

## GPU를 이용한 Gaussian Hole-Filling Algorithm 가속<sup>1)</sup>

박준호<sup>○</sup>, 한탁돈<sup>\*</sup>

<sup>○\*</sup>연세대학교 컴퓨터과학과

e-mail: [bluedawn@msl.yonsei.ac.kr](mailto:bluedawn@msl.yonsei.ac.kr)<sup>○</sup>, [hantack@msl.yonsei.ac.kr](mailto:hantack@msl.yonsei.ac.kr)<sup>\*</sup>

## Accelerating Gaussian Hole-Filling Algorithm using GPU

Jun-ho Park<sup>○</sup>, Tack-Don Han<sup>\*</sup>

<sup>○\*</sup>Dept. of Computer Science, Yonsei University

### ● 요약 ●

3차원 멀티미디어 서비스에 대한 관심이 높아짐에 따라 관련 연구들이 현재 다양하게 논의되고 있다. Stereoscopy 영상을 생성하기 위한 기존의 방법으로는 두 대의 촬영용 카메라를 일정한 간격으로 띄워놓고 피사체를 촬영한 후 해당 좌시점과 우시점을 생성하는 방법을 이용하였다. 하지만 이는 영상 대역폭의 부담을 가져오게 된다. 이를 해결하기 위하여 Depth 정보와 한 장의 영상을 이용한 DIBR (Depth Image Based Rendering) Algorithm에 대한 연구가 많이 이루어지고 있다. 그중 Gaussian Depth Map을 이용한 Hole-Filling 방법은 DIBR에서 가장 자연스러운 결과를 보여주지만 다른 DIBR Algorithm들에 비해 속도가 현저히 느리다는 단점이 있다. 본 논문에서는 영상 생성의 고속화를 위해 GPU를 이용한 Gaussian Hole-Filling Algorithm의 병렬처리 구조를 제안하고 이를 이용한 DIBR Algorithm 생성과정을 제시한다.

**키워드:** DIBR, Hole-Filling, GPU, CUDA, Parallel Processing

### 1. 서론

최근 2개의 영상을 하나로 합성하여 입체감을 나타낼 수 있는 3D TV가 등장하였다. 이는 현재 각종 멀티미디어 서비스에 사용되고 있으며 각 가정으로까지 점점 활성화되는 추세에 있다. 이처럼 많은 사용자들은 기존의 TV에서는 느끼기 힘들었던 영상의 입체감과 입체감을 느끼려는 시도가 늘어나고 있다. 이에 관련 연구들이 다양한 기업들에 의해 시도되고 있다.

DIBR (Depth Image-Based Rendering)은 기존의 Stereoscopy 영상 생성 방법과 다른 Algorithm이다. 이는 한 장의 Color 영상과 이에 해당되는 Depth 영상을 이용하여 가상의 좌시점과 우시점을 생성하는 방법이다. DIBR Algorithm을 이용함으로써 제한된 입력영상의 시점으로부터 다양한 시점의 영상을 생성할 수 있다. 때문에 사용자는 이를 이용하여 사용자가 원하는 임의의 시점 영상을 생성할 수 있게 된다.

하지만 DIBR 영상을 생성하는 과정에서 정보가 없는 Hole 영역이 발생하게 되고, 이를 소프트웨어적으로 보간 해줘야 한다. 때문에 일반적으로 DIBR 영상을 방법으로는 영상을 Warping한 후

발생하는 Hole을 각기 다른 Hole-Filling Algorithm을 이용해 보간해 주는 과정을 거친다. 이에 Hole을 보간하는 다양한 Algorithm들이 제시되고 있다.

Zhang은 Warping과 Hole-Filling 과정에서 DIBR 영상의 왜곡과 잡음을 제거하는 방법으로 Depth 영상을 Gaussian-Smoothing하는 방법을 제안하였고 이를 이용하여 영상의 Hole에서 발생하는 기하학적 잡음을 크게 줄일 수 있었다[1]. 이밖에도 여러 DIBR Algorithm들이 있지만, Gaussian-smoothing은 결과 영상의 Hole 영역의 품질이 매우 우수함을 보여주었다. 하지만 이 방법은 다른 방식들에 비해 느린 처리 속도를 가지고 있어 DIBR을 실시간으로 적용하기에 어려움이 있다.

DIBR은 Pixel 단위의 반복된 연산이 많지만 Smoothing Depth 영상을 이용할 경우에는 수식의 복잡도가 늘어 처리속도가 지연되게 된다. 본 논문에서는 Gaussian 분포의 Table을 이용한 smoothing을 통해 수식의 복잡도를 감소시켰다. 또한 이를 병렬 처리가 가능한 GPU를 이용하여 반복적인 연산을 효율적으로 처리하여 속도를 향상시켰다.

1) 이 논문은 2011년 정부 (교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임  
(한국연구재단-2011-No.2011-0027450)

## II. 관련 연구

### 1. DIBR 영상의 생성과정



그림 1. DIBR System 구현과정  
Fig. 1. DIBR system realization process

그림 1은 DIBR Algorithm을 생성하는 일반적인 세 단계의 과정을 보여주고 있다. 먼저 Pre-processing 과정에서는 후의 Depth 영상을 사용하기 위해 Depth 카메라로부터 얻은 데이터를 가공한다. 특성상 Depth영상은 데이터에 잡음이 심하기 때문에 이를 바로 사용하기 어렵다. 또한 보다 정확한 Warping과 Hole-filling과정을 위해 사용자가 설정한 객체에만 반응해야 한다. 전처리를 거친 후 영상을 Warping하는 과정을 거치게 되는데, 이는 입체감을 주고자 하는 대상 객체를 배경에서 떼어내어 이동하는 과정을 말한다. 이때 두 눈의 시점이 다르기 때문에 좌영상과 우영상을 따로 생성해주어야 한다.

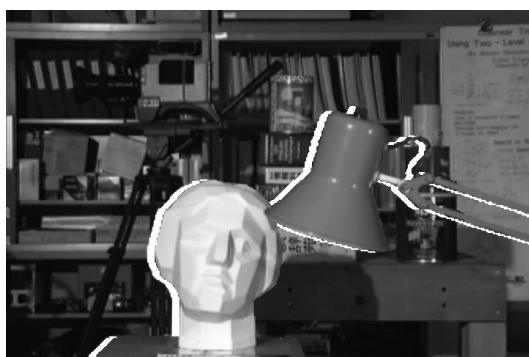


그림 2. DIBR Algorithm에서 발생하는 Hole 영역  
Fig. 2. Hole area in DIBR Algorithm

위와 같이 DIBR을 이용하여 가상시점의 새로운 영상을 생성할 경우 색상정보와 깊이정보가 없는 새로운 영역이 발생하게 되는데 이를 Hole이라 하고, Hole-Filling Algorithm을 통해 처리한다[2].

### 2. Gaussian Filter를 활용한 DIBR Algorithm

Depth map에 Gaussian-smoothing하는 방법을 사용함으로써 새롭게 생성된 합성 영상의 품질이 향상될 수 있다[3]. Depth 영상에 Gaussian Filter  $g(x, \sigma)$ 를 적용하기 위해 아래의 식(1)을 이용한다.

$$g(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{x^2}{\sigma^2}\right\}, \quad \left(\text{for } -\frac{w}{2} \leq \frac{w}{2}\right) \quad (1)$$

$w$ 는 영상의 window size를,  $\sigma$ 는 표준편차를 가리키며  $\sigma$  값에 따라 Depth 영상의 Smoothing 정도를 결정할 수 있다. 다음으로 사용자가 설정한 객체를 이동시키는 Warping과정을 수행해야 하는데 이때 식 (2)를 이용하여 Hole-Filling과정도 동시에 진행된다.  $x_l$ 과  $x_r$ 은 각각 좌영상과 우영상 각각을 가리키며  $t_x$ 는 두 가상 카메라 사이의 거리이다.  $x_c$ 와  $Z$ 는 Depth 영상과 Color영상을 통해 결정된다.

$$x_l = x_c + \frac{t_x}{2} \frac{f}{Z}, \quad x_r = x_c - \frac{t_x}{2} \frac{f}{Z} \quad (2)$$

## III. 본 론

### 1. DIBR 영상의 생성과정

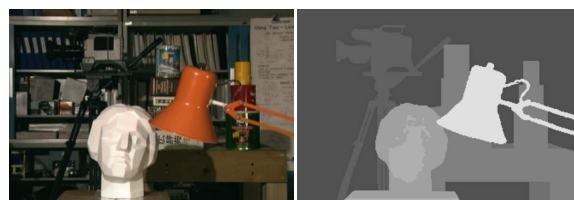


그림 3. 실험에 사용된 Color영상과 Depth영상  
Fig. 3. Color Image and Depth Image used in the test

GPU는 병렬처리과정에서 단일 명령어로 복수의 데이터를 처리할 수 있는 SIMT(Single Instruction, Multiple Threads) 구조를 가지고 있다. 이는 Pixel들을 Thread에 배분하여 다수에 데이터에 같은 명령을 동시에 수행하는 것이 가능하다[4].

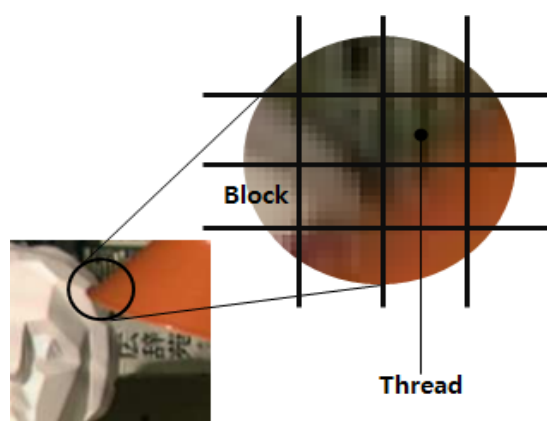


그림 4. 병렬처리를 위한 영상 Pixel들의 분배  
Fig. 4. The divide of Pixels for Parallelism

본 논문에서는 그림 4와 같이 GPU의 병렬처리를 이용하여 Depth영상에 Smoothing을 적용하였다. Pixel들은 GPU상에서 Gaussian Table과 함께 연산을 수행해야 하기 때문에 우선 GPU

Device상으로 Mapping된다. GPU상 각각의 Kernel들은 각 Pixel들을 Gaussian Table과 Thread단위로 병렬처리 하게 된다 [5].

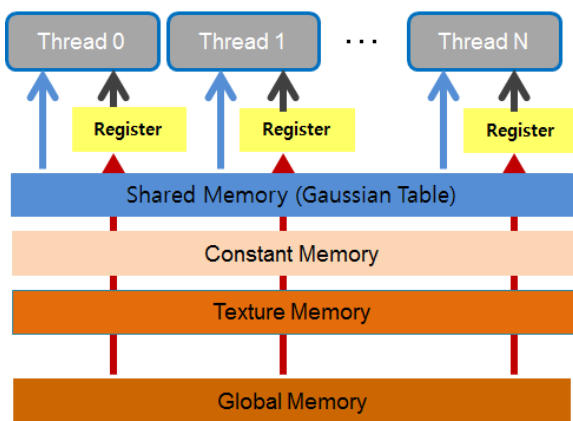


그림 5. Gaussian Table과 각 Thread들의 연결  
Fig. 5. Connection between Thread and Gaussian Table

Gaussian Table은 Thread가 Memory를 참조하는 빈도수가 Pixel수에 비례하기 때문에 그림 5와 같이 고속으로 접근이 가능한 Shared Memory에 저장한다. Thread의 개수는 Depth영상의 Width와 동일하기 때문에, Block의 개수는 Height와 동일하다. 때문에 Shared Memory에 각각의 Table을 저장하여 사용한다. 이는 Table을 중복해서 저장하기 때문에 Global Memory에 Table을 저장하여 사용하였을 때보다 많은 Memory를 사용한다. 하지만 GPU는 Global Memory에서 Data를 참조하기 위해 접근하는 지연시간이 약 500clock cycle로 매우 높고 이는 성능 하락의 주요 원인 중 하나로 작용한다. 또한 처리될 데이터가 GPU상의 Register로 Load되기 때문에 Global Memory에 직접 접근하도록 하는 경우, 접근지연으로 인해 손실이 발생할 수 있기 때문에 본 논문에서는 Shared Memory를 이용하였다.

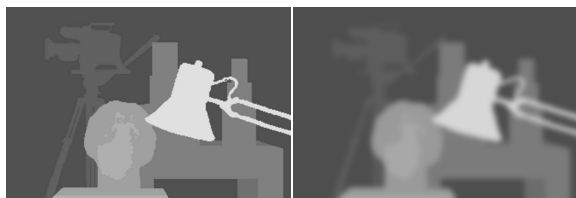


그림 6. GPU 병렬처리를 이용한 Smoothing Depth영상  
Fig. 6. The Smoothing Depth Image Using GPU

그림 6은 위의 병렬처리 과정을 거친 Smoothing Depth 영상의 최종 결과물을 보여주고 있다. 실제 Gaussian 연산을 이와 같이 행렬 Table로 사용하는 것은 실제 영상 처리에서는 부정확한 결과를 가져올 수 있다. 하지만 다른 영상처리와 달리 DIBR Algorithm에서의 Gaussian 연산은 Depth영상에서 Edge와 객체

간의 명암의 경계면을 완화하려 하기 때문에 최종 영상에는 거의 영향을 미치지 않게 된다.

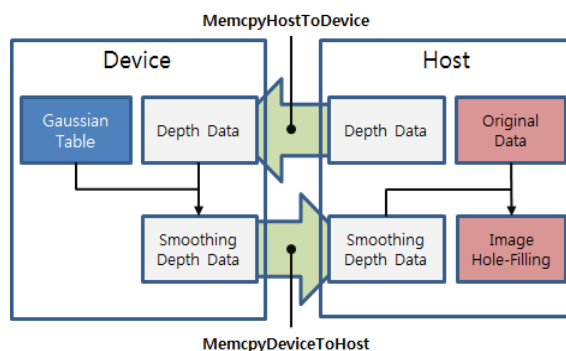


그림 7. 본 DIBR에서 CPU와 GPU Device의 주요 처리과정  
Fig. 7. Principal steps of the CPU and GPU Device of the DIBR

Depth영상을 병렬처리 하는 전체적인 과정을 나타내면 그림 7과 같다. 우선 Smoothing 전처리를 위하여 Depth 영상을 GPU Device로 보내준다. Pixel들은 각각의 Kernel에서 전역 메모리에 할당되어 있는 Gaussian Table과 함께 병렬로 연산을 수행하게 되고 다시 Host로 복호화 되는 과정을 거친다. 복호된 Smoothing Depth영상은 Color영상과 함께 식 2를 이용하여 연산하여 좌영상과 우영상을 각각 생성하게 된다.

#### IV. 실험 결과

표 1. "Table" DIBR영상의 CPU와 GPU 처리속도 비교  
Table1. Comparison CPU with GPU of "Table" Image

| List                              | CPU   | Our Algorithm (CPU + GPU) |
|-----------------------------------|-------|---------------------------|
| Gaussian Smoothing                | 273ms | 24ms                      |
| Warping and Hole-Filling Equation | 31ms  | 31ms                      |
| CPU-to-GPU Memory Copy            | None  | 6ms                       |
| Total Time                        | 314ms | 61ms                      |

표 1은 본 논문에서 제안한 병렬구조를 이용한 GPU 기반 Gaussian Hole-Filling Algorithm과 CPU 기반의 DIBR Algorithm의 처리속도를 측정한 결과이다. CPU에서는 평균 273ms의 처리속도를 보여줌으로서 기존에 언급된 느린 Gaussian Smoothing을 보여주고 있다. 이를 본 논문에서 제안한 GPU 기반 병렬처리를 수행함으로써 약 10배 이상 향상시켰다. Warping과 Hole-Filling을 수행하는 과정은 두 경우다 CPU에서 수행하였기 때문에 동일하게 31ms로 측정되었다. 마지막으로 Memory Copy

는 CPU와 GPU간의 영상 데이터를 이동하는데 발생하는 시간을 의미한다. GPU기반 Algorithm에서는 CPU와 GPU간 데이터 이동이 발생해 약간의 지연이 있음을 확인할 수 있다.

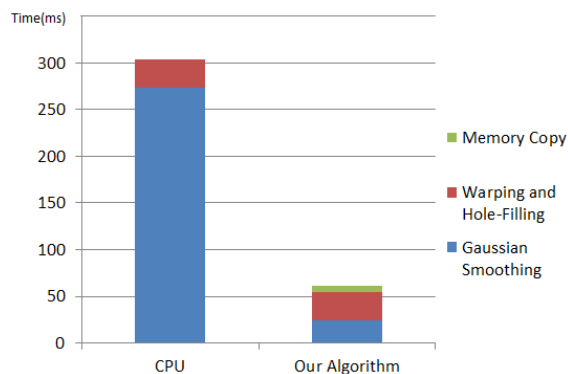


그림 8. 전체 영상 생성시간의 분석  
Fig. 8. Analysis of Total Processing Time

또한 각각의 과정에서 걸린 시간을 합한 그림 8에서 는 GPU를 이용함으로써 기존 생성방법에 비해 전체 영상 생성시간이 약 5배 정도 향상된 것을 확인할 수 있다.

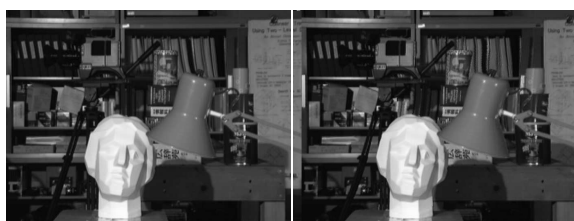


그림 9. 원본 영상(좌)과 최종 결과 영상(우)의 비교  
Fig. 9. Comparison Original image with Final result Image

## V. 결론

본 논문에서는 품질은 뛰어나지만 수식의 복잡도가 높아 영상의 처리속도가 높은 Gaussian-smoothing을 제안한 병렬처리 구조를 CUDA를 이용하여 DIBR Algorithm을 수행하였다. 전체 Algorithm에서 Gaussian-smoothing이 오버헤드가 가장 크기 때문에 영상의 크기가 커지면 커질수록 본 병렬처리 구조는 더 효율적인 결과를 나타낼 것이다. 본 연구의 성능분석과 구조는, 현재 활발히 연구되고 있는 3D TV DIBR 및 각종 응용분야에서 다양한 방법으로 적용될 수 있을 것이다.

## 참고문헌

- [1] W. J. Tam, G. Alain, L. Zhang, T. Martin, R. Renaud, "Smoothing depth maps for improved stereoscopic image quality", Proceedings of SPIE Conference on Three-dimensional TV, Video, and Display III, Vol. 5599, pp.162-172, Philadelphia, U.S.A., Oct. 2004.
- [2] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV", Proceedings of SPIE Conference on Stereoscopic Displays and Virtual Reality Systems XI, Vol. 5291, pp. 93-104, CA, U.S.A., Jan. 2004.
- [3] L. Zhang and W. Tam, "Stereoscopic image generation based on depth images for 3D TV," IEEE Trans. Broadcast., vol. 51, no. 2, pp. 191-99, June 2005.
- [4] G. Bravo, S. Zinger and P.H.N. de With, "Real-time free-viewpoint DIBR on GPUs for 3DTV systems", ICCV-Berlin, Sept. 2011.
- [5] I.Y Shin, Y.S. Ho and E.K. Lee, "Fast Stereo Video Generation Based on Depth Video using GPU", The Institute of Electronics Engineer of Korea(IEEK), Vol.32, pp. 259-260, Jul. 2009