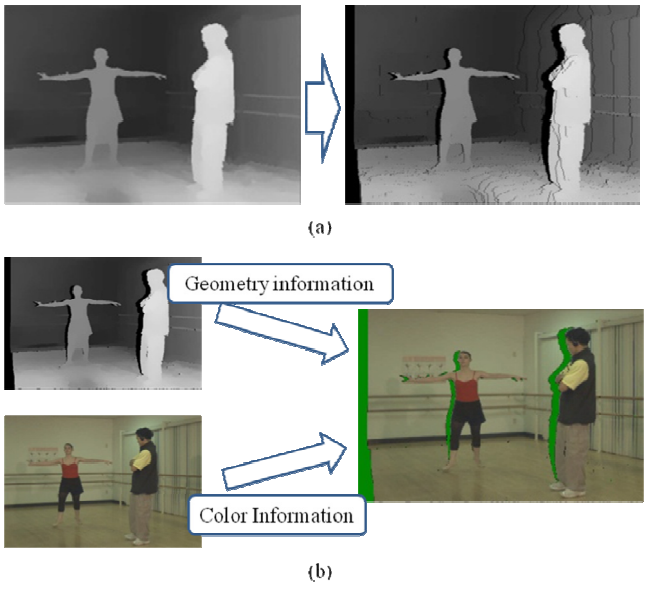


**Figure 4:** (a) Forward warping result (b) Backward warping result.

### D. Fast Hole Filling Method

Although warping process fills up proper pixel values in the right viewpoint image, there are still unknown hole regions which cannot find a same fetch from the left color image due to occlusion problem. Thus, we have to find the most plausible value by using surrounding pixel information. Most of the presented hole filling methods use image interpolation or in-painting algorithm. In order to get best quality hole filled image, neighboring background pixel values and their geometric information are used. The reason why we use background region information is that background pixels rather than the foreground ones as the disoccluded area is more reasonable by definition of the diocclusion [5].

In this paper, we focused on a real-time implementation. So, we have to use simple and efficient hole filling method. In the past research, directional interpolation method is used with reasonable assumption that hole regions belong to background not to foreground objects [6]. Figure 5 shows how hole regions are filled with neighboring background depth values.

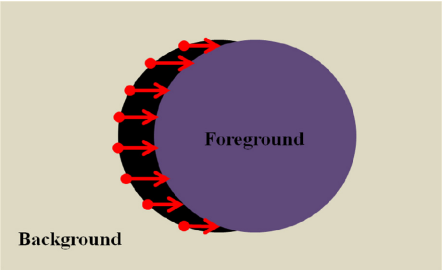


**Figure 5:** Directional image interpolation method.

## III. Proposed Algorithm

In many cases, directional image interpolation produces reasonable interpolated depth map. However, there is an erroneous interpolated region in Fig. 6. The shape of foreground object is changed due to the fact that interpolation was conducted by using foreground depth value.

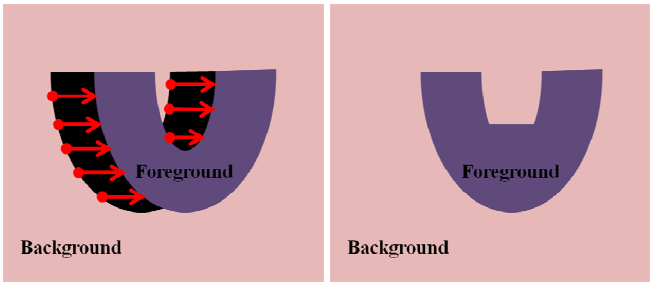


**Figure 6:** (a) Warped color image with hole region (b) Result of traditional directional interpolation method.

### A. Proposed Hole Filling Method

Generally, a width of hole region is determined by disparity difference between left end position and right end position in the stereo image warping. Also, if the width of hole region is the same with disparity difference between left end point and right end point, two end points of hole are adjacent point in the reference view depth map. Thus, in this case, traditional directional interpolation produce valid interpolated depth map. However, if the width of hole region is not the same with disparity difference between two end points of hole region. Thus, we have to use different interpolation method to avoid error. Figure 7 shows proposed hole filling algorithm. If a disparity difference between two end point of hole region is larger than horizontal length of hole, we take pixel value from the reference image point which is positioned with foreground disparity for each hole position. To facilitate the understanding, Fig. 8 shows proposed algorithm graphically. In the result of proposed method, there are not foreground shape change.

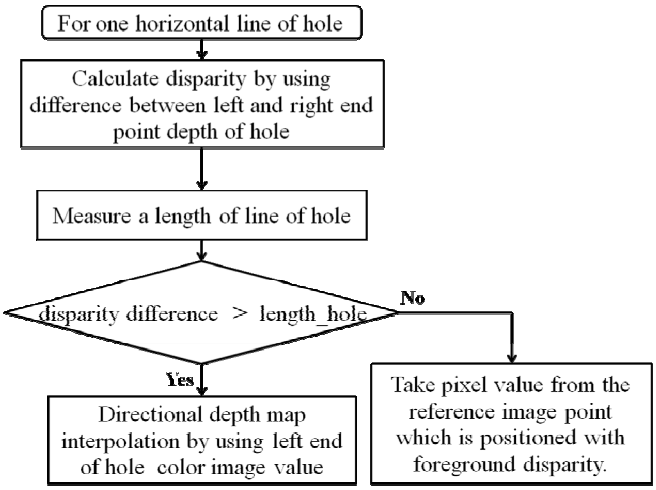


**Figure 7:** Block diagram of proposed hole filling algorithm.

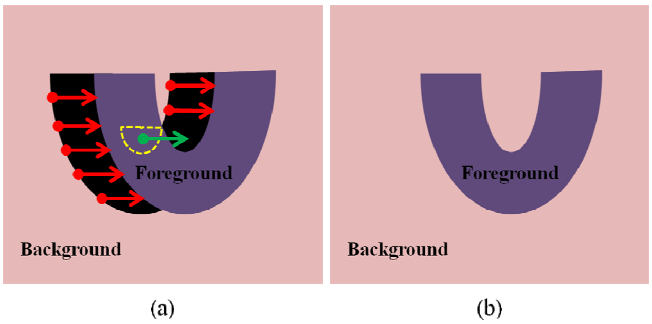


**Figure 8:** (a) Proposed interpolation method (b) Result of proposed method.

### B. GPU Parallel Programming

In order to achieve real-time implementation, we use parallel programming which executed on the GPU called CUDA. The architecture of CPU and GPU are very different. GPU has a ;lot of cores capable of calculating floating point operation. Conversely, GPU has a small number of instruction control unit. Thus, GPU has a Single Instruction Multiple Threads (SIMT) structure [7]. So, DIBR algorithm is very

suitable for GPU programming due to that all of image pixels are determined with same operation in the DIBR algorithm excepting forward warping process. There is a important condition of SIMT parallel processing. It is data independency between all data executed simultaneously. If we use rectangle parallel structure for the forward warping, there will be simultaneously overlapped data. In this case, we cannot know that which data will be remained. To remove a dependency of forward warping process, we determined vertical parallel structure as shown in Fig 9. Virtual right viewpoint depth image can be forward warped by allowing vertical parallel structure. We can use more than one vertical structure, if the space between two structures is less than maximum disparity. The reason why we use vertical structure is that the distribution property of pixels which has possibility about overlap simultaneously.

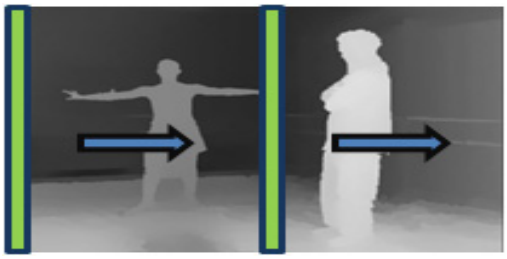

**Figure 9:** Vertical parallel structure for forward warping.

## IV. Experimental Results

For the experiment, we used a ballet sequence produced by Microsoft Research. Figure 10 shows results of traditional algorithm and proposed algorithm. Significant errors are produced in Fig. 10 (a) while Fig. 10 (b) shows better quality which is result of proposed method. Left color image and generated right viewpoint image are presented in Fig. 11. Table 1 shows a comparison of computation time CPU and GPU. GPU device can boost computation power about 15time than a CPU operation.

(a) (b)

**Figure 10:** (a) Result of the usual directional interpolation (b) Result of the proposed hole filling algorithm.

(a) (b)

**Figure 11:** (a) Left viewpoint input image (b) Right viewpoint image generated by using (a) and corresponding depth map.

**Table 1:** Comparison of computation time for a 720x480 sequence.

| System | Time | Relative speed |
|---|---|---|
| CPU<br>P4 Core2 2.66GHz | 125.7ms | 1 |
| GPU(CUDA)<br>nVIdia GTX 260 | 8.15ms | 15.4 |

## IV. Conclusions

In this paper, we developed a fast virtual viewpoint image generation method based on the DIBR method. In order to generate a virtual view image in real-time for the 3D broadcasting system, we use GPU parallel programming which called CUDA. Simple and efficient hole filling method is proposed to reduce complexity and prevent hole filling error. In order to take advantage of the single instruction multiple threads structure of GPU, we collect independent data sets and use high speed memories. From these techniques, we can produce virtual viewpoint images 15 times faster than the serial CPU processing. Therefore, low bandwidth and real-time stereo video broadcasting system is possible.

## Acknowledgments

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2010-(C1090-1011-0003))

## References

1. O. Schreer, P. Kauff, and T.Sikora, 3D Videocommunication, John Wiley&Sons, 2005.
2. L. McMilan, “A List-Priority Rendering Algorithm for Redisplaying Projected Surfaces,” Technical Report TR95-005, Univ. of North Carolina, 1995.
3. G. Wolberg, Digital Image Warping, IEEE Computer Society Press, 1990.
4. ISO/IEC JTC1/SC29/WG11, Contribution for 3D Video Test Material of Outdoor Scene, M1537, 2008.
5. K. Oh, S. Yea, and Y. Ho, “Hole Filling Method using Depth Based In-painting for View Synthesis in Free Viewpoint Television and 3-D Video,” Picture Coding Symposium, pp. 39 (1-4), 2009.
6. H. Shin, Y. Kim, H. Park, and J. Park, “Fast View Synthesis using GPU for 3D display,” IEEE Transactions on Consumer Electronics, Vol. 54, No. 4, pp. 2068-2076, 2008.
7. nVIDIA Corporation, “CUDA 2.3 Programming Guide,” in http://www.nvidia.com/object/cuda_develop.html, 2009.