GPU기반의 디지털 홀로그램 고속 생성을 위한 최적화 기법

송중석 박종일1)

한양대학교 전자컴퓨터통신공학과

sj0818@mr.hanyang.ac.kr jipark@hanyang.ac.kr

An Optimization for fast digital hologram generation based on GPU

Song, Joong-Seok Park, Jong-Il

Dept. Electronics and Computer Engineering, Hanyang University

요약

디지털 홀로그램은 일반적으로 computer generated hologram(CGH)기법에 의해서 생성된다. 하지만 원리적으로 CGH 기법은 많은 연산량과 복잡도를 요구하고 있기 때문에 실시간으로 디지털 홀로그램을 생성하는 것은 매우 어렵다. 본 논문에서는 CGH 고속연산을 위해 graphics processing unit(GPU)의 병렬처리구조인 CUDA를 사용하였고, 추가적으로 다중 GPU 연산처리를 위해 OpenMP를 사용하였다. 더 나아가 이를 최적화하기 위해서 상수화, 벡터화, 루프풀기 등의 기법들을 제안한다. 결과적으로, 본 논문에서 제안된 기법을 통해서 기존 CPU에서의 CGH 연산속도에 비해 약 8,300배 정도의 속도를 개선할 수 있었다.

1. 서론

3D 영상을 효과적으로 표현할 수 있는 기법에는 애너글리프(Anaglyph), 스테레오 디스플레이, 홀로그램 등의 기법이 있다. 이 중에서 홀로그램은 3차원 객체를 가장 육안에 가깝게 관찰할 수 있는 기법으로 많은 연구가 이루어지고 있다. 홀로그램은 일반적으로 광학계(optical system)에 의해서 형성되는데, 이와 같은 경우는 외부의 광원에 매우 민감하여 실험환경의 광원들을 정교하게 제어해 줘야한다. 또한, 광학계의 미세한 떨림이나 움직임들이 홀로그램 생성에 큰 영향을 줄 수 있기 때문에 매우 안정된 실험환경은 필수적이다[1]. 이런 문제들은 홀로그램의 기술적 발전에 걸림돌이 되고 있으며 이를 해결하기 위해 Brown 등은 computer generated hologram(CGH)기법을 제안하였다[2]. 이 기법은 기존의 광학계를 수학적으로 모델링하여 일반 범용 컴퓨터에서도 홀로그램을 생성할 수 있게 해주는 기술이다. 따라서 이 기술을 사용하면 실제 공간에서의 3차원 객체로부터 쉽게 디지털 홀로그램을 추출해 낼 수 있다. 하지만 CGH를 통해 디지털 홀로그램을 계산하기 위해서는 방대한 양의 데이터에 대해서 복잡한 연산을 해야하는 문제가 있기 때문에, 이런 문제를 해결하기 위한 최적화 방법들이 많이 연구가 되고 있다. 예를 들어, Lucente 등은 기존의 CGH 연산과정들을 Look-up Table(LUT)에 저장하고 이를 이용해서 디지털 홀로그램을 고속으로 생성하는 알고리즘을 제안하였다[4, 5]. 하지만 이 기법은 3D 객체의 크기가 커질수록 LUT를 형성할 때 소요되는 메모리가 증가한다는 문제점이 있다. 그리고 Choi 등은 Field-programmable Gate Array(FPGA)를 기반으로 반복 가산 기법을 통하여 디지털 홀로그램을 근 실시간으로 생성하는 획기적인 알고리즘을 제안하였다[6]. 하지만 FPGA 같은 경우, 하드웨어 제작에 드는 비용이 많고, 소요기간이 길다는 단점이 있다.

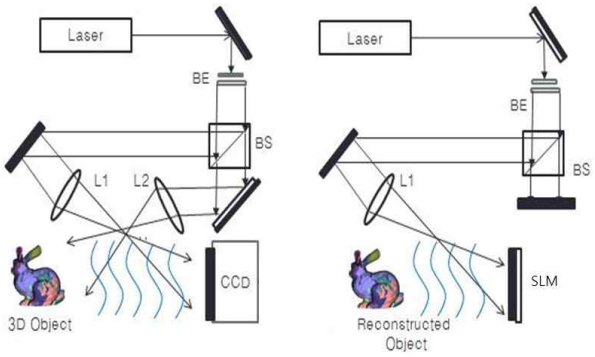
본 논문에서는 추가적인 하드웨어 없이 graphics processing unit(GPU)의 병렬처리구조를 이용하여 고속으로 디지털 홀로그램을 생성할 수 있는 최적화 기법을 제안한다. 병렬처리의 대표적인 연산기법인 CUDA[7]와 OpenMP[8]를 사용하여 CGH 연산시간을 대폭 줄이고, 더 나아가 CGH 연산량을 줄일 수 있는 최적화 기법들(상수화, 벡터화, 루프 풀기)을 적용하여 고속으로 디지털 홀로그램을 생성한다.

2. 홀로그램의 생성

그림 1은 광학계를 이용해서 홀로그램을 생성하는 과정이다. 그림 1-(a)은 홀로그램을 기록하는 과정을 보여주고 있다. 광 분리기(BS)는 빔을 물체파(L2)와 참조파(L1)로 나누고, 물체파는 3D 객체에 조사된 뒤 다시 반사되어 Charge Coupled Device(CCD)로 조사되고, 참조파는 CCD로 바로 조사된다. CCD로 조사되는 물체파와 참조파 사이에 간섭현상이 발생하면서 간섭무늬가 생성이 되고 이것이 CCD에 기록된다. 이 때 기록된 무늬를 홀로그램이라고 부른다. 그림 1-(b)는 홀로그램을 이용해서 3D 객체를 복원하는 과정을 보여주고 있다. 홀로그램에 참조파를 조사해주면 3D 객체 영상을 복원할 수 있다. 광학계 기반의 홀로그램 생성은 그림 1과 같이 복잡한 광학시스템이 필요하고 주

1) 교신저자

변 광원들을 잘 통제할 수 있는 안정된 실험 환경이 필요하다.



(a) 홀로그램 기록 (b) 홀로그램 복원

그림 1: 홀로그램 생성 과정.

이와 반대로, 디지털 홀로그램은 CGH 기법을 통해 추가적인 하드웨어 장치 없이 범용 컴퓨터에서도 쉽게 생성될 수 있다. 그림 2는 디지털 홀로그램의 생성 과정을 나타낸다.

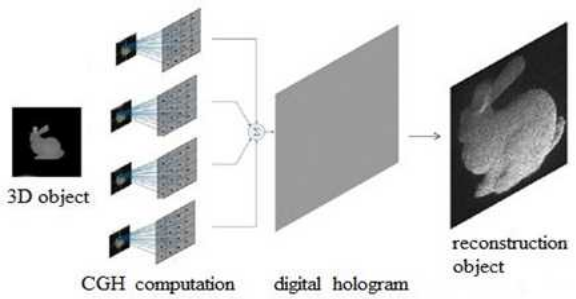


그림 2: 디지털 홀로그램 생성 과정.

그림 2를 보면, 3D 객체의 모든 광원에 대해서 CGH 연산이 이루어지고 있음을 볼 수 있다. 각각의 광원마다 디지털 홀로그램이 생성되고 이것들은 다시 하나의 디지털 홀로그램으로 누적 연산이 된다. 이 과정은 매우 많은 연산시간을 필요로 한다. 예를 들어, 그림 2에서 사용되고 있는 3D 객체는 6982개의 광원으로 구성되어 있는데, 이때 1024×1024pixel 크기의 디지털 홀로그램을 만든다면, 7,321,157,632 번의 CGH 연산을 수행해야 한다. 그래서 이러한 많은 연산량을 줄이기 위해 CGH 연산의 고속화와 최적화 과정이 필요하다.

3. CUDA와 OpenMP를 활용한 CGH 고속 연산

디지털 홀로그램은 일반적으로 CGH 알고리즘에 의해서 생성된다. 다음 수식 (1)은 일반적인 CGH 알고리즘을 나타낸다.

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

I_α는 디지털 홀로그램이고 A_j는 3D 객체의 깊이정보를 나타낸다. α와 j는 홀로그램과 3차원 객체의 인덱스를 나타내고 k는 참조파의 파수(wave number)로 $2\pi/\lambda$로 정의된다. p는 홀로그램의 화소 크기(pixel pitch), x_α 와 y_α 는 홀로그램의 좌표, x_j, y_j, 및 z_j는 3차원 객체의 좌표를 나타낸다. CGH 알고리즘은 그림 2에서 볼 수 있듯이 중복연산이 되는 과정들이 있어서 이 부분들을 병렬처리 할 수 있도록 구조화 할 수 있다. 다음 수식 (2)는 수식 (1)을 효율적으로 병렬처리를 할 수 있게 구조를 변경한 수식이다.

$$I_\alpha = \sum_j^N A_j \cos(2\pi(\theta_z + \theta_H))$$
$$(\theta_z = \frac{Z_J}{\lambda},\ \theta_H = \frac{p^2}{2\lambda z_j}((x_\alpha - x_j)^2 + (y_\alpha - y_j)^2)) \quad (2)$$

본 논문에서는 CUDA를 이용해서 수식 (2)를 구현하였다. 그림 3은 CUDA를 이용한 디지털 홀로그램의 생성을 보여주고 있다. CGH 연산은 3D 객체의 광원에 대해서 디지털 홀로그램의 화소의 크기만큼 수행한다. CUDA는 디지털 홀로그램의 전체 화소를 쓰레드의 그룹 단위(Grid, Block)로 묶어서 각 그룹에 대해 병렬적으로 연산을 수행한다. 본 논문에서는 CUDA에 OpenMP를 적용하여 단일 GPU가 아닌 다중 GPU 기반의 시스템을 구축하여 CGH연산을 수행하였다.

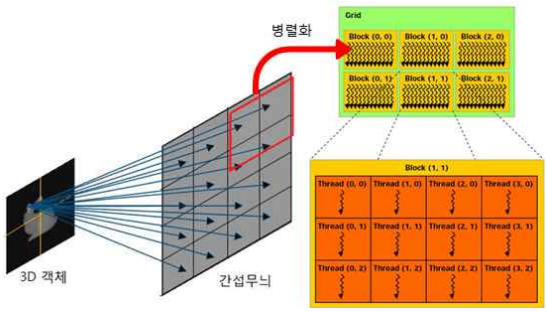


그림 3: CUDA 기반의 CGH 연산.

CUDA는 원리적으로 다중 GPU에 대해 다중 커널 연산을 지원하지 않는다. 그러나 OpenMP의 다중 쓰레드를 사용하면 다중 GPU를 기반으로 다중 커널 연산을 수행할 수 있다. 그림 4는 CUDA와 OpenMP를 이용하여 디지털 홀로그램을 생성하는 과정을 보여주고 있다.

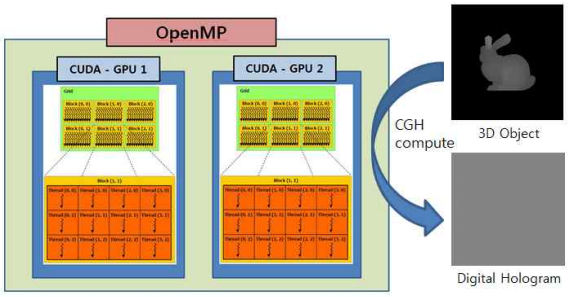


그림 4: CUDA, OpenMP 기반의 CGH 연산.

다중 커널 기반의 CGH 연산은 단일 커널 기반의 CGH 연산보다

더 빠르게 디지털 홀로그램을 생성할 수 있다. OpenMP는 CPU의 코어 개수만큼 쓰레드를 생성할 수 있기 때문에 그 만큼 GPU를 추가적으로 늘린다면 더 많은 커널을 활용할 수 있다. 본 논문에서는 2개의 GPU를 활용하여 멀티 커널 기반의 CGH 연산을 수행하였다.

4. CGH 고속 연산을 위한 최적화

CGH 알고리즘을 이용해서 디지털 홀로그램을 생성하는 과정은 원리적으로 많은 데이터를 연산해야하는 문제가 있다. 그래서 본 논문에서는 디지털 홀로그램을 고속으로 생성하기 위해 상수화, 벡터화, 루프 풀기와 같은 최적화 기법들을 제안한다.

상수화 기법은 CGH 연산 수식을 구성하고 있는 수학적인 모델기반의 광학계 파라미터들을 사전에 미리 계산하여 상수로 고정시키는 기법이다. CGH 수식인 (2)와 같은 경우 코사인, 제곱, 나눗셈 등과 같이 연산량이 많은 요소들로 구성되어 있다. 이러한 요소들을 줄일 수 있다면 전체적인 CGH 연산시간을 효과적으로 줄일 수 있다. 다음 수식 (3)은 상수화 기법을 나타낸다.

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

$$
\left(A=\frac{2\pi}{\lambda},\quad B=\frac{2\pi p^2}{2\lambda}\right) \qquad \text{A,B = Constants.}
$$

수식 (3)을 보면 기존의 CGH 수식에서 중복 연산되는 부분인 A와 B를 사전에 미리 계산하여 상수로 고정시키는 것을 볼 수가 있다. 기존 CGH 수식은 곱셈 연산이 6개인 반면, 상수화 기법을 적용한 후에는 곱셈 연산이 2개로 줄어있음을 확인할 수 있다. 3D 객체의 모든 광원에 대해서 디지털 홀로그램의 해상도 크기만큼 CGH 연산을 수행하기 때문에 CGH 수식의 연산 복잡도를 줄인다면, 디지털 홀로그램을 생성하는데 필요한 전체적인 연산시간을 줄일 수 있다.

벡터화 기법은 디지털 홀로그램을 생성할 때 사용되는 다양한 타입의 변수들을 CUDA를 기반으로 한 float 타입의 벡터 변수로 대체하는 기법이다[3]. CUDA는 구조적으로 floating point 연산에 최적화 되어있다. 본 논문에서는 3D 객체의 광원 정보들을 저장하기 위해서 CUDA에서 지원하는 float 타입의 float4 벡터 변수를 사용한다. 이 기법은 CUDA의 커널내의 CGH 연산시간을 효과적으로 줄일 수 있다. 그림 5는 float4 벡터 변수를 어떻게 사용하는지 보여주고 있다.

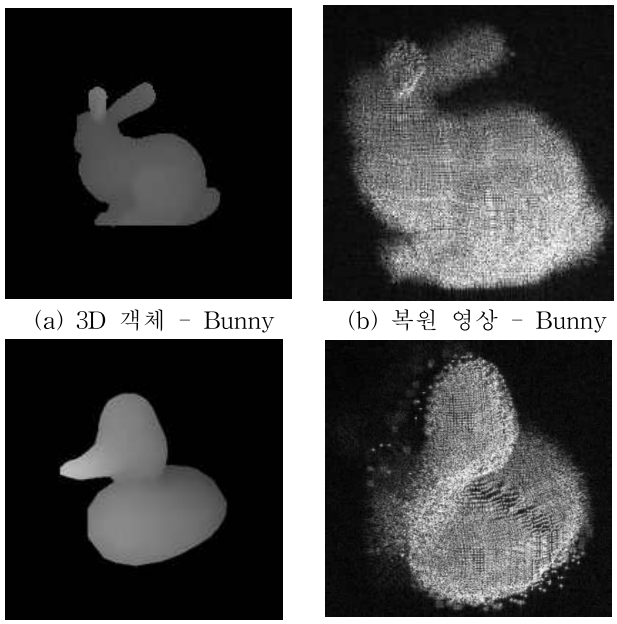
루프 풀기는 대표적인 프로그램 최적화 기법으로 널리 알려져 있다. 루프 풀기의 목적은 루프의 반복 횟수를 줄이고 그 만큼 루프 반복에 필요한 비교문을 생략시켜 연산속도를 증대시키는데 있다. 본 논문에서는 CGH 연산에 사용되는 루프를 풀어줌으로써 전체적으로 디지털 홀로그램의 생성시간을 효과적으로 줄이고 있다.

```
1  float4 *DATA = (float4*)malloc(SolNumber*sizeof(float4));
2
3  int k=0;
4
5  for( i = 0 ; i < DEPTH_HEIGHT; i++)
6      for( j = 0 ; j < DEPTH_WIDTH; j++)
7      {
8          if(ptr[i*DEPTH_WIDTH+j] == 0) continue;
9
10         DATA[k] = make_float4(i,j,ptr[i*DEPTH_WIDTH+j],0);
11         k++;
12     }
```

그림 5: float4 변수

5. 실험 및 결과

본 논문에서는 3장에서 제안한 수식 (3)을 이용하여 디지털 홀로그램을 생성하였다. 본 논문에서는 그림 6의 Bunny와 Duck과 같은 3D 객체영상들을 실험영상으로 사용하였다.



(a) 3D 객체 - Bunny (b) 복원 영상 - Bunny

(c) 3D 객체 - Duck (d) 복원 영상 - Duck

그림 6: 실험 영상

실험에 사용된 3D 객체들의 크기 200×200×8bit이며, 광원은 Bunny가 6,982개, Duck이 5,830개를 갖고 있다. 생성되는 홀로그램 해상도는 1024×1024pixel 이며, Reference wave length(λ)는 633mm, Reconstruction distance는 100nm, Pixel pitch(p)는 10.4 um×10.4 um로 파라미터를 설정하였다. 실험환경은 표1 와 같다.

표1. 실험환경.

항목	세부사항
CPU	Intel(R) Core Quad 2.4GHz
GPU	Nvidia Geforce GTX 580, 2개
OS	MS window 7
RAM	4.0GB
Compiler	MS Visual C++

실험 결과는 다음 표 2와 표 3과 같다. GPU를 사용하지 않고 최적화가 되어있지 않은 환경에서 CGH 연산 시간은 Bunny에서는 1,209,234ms, Duck에서는 1,008,134ms 이고, GPU를 사용하면서 본 논문에서 제안한 최적화 기법을 적용한 CGH 연산은 Bunny에서는 145ms, Duck에서는 122ms가 나왔다.

표2 : Bunny 영상의 CGH 연산 시간

	항목	연산 시간 [ms]
1	CPU	1,209,234
2	CUDA	2,614
3	CUDA, OpenMP	1,110
4	3 + 최적화 기법	145
연산 속도비 1 vs 4		1 : 8,339

표3 : Duck 영상의 CGH 연산 시간

	항목	연산 시간 [ms]
1	CPU	1,008,134
2	CUDA	2,175
3	CUDA, OpenMP	923
4	3 + 최적화 기법	122
연산 속도비 1 vs 4		1 : 8,263

6. 결론

본 논문에서는 디지털 홀로그램을 고속으로 생성하기위해 CGH연산을 CUDA와 OpenMP 기반에서 구현하였고, CGH 연산량을 줄이기 위해서 상수화, 벡터화, 루프 풀기와 같은 최적화 기법들을 제안하였다. 실험결과, 본 논문에서 제안하는 기법은 기존의 CPU기반의 기법보다 약 8,300배 정도 빠르게 디지털 홀로그램을 생성할 수 있는 것을 볼 수 있었다.

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

감사의 글

본 연구는 지식경제부 및 정보통신연구진흥원의 IT 산업원천기술 개발사업의 일환으로 수행하였음.[2009-F-208-01, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호처리 요소 기술 및 SoC 개발].

참고문헌

[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 wit binary masks”, Applied Optics, volume 5, pp. 967-969, June 1966.

[3] T. Shimobaba, T. Ito, N. Masuda, Y. Ichihashi and N. Takada. ”Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL”, Applied Physics. Optics, volume 18, pp. 9955-9960, February 2010.

[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 SPIEP practical HolographyXXI, 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/.