Multi-GPU 기반의 고속 디지털 홀로그램 생성

송 중 석^{a)}, 박 정 식^{a)}, 서 영 호^{b)}, 박 종 일^{a)†}

Fast Generation of Digital Hologram Based on Multi-GPU

Joong-Seok Song^{a)}, Jungsik Park^{a)}, Young-Ho Seo^{b)}, and Jong-Il Park^{a)†}

요 약

실시간 홀로그래피 방송을 제작하기 위해서는 디지털 홀로그램을 고속으로 생성하는 것이 중요하다. 본 논문에서는 디지털 홀로그램 생성을 위한 Computer-Generated Holography(CGH) 식의 병렬 구조를 최적화하고, Compute Unified Device Architecture(CUDA)와 Open Multi-Processing (OpenMP)를 이용한 Multi Graphic Processing Unit(Multi-GPU) 기반의 디지털 홀로그램의 고속 생성을 위한 최적화 기법을 제안한다. 디지털 홀로그램을 생성하는 과정은 독립적인 연산을 할 수 있는 다수의 개체로 병렬화 할 수 있는 구조이기 때문에 이에 특화된 CUDA와 OpenMP를 사용함으로써 CGH식을 고속으로 연산할 수 있다. 여기서 더 나아가 이를 최적화하기 위해서 상수화, 벡터화, 루프폴기 등의 방법을 제안한다. 본 논문에서 제안된 기법을 통해서 기존 CPU에서의 CGH 연산속도에 비해 약 9,700배 정도의 속도를 개선할 수 있었다.

Abstract

Fast generation of digital hologram is of importance for real-time holography broadcasting. In this paper, we propose such a method that parallelizes the Computer-Generated Holography (CGH) algorithm for digital hologram generation and make it faster using Multi Graphic Processing Unit (Multi-GPU) with help of the Compute Unified Device Architecture (CUDA) and the Open Multi-Processing (OpenMP). In addition, we propose optimization methods such as fixation variable, vectorization, and loop unrolling for making the CGH algorithm much faster. Experimental results show that our method is about 9,700 times faster than a CPU-based one.

Keyword : 3D display, digital hologram, GPU

1. 서 론

최근 3D 디스플레이 콘텐츠가 서서히 늘어나고 상용화

가 이루어지면서 점차 3D 영상 디스플레이 기법에 대한 관심이 높아지고 있다. 특히, TV 산업분야에서는 3D 영상 디스플레이 장치 및 원천기술이 매우 중요하기 때문에 학계와 산업계에서 많은 역량을 기울이고 있다. 최근 떠오르는 이슈로 가장 이상적이며 궁극적인 3D 영상 디스플레이 기법인 홀로그래피가 주목을 받고 있다. 홀로그래피 기법은 3D 영상을 디스플레이 하는데 가장 좋은 방법이라고 알려져 있다. 홀로그래피 기술은 특별한 관찰 장비 없이 육안으로 관찰하는 것과 거의 비슷한 깊이 정보와 영상을 제공한다. 그러나 3D 디스플레이 기술로서 활용하기 위해서는 몇

a) 한양대학교
Hanyang University

b) 광운대학교
Kwangwoon University

† 교신저자 : 박종일 (jipark@hanyang.ac.kr)

※ 본 연구는 지식 경제부 및 한국산업평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호 처리 요소 기술 및 SoC 개발 · 접수일(2011년9월6일), 수정일(2011년10월31일), 게재확정일(2011년10월31일)]

가지의 문제점들을 극복해야 하기 때문에 실질적으로 상용화를 하기에는 많은 어려움이 있다. 가장 큰 문제점으로는 실제 크기의 영상을 홀로그램으로 제작하기 위해 고비용의 대형 광학계 장치가 필요하다는 점이 있고, 이 장치는 주변 광원에 영향을 많이 받기 때문에 홀로그램을 생성할 때 매우 안정적인 실험 환경이 필요하다는 제약이 있다^[1]. 이런 문제를 해결하기 위해 1966년, Brown과 Lohmann에 의해 Computer-Generated Holography(CGH)기법이 제안되었다^[2]. CGH는 기존의 광학계 장치와 광학 변수들을 수학적으로 모델링하여 일반적인 컴퓨터로도 손쉽게 홀로그램을 만들어 낼 수 있는 기법이다. 하지만 CGH기법을 이용하여 디지털 홀로그램을 생성하기 위해서는 상당한 양의 데이터들을 연산해야 하기 때문에 이를 고속으로 처리하는 것이 매우 중요하다.

해외에서는 현재 고속으로 CGH를 연산하기 위한 연구가 많이 진행되고 있다. 일본의 Chiba대에서는 3차원 객체와 생성하려는 디지털 홀로그램의 거리의 차를 반복적으로 연산하여 생성시간을 줄이는 기법을 제안하였다^[3]. 하지만 이 기법은 많은 승산 연산이 필요하다는 단점이 있다. 그리고 미국의 MIT에서는 일반적인 연산식의 각 항들을 미리 계산해 놓은 LUT(Look-up Table)를 이용하는 기법을 제안하였으나, 이 기법은 많은 메모리 자원을 필요로 한다는 단

점이 있다^[4,5]. 국내에서도 고속으로 CGH 연산을 하기위한 연구가 이루어지고 있는데 대표적으로 광운대에서는 Field Programmable Gate Array(FPGA) 기반의 하드웨어 칩을 활용해서 고속으로 디지털 홀로그램을 생성하는 기법을 제안하였다^[6]. FPGA를 이용하면 구현과정이 매우 복잡하고 개발기간이 오래 걸린다는 단점이 있지만 하드웨어의 직접적인 연산능력으로 인해 소프트웨어 기반에 비해 상대적으로 고속으로 연산할 수 있다는 장점이 있다. 본 논문에서는 Graphic Processing Unit(GPU)과 Central Processing Unit(CPU)의 병렬처리 연산능력을 활용하여 고속으로 디지털 홀로그램을 생성하는 방식을 제안하며 이를 위한 몇 가지 최적화 기법들을 제안한다.

II. 디지털 홀로그래피

1. 컴퓨터 생성 홀로그램(CGH)

일반적인 홀로그램은 그림 1과 같은 광학계 장치를 이용하여 생성할 수 있다. 그림 1-(a)는 홀로그램을 기록하는 과정이다. 레이저 빔이 분광기에 의해 물체파와 기준파로 나뉘는데, 물체파는 물체에 한번 조사된 후 필름으로 다시 조사되

그림 1 . 디지털 홀로그램 생성 과정
Fig. 1. Generation of digital hologram

고, 기준파는 곧바로 필름으로 조사된다. 이때 두 빛이 만나면서 빛의 간섭현상이 발생하고 이때 생기는 간섭패턴이 감광 필름에 기록되는데 이것을 홀로그래프라고 부른다. 그림 1-(b)는 홀로그래프를 다시 객체로 복원하는 과정으로, 생성하였던 홀로그래프에 동일한 기준파를 조사하면 다시 원래 3D 객체형상을 복원한다. 이 방법은 안정된 실험 환경이나 고가의 광학계 장치 등을 필요로 하기 때문에 실질적으로 홀로그래프를 생성하기에는 어려움이 있다. 그래서 본 논문에서는 범용 컴퓨터와 같은 디지털 전자장치를 활용하여 홀로그래프를 생성할 수 있는 CGH기법을 사용한다. CGH기법은 광학계를 수학적으로 모델링하여 범용 컴퓨터로 디지털 홀로그래프를 생성하는 기법이다. 다음 식 (1)은 CGH 연산을 나타낸다.

$$I_{\alpha} = \sum_j^N A_j \cos(k \sqrt{(px_{\alpha} - px_j)^2 + (py_{\alpha} - py_j)^2 + z_j^2}) \quad (1)$$

표 1. CGH 광학계 파라미터
Table 1. CGH Optics Parameter

파라미터	내용
α	디지털 홀로그래프 Index
j	3D 객체 Index
I_{α}	디지털 홀로그래프
A_j	3D 객체
x_{α}, y_{α}	디지털 홀로그래프 좌표
x_j, y_j, z_j	3D 객체 좌표
p	홀로그래프의 화소 크기 (pixel pitch)

그림 2 . 디지털 홀로그래프 생성 과정
Fig. 2. Generation of digital hologram

위 수식은 3D 객체의 깊이영상 정보를 이용해서 홀로그래프를 생성하는 것을 나타낸다. 3D 객체 영상에서 깊이정보를 갖고 있는 픽셀들을 광원이라고 하고, 식 (1)은 3D 객체의 모든 광원에 대해서 동일하게 연산을 수행한다. 3D 객체의 모든 광원들은 각각 디지털 홀로그래프를 한 장씩 생성하고, 전체 광원에서 생성되는 각각의 디지털 홀로그래프들을 한 장으로 축적시켜야 비로소 3D 객체 영상에 대한 디지털 홀로그래프가 생성된다. 그림 2는 디지털 홀로그래프의 생성과정을 표현하고 있다.

2. CUDA와 OpenMP를 이용한 CGH연산 고속화

2.1 CGH 연산의 고속화

CGH 기법은 원리적으로 많은 데이터를 연산해야 하기 때문에 고속으로 연산하는 기법이 매우 중요하다. CGH연산식 (1)은 그림 2와 같이 3D 객체의 모든 광원에 대해서 병렬적으로 연산되는 특징을 갖고 있다. 각 광원에서 생성되는 디지털 홀로그래프들을 한 장으로 축적하면 3D 객체에 대한 디지털 홀로그래프가 생성하는데 이 과정이 전체적으로 상당한 연산량을 필요로 한다. 예를 들어서, 생성하려는 3D 객체의 크기가 200×200 pixel 이고 생성하려는 디지털 홀로그래프의 크기가 1,024×1,024 pixel 이라면, 총 41,943,040,000번을 연산해야 디지털 홀로그래프 한 장을 생성할 수 있다. 이런 방대한 연산량 때문에 CGH를 고속으로 연산하는 방법은

현재까지도 해결해야할 핵심 문제로 남아있다. 이 문제를 해결하기 위해, 본 논문에서는 Multi-GPU기반에서 CUDA와 OpenMP의 병렬처리 연산을 통해 CGH를 고속으로 연산하는 방법을 제안하고자 한다.

2.2 CUDA & OpenMP

CPU와는 달리 GPU는 구조적으로 연산능력이 매우 뛰어나다. 그림 3을 보면 CPU보다 GPU가 더 많은 Arithmetic and Logic Unit(ALU)을 갖고 있음을 볼 수 있다. ALU의 개수가 많으면 더 많은 양의 데이터들을 효율적으로 연산할 수 있다. 그래서 방대한 데이터 연산 처리가 필요한 의학, 경제, 이미지 프로세싱 등 기타분야에서는 이미 GPU가 많이 활용되고 있다. 본 논문에서는 CGH의 방대한 데이터 연산을 위해서 NVIDIA사에서 개발한 GPU기반 병렬처리 아키텍처인 CUDA를 사용한다^[7]. CUDA는 GPU를 구

그림 3 . CPU와 GPU의 구조

Fig. 3. Architecture of CPU and GPU

그림 4 . Grid 와 Block

Fig. 4. Grid & Block in CUDA

성하고 있는 다수의 스트림 코어들을 통해 동시에 다수의 작업들을 연산할 수 있는 시스템이다. 그림 4와 같이 Grid와 Block이라는 단위로 작업들을 나누어서 GPU의 각각의 스트림 코어에 할당을 하고 동시에 연산을 수행한다. 일반적으로 스트림 코어의 개수가 많으면 많을수록 동시에

※ 3D 객체의 각 광원은 각각의 Thread로 CGH연산을 수행하며 디지털 홀로그램을 한 장 생성한다.

그림 5 . CUDA를 이용한 디지털 홀로그램 생성

Fig. 5. Generation of digital hologram using CUDA

연산할 수 있는 작업의 개수도 늘어나기 때문에 일반적으로 Single-GPU기반보다 Multi-GPU 기반으로 사용했을 때가 성능이 더 뛰어나다. 하지만 CUDA는 원리적으로 Multi-Kernel을 지원하지 않기 때문에 CPU의 Multi-Core를 활용한 OpenMP를 이용해서 병렬적인 CUDA 시스템을 구축해야 다수의 GPU를 활용할 수 있게 된다^[8]. 그림 5는 각 광원에 대한 CGH 연산이 CUDA를 통해서 병렬적으로 동시에 처리됨을 나타내고 있다. 이러한 과정이 3D 객체의

모든 광원에서 반복된다. 그림 6은 OpenMP를 활용한 병렬적 CUDA시스템과 이를 이용한 CGH연산과정을 보여주고 있다. 3D 객체의 정보(광원의 좌표, 깊이정보)를 GPU의 개수만큼 균등 분할하여 각 GPU의 CUDA 커널에 전송하고 각 커널은 병렬적으로 전송받은 데이터를 CGH 수식에 대입하여 연산을 하게 된다. 연산 후 생성된 각각의 디지털 홀로그램들을 한 장으로 축적하면 3D객체의 디지털 홀로그램이 생성된다.

그림 6 . Multi-GPU 기반에서의 디지털 홀로그램 생성
Fig. 6. Generation of digital hologram based on Multi-GPU

3. CGH 연산 최적화 기법

GPU기반의 시스템에서 효율적인 CGH연산을 수행하기 위해서는 시스템에 맞는 최적화 작업이 필요하다. 본 논문에서는 상수화, 벡터화, 루프 풀기 등을 통해서 전체적인 CGH연산을 최적화하였다.

3.1 상수화

CGH연산식 (1)을 Taylor 전개 후 근사화를 하면 다음 수식 (2)와 같이 정리할 수 있다.

$$I_{\alpha} = \sum_j^N A_j \cos\left(\frac{2\pi}{\lambda} \left(Z_j + \frac{p^2}{2Z_j} ((x_{\alpha} - x_j)^2 + (y_{\alpha} - y_j)^2)\right)\right) \quad (2)$$

식 (2)를 정리하면 다음과 같이 정리할 수 있다.

$$\begin{aligned} I_{\alpha} &= \sum_j^N A_j \cos\left(2\pi \left(\frac{Z_j}{\lambda} + \frac{p^2}{2\lambda Z_j} ((x_{\alpha} - x_j)^2 + (y_{\alpha} - y_j)^2)\right)\right) \\ &= \sum_j^N A_j \cos\left(\frac{Z_j 2\pi}{\lambda} + \frac{2\pi p^2}{2\lambda} \times \frac{((x_{\alpha} - x_j)^2 + (y_{\alpha} - y_j)^2)}{Z_j}\right) \end{aligned} \quad (3)$$

식 (3)을 통해 3D 객체의 모든 광원에 대하여 연산을 하

게 되는데 반복적으로 연산되는 부분들을 사전에 미리 계산해서 정의를 해 놓으면 다음의 식 (4)와 같이 단순화 할 수 있다.

$$\begin{aligned} &= \sum_j^N A_j \cos\left(\frac{Z_j 2\pi}{\lambda} + \frac{2\pi p^2}{2\lambda} \times \frac{((x_{\alpha} - x_j)^2 + (y_{\alpha} - y_j)^2)}{Z_j}\right) \\ &= \sum_j^N A_j \cos\left(Z_j A + \frac{BR}{Z_j}\right) \quad (R = ((x_{\alpha} - x_j)^2 + (y_{\alpha} - y_j)^2)) \quad (4) \\ &\quad \left(A = \frac{2\pi}{\lambda}, \quad B = \frac{2\pi p^2}{2\lambda} \quad A, B: \text{Constants}\right) \end{aligned}$$

식 (4)에서 나타내는 것과 같이 상수 A와 B로 미리 광학계 변수들을 계산해서 정의해 놓으면 매번 불필요한 연산을 할 필요가 없어진다. CGH 연산은 모든 광원에 대해서 적용되기 때문에 식의 연산 복잡도를 줄일 수 있다면 디지털 홀로그램을 생성하는데 필요한 전체적인 연산량을 크게 줄일 수 있다.

3.2 벡터화

CUDA는 구조적으로 Floating unit이 따로 있기 때문에 부동소수점 연산에 최적화가 되어 있다. 그래서 본 논문

```

1  #define PTMPSQ_d_WT 5.3680041297357351764902622432859e-4f
2
3  __global__ static void reduceKernel(
4  float SolNumber, float4 *DATA_device, float *out_data)
5  {
6      int idx = blockIdx.x * blockDim.x + threadIdx.x;
7      int idy = blockIdx.y * blockDim.y + threadIdx.y;
8
9      int index = idy*IMAGE_WIDTH+idx;
10
11      float m_A = 76.589839037841389192655003608137f;
12
13      float TempA,TempB,TempC,TempD,TempE,TempF,TempG,TempH;
14      float4 RA, zA, ThetaA, DEPTH;
15
16      int k;
17
18      for(k = SolNumber-5; k >= 0; k = k-4)
19      {
20          TempA = idx-DATA_device[k+0].x*20;
21          TempB = idy-DATA_device[k+0].y*20;
22          TempC = idx-DATA_device[k+1].x*20;
23          TempD = idy-DATA_device[k+1].y*20;
24          TempE = idx-DATA_device[k+2].x*20;
25          TempF = idy-DATA_device[k+2].y*20;
26
27          TempG = idx-DATA_device[k+3].x*20;
28          TempH = idy-DATA_device[k+3].y*20;
29
30          RA.x = TempA*TempA + TempB*TempB;
31          RA.y = TempC*TempC + TempD*TempD;
32          RA.z = TempE*TempE + TempF*TempF;
33          RA.w = TempG*TempG + TempH*TempH;
34
35          ThetaA.x = m_A * DATA_device[k+0].w + PTMPSQ_d_WT *
36              (__fdivdef(RA.x, DATA_device[k+0].w));
37          ThetaA.y = m_A * DATA_device[k+1].w + PTMPSQ_d_WT *
38              (__fdivdef(RA.y, DATA_device[k+1].w));
39          ThetaA.z = m_A * DATA_device[k+2].w + PTMPSQ_d_WT *
40              (__fdivdef(RA.z, DATA_device[k+2].w));
41          ThetaA.w = m_A * DATA_device[k+3].w + PTMPSQ_d_WT *
42              (__fdivdef(RA.w, DATA_device[k+3].w));
43
44          out_data[index] +=
45              DATA_device[k+0].z*_cosf(ThetaA.x) +
46              DATA_device[k+1].z*_cosf(ThetaA.y) +
47              DATA_device[k+2].z*_cosf(ThetaA.z) +
48              DATA_device[k+3].z*_cosf(ThetaA.w);
49      }

```

그림 7. 최적화 기법이 적용된 CGH 연산을 위한 CUDA Kernel

Fig. 7. Kernel for the CGH calculation using CUDA with optimization

서는 광학계 변수들을 최대한 Float type으로 정의하고 CUDA에서 제공하는 Float4 벡터형 변수를 최대한 사용하였다. Float4 벡터형 변수는 x, y, z, w 총 4개의 Float type 변수를 통해 벡터를 표현할 수 있으며 이를 통해 CUDA 커널 내에서 CGH 연산을 효율적으로 구조화할 수 있다. 본 논문에서는 3D 객체의 좌표와 깊이정보들을 Float4 벡터형 변수에 저장하여 CUDA 커널로 전송할 때 사용한다.

3.3 루프 풀기

루프 풀기는 대표적인 프로그램 최적화 기법으로 널리 알려져 있다. 루프 풀기의 목적은 루프의 반복 횟수를 줄이고 그 만큼 루프 반복에 필요한 비교문을 생략시켜 연산속도를 증대시키는데 있다. CGH 연산은 3D 객체의 모든 광원에 대해 반복해서 연산을 수행하므로 루프 풀기를 적용

하면 효과적으로 연산시간을 줄일 수 있다. 그림 7은 CUDA 커널 내에서 최적화 기법들을 적용한 CGH 연산을 보여주고 있다. 18번째 줄부터 48번째 줄까지의 반복문은 루프풀기를 4개의 단위로 풀어서 CGH 연산을 하고 있는 것을 보여주고 있다.

III. 실험 및 결과

본 논문에서는 2장에서 정리한 수식 (4)를 이용하여 디지털 홀로그램을 생성하였다. 실험영상으로는 그림 8과 같이 Bunny, Kettle, Ring의 3D 객체영상들을 실험 영상으로 사용하였다. 실험에 사용된 3D 객체들의 크기는 200×200×8bit 이며, 광원은 Bunny가 6,899개, Kettle이 7,157개,

그림 8. (a) 3D 객체 영상 : Bunny, (b) 3D 객체 영상 : Kettle, (c) 3D 객체 영상 : Ring, (d) 3D 객체 복원 영상 : Bunny, (e) 3D 객체 복원 영상 : Kettle, (f) 3D 객체 복원 영상 : Ring
 Fig. 8. (a) 3D object : Bunny, (b) 3D object : Kettle, (c) 3D object : Ring, (d) 3D object reconstruction : Bunny, (e) 3D object reconstruction : Kettle, (f) 3D object reconstruction : Ring

Ring이 4,565개를 갖고 있다. 생성되는 홀로그램 해상도는 $1,024 \times 1,024$ pixel과 $1,536 \times 1,536$ pixel이며, Reference wave length(λ)는 633nm, Reconstruction distance는 1000mm, Pixel pitch(p)는 $10.4\mu\text{m} \times 10.4\mu\text{m}$ 로 파라미터를 설정하였다. 실험환경은 표 2와 같다. 그림 8은 실험에 사용된 3D 객체의 실험 영상들이다. 그림 8-(d),(e),(f)는 본 논문에서 생성한 디지털 홀로그램을 Holovision이라는 광학 프로그램을 사용해서 복원한 영상이다.

표 2. 실험 환경

Table 2. Environment of experiment

항목	세부사항
CPU	Intel(R) Core Quad 2.4GHz
GPU	Nvidia Geforce GTX 580, 2개
RAM	4.0 GB
OS	Microsoft Windows 7
Tool	Microsoft Visual Studio 2008
	OpenCV 2.0
	CUDA 3.0
	Holovision 2.2.1

표 3은 CPU기반과 Multi-GPU기반에서의 CGH 연산속도를 비교하고 있다. 1번 항목은 CPU 기반에서 CGH를 연산한 결과이고, 2번 항목은 Multi-GPU 기반에서 최적화를

적용하지 않고 CGH를 연산한 결과이다. 최적화를 적용하지 않은 Multi-GPU기반이 CPU기반보다 약 1,900배 가량 연산속도가 개선되었음을 볼 수 있다. 그리고 3번 항목부터 5번 항목은 Multi-GPU 기반에서 벡터화, 상수화, 루프 풀기 최적화 기법을 적용한 결과이다. 벡터화 같은 경우 약 1%, 루프 풀기 같은 경우 약 12%정도로 미미한 개선효과를 보이고 있지만, 상수화 같은 경우 약 78%로 개선효과가 상당하다는 것을 보여주고 있다.

IV. 결론

본 논문에서는 Multi-GPU를 이용해서 고속으로 CGH 연산을 하는 기법을 제안한다.

CGH는 3D 객체의 모든 광원에 대해서 독립적이기 때문에 병렬연산이 가능하다. 그래서 본 논문에서는 CUDA와 OpenMP를 이용해서 CGH연산을 고속화 하였고, 전체적인 CGH연산량을 줄이기 위해서 상수화, 벡터화, 루프 풀기와 같은 최적화 기법들을 제안하였다. 실험 결과, 본 논문에서 제안하는 기법은 기존의 CPU기반의 기법보다 약 9,700배 정도 빠르게 디지털 홀로그램을 생성할 수 있는 것을 볼 수 있었다.

본 논문에서 제안하는 최적화 기법은 실시간 홀로그램 생성의 기반이 되는 기술이 될 것으로 전망한다.

표 3. CPU 기반과 Multi-GPU 기반의 CGH 연산속도 비교

Table 3. Comparison of CGH calculation times CPU based system, Multi-GPU based system

항목		광원 1K개의 CGH 연산 속도 (ms)					
		1,024×1,024 pixel			1,536×1,536 pixel		
		Bunny	Kettle	Ring	Bunny	Kettle	Ring
1	CPU	185,682.2	186,214.6	185,454.7	407,992.4	407,489.7	416,786.4
2	Multi-GPU	96.6	99.6	98.7	223.6	232.5	231.2
3	Multi-GPU + 벡터화	95.1	95.6	98.0	213.2	212.6	220.9
4	Multi-GPU + 벡터화, 상수화	21.0	21.5	24.4	46.6	45.8	54.8
5	Multi-GPU + 벡터화, 상수화, 루프 풀기	18.2	18.1	21.6	40.4	39.8	47.0
연산 속도 비 (1 항목 : 5 항목)		1 : 10,202	1 : 10,288	1 : 8,585	1 : 10,099	1 : 10,238	1 : 8,868

참 고 문 헌

- [1] W. H. Ryu and M. H. Jeong, "A study on three-dimensional computer generated holograms by 3-D coordinates transformation", Applied Optical Society of Korea, volume 17, pp. 525-531, December 2006
- [2] B. R. Brown and A. W. Lohmann, "Complex spatial filtering with binary masks", Applied Optics, volume 5, pp. 967-969, June 1966.
- [3] H. Yoshikawa, S. Iwase, and T. Oneda, "Fast computation of Fresnel holograms employing difference", Optical Review, volume 8, pages 331-335, 2000.
- [4] M. Lucente, "Interactive computation of holograms using a look-up table", Journal of Electronic Imaging, volume 2, pages 28-34, January 1993.
- [5] D. E. Smalley and Q. Y. J. Smithwick, and V. M. Bove, "Holographic video display based on guided-wave acousto-optic devices", Proceedings of SPIE practical Holography XXI, volume 6488, February 2007.
- [6] H. J. Choi and Y. H. Seo, "Fast computation algorithm of Fresnel holograms using recursive addition method", Applied The Journal of Korea Information and Communications Society, volume 33, May 2008.
- [7] NVIDIA official website <http://www.nvidia.com/>.
- [8] OpenMP official website. <http://www.openmp.org/>.

저 자 소 개

송 중 석

- 2010년 : 한국산업기술대학교 전자공학과 학사
- 2010년 ~ 현재 : 한양대학교 전자컴퓨터통신공학과 석사 과정
- 주관심분야 : 3차원 영상처리, 컴퓨터 비전, 디지털 홀로그램, GPU 컴퓨팅

박 정 식

- 2010년 : 한양대학교 전자전기컴퓨터공학부 학사
- 2010년 ~ 현재 : 한양대학교 전자컴퓨터통신공학과 석사 과정
- 주관심분야 : 3차원 영상처리, 증강현실, 컴퓨터 비전, GPU 컴퓨팅

서 영 호

- 1999년 : 광운대학교 전자재료공학과 학사
- 2001년 : 광운대학교 일반대학원 석사
- 2004년 : 광운대학교 일반대학원 박사
- 2003년 ~ 2005년 : 한국전기연구원 연구원
- 2005년 ~ 2008년 : 한성대학교 조교수
- 2008년 ~ 현재 : 광운대학교 교양학부 부교수
- 주관심분야 : 2D/3D 영상 및 비디오 처리, 디지털 홀로그램, SoC 설계, 워터마킹/암호화

박 종 일

- 1987년 : 서울대학교 전자공학과 학사
- 1989년 : 서울대학교 전자공학과 석사
- 1995년 : 서울대학교 전자공학과 박사
- 1992년 ~ 1994년 : 일본 NHK 방송기술연구소 객원연구원
- 1995년 ~ 1996년 : 한국방송개발원 선임연구원
- 1996년 ~ 1999년 : 일본 ATR 지능영상통신연구소 연구원
- 1999년 ~ 현재 : 한양대학교 공과대학 컴퓨터공학부 교수
- 주관심분야 : 가상현실, 컴퓨터그래픽스/비전, 3차원 영상처리, 인간컴퓨터 상호작용