

일반논문(Regular Paper)

방송공학회논문지 제22권 제6호, 2017년 11월 (JBE Vol. 22, No. 6, November 2017)

<https://doi.org/10.5909/JBE.2017.22.6.800>

ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

GPGPU를 이용한 홀로그램 생성 가속화 방법

이 윤 혁^{a)}, 김 동 욱^{a)}, 서 영 호^{b)†}

Hologram Generation Acceleration Method Using GPGPU

Yoon-Hyuk Lee^{a)}, Dong-Wook Kim^{a)}, and Young-Ho Seo^{b)†}

요 약

컴퓨터를 이용하여 홀로그램을 생성하기 위해서는 방대한 양의 계산이 필요하다. 이를 고속화하기 위해 GPGPU(General Purpose computing on Graphic Process Unit)를 이용하여 병렬 프로그래밍을 통한 고속화 방법들이 많이 연구되었다. 본 논문에서는 홀로그램 화소 기반의 병렬처리에서 생기는 병목현상을 줄이고, 공통항을 이용한 가속화 방법을 제안한다. 또한 최적의 스레드를 결정하기 위해 nVidia사의 CUDA와 함께 제공되는 Visual Profiler를 이용한 최적화 방법을 소개한다. 구현 결과 기존 연구 대비 최대 40%의 계산시간을 줄일 수 있었다.

Abstract

A large amount of computation is required to generate a hologram using a computer. In order to accelerate the computation, many methods of acceleration by parallel programming using GPGPU(General Purpose computing on Graphic Process Unit) have been researched. In this paper, we propose a method of reducing the bottleneck caused by hologram pixel based parallel processing and using the shareable variables. We also propose how to optimize using Visual Profiler supported by nVidia's CUDA to make threads work optimally. The experimental results show that the proposed method reduces the calculation time by up to 40% compared with the existing research.

Keyword : Computer generated hologram, GPGPU, Parallel processing

1. 서 론

a) 광운대학교 전자재료공학과(Dept. of Electronic Materials Engineering, Kwangwoon University)

b) 광운대학교 인제니움학부대학(Ingenium College of liberal arts, Kwangwoon University)

† Corresponding Author : 서영호(Young-Ho Seo)
E-mail: yhseo@kw.ac.kr
Tel: +82-2-940-8362

ORCID: <http://orcid.org/0000-0003-1046-395X>

※이 논문은 2014년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2014R1A2A1A11052433).

· Manuscript received August 28, 2017; Revised October 10, 2017; Accepted, October 10, 2017.

컴퓨터 생성 홀로그램(Computer Generated Hologram, CGH)를 이용하여 하나의 홀로그램을 생성하기 위해서는 많은 연산량이 필요하다^[1]. 고속의 홀로그램 생성을 위한 연구는 크게 두 가지로 분류되는데, 그 중 하나로는 FPGA를 이용하여 하드웨어로 구현한 것이고^[2-6], 다른 한 가지는 GPGPU(General Purpose computing on Graphic Process

Copyright © 2017 Korean Institute of Broadcast and Media Engineers. All rights reserved.

“This is an Open-Access article distributed under the terms of the Creative Commons BY-NC-ND (<http://creativecommons.org/licenses/by-nc-nd/3.0>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited and not altered.”

Unit)를 이용하여 병렬 프로그램을 이용하는 것이다^[7~12].

FPGA(Field Programmable Gate Array)를 이용한 방법으로는 CGH를 연산하기 위한 전용 연산 시스템인 HORN-6가 연구되었다^[3]. 100% 파이프라인 구조를 기반으로 하는 CGH 프로세서에 대한 연구도 진행되었다^[4]. 이 연구에서는 프레넬 변환을 수행하기 위한 CGH 셀의 하드웨어 구조를 제안한 후에 이를 확장하여 CGH 커널을 구성하였고, 이를 다시 확장하여 CGH 프로세서를 구현하였다. [4]의 하드웨어는 [2]보다 최대 87.32%의 높은 성능을 갖는다. [5]의 논문에서 제안된 FPGA 기반의 하드웨어는 $1,920 \times 1,080$ 크기의 HD급의 홀로그램을 실시간으로 생성할 수 있다. [6]의 논문에서는 메모리 접근을 줄일 수 있는 하드웨어 구조 및 동작 방식을 제안하여 [5]에서 제안된 하드웨어에 비해서 메모리 접근 횟수를 약 2,000배 감소 시켰다.

GPU를 이용한 방법으로 싱가포르^[9]는 CGH 수식을 복소 형태로 변환한 후에 연산을 분리하는 알고리즘을 제안하였다. 분리된 항을 각각 룩업 테이블 (look-up table, LUT)로 만든 후에 연산을 고속화시키고, GPU로 구현하였다. 1K개의 광원으로 구성된 객체에 대해 $1,024 \times 768$ 크기의 홀로그램을 0.3초당 한 장씩 생성할 수 있었다. 중국 Zhongshan 대학의 Wang^[10]은 GPU를 이용하여 3D 메쉬 모델 기반의 CGH를 생성하였다. 또한 일본 Chiba대의 Shimobaba^[11]는 AMD사의 GPU 기반의 GPU 프로그래밍 기법을 활용하여 CGH 생성을 고속화하였고, HD크기의 홀로그램을 0.31초당 한 장씩 생성할 수 있었다. 광운대는 광원 및 홀로그램의 블록화를 통해서 GPU를 최적으로 이용할 수 있는 연구와 GPU 내부의 다양한 메모리의 사용에 따른 CGH 생성의 효율성을 분석하여 성능을 향상시키고자 시도하였다^[12]. 최근에는 OpenMP 기반의 CPU 병렬화 기법을 이용하여 다수의 GPU를 병렬적으로 동작시켜서 CGH의 성능을 향상시키는 연구들이 수행되어 오고 있다. 한양대^[13]에서는 OpenMP를 활용한 두 개의 GPU를 이용하여 병렬 동작의 최적화 방법을 제안하여 CPU대비 9,700배의 성능 개선을 할 수 있었다. 광운대^[14]에서는 외부 확장 장치를 이용하여 4개의 고성능 GPU를 병렬화하여 컬러 홀로그램을 고속으로 생성하는 시스템을 제안하고 구현하였다. 광운대^[16]는 메모리에 따른 최적화된 방법과 스레드당 계산하는 홀로그램 화소의 개수에 따른 최적화 방법에 대

하여 가이드라인을 제시하였다.

본 논문에서는 홀로그램 화소 기반의 병렬처리에서 생기는 병목현상을 줄이고, 공통항을 이용하여 가속화하고 이를 최적화하기 위한 방법을 제안한다. 본 논문은 2장에서 병렬화 방법에 대하여 설명하고 3장에서 제안하는 방법에 대하여 설명하고, 4장에서 구현결과와 5장에서 결론으로 마무리한다.

II. 컴퓨터 생성 홀로그램의 병렬화

컴퓨터 생성 홀로그램은 모든 홀로그램 평면의 좌표에서 입력되는 모든 광원에 대하여 홀로그램의 밝기를 구하고 누적하여 얻는다. GPGPU(General Purpose computing on Graphic Process Unit)를 이용한 병렬처리를 방법은 그림 1에 나타냈다. 그림 1(a)는 광원기반의 병렬처리로, 병렬처리 단위인 스레드에서 하나의 광원정보를 이용하여 여러 개의 홀로그램 좌표에서의 밝기를 구하는 방법이다. 이 방

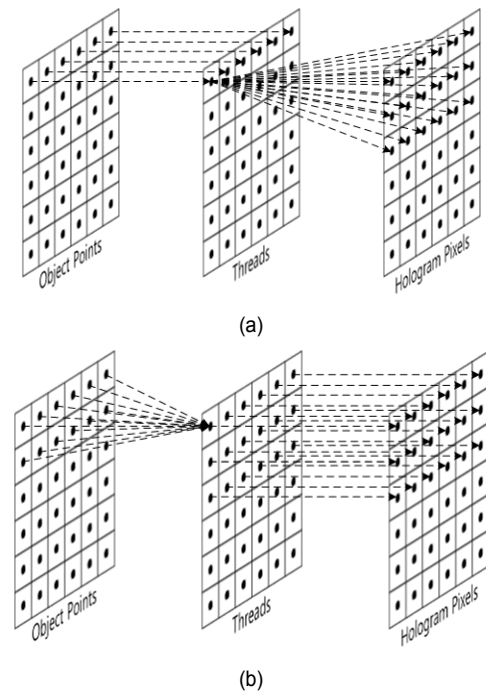


그림 1. 컴퓨터 생성 홀로그램의 병렬처리 방법
Fig. 1. Parallel processing method of computer-generated hologram

법은 독립적으로 수행되는 모든 스트레드에서 계산이 완료된 후 같은 화소에 대한 결과를 다시 누적해야 하는 과정이 추가적으로 필요하다. 반면 그림 1(b)는 홀로그래프 화소 기반의 병렬처리로, 스트레드에서 여러 광원정보를 이용하여 하나의 홀로그래프 좌표에서의 밝기를 구하는 방법이다. 이 방법은 하나의 스트레드에서 모든 광원을 이용하여 계산하기 때문에 계산과 동시에 누적하는 과정이 포함되어 광원 기반의 계산 방법에 비하여 유리하다.

식 1에 컴퓨터 생성 홀로그래프 수식을 나타내었다. $I(u,v)$ 는 홀로그래프 평면의 좌표 u,v 에서의 밝기 이고, $A_j(x,y,z)$ 는 x,y,z 좌표에서의 광원의 밝기 이다. N 은 광원의 개수 이고, λ 는 홀로그래프 생성에 사용한 참조파의 파장이고, p 는 홀로그래프와 광원의 화소의 크기이다.

$$I(u,v) = \sum_{j=1}^N A_j(x,y,z) \cos \left[2\pi \left\{ \frac{z}{\lambda} + \frac{p^2}{2\lambda z} (u-x)^2 + \frac{p^2}{2\lambda z} (v-y)^2 \right\} \right] \quad (1)$$

식 1에서 $\frac{z}{\lambda} + \frac{p^2}{2\lambda z} (u-x)^2$ 는 광원정보 A_j 와 홀로그래프의 u 좌표에 대한 계산으로 같은 u 좌표 간 서로 공유가 가능하고, $\frac{p^2}{2\lambda z} (v-y)^2$ 는 같은 v 좌표 간 공유가능하다. 공유

가능한 항들을 θ_{xz}, θ_y 로 표현하여 식 (2)와 같이 표현할 수 있다.

$$I(u,v) = \sum_{j=1}^N A_j(x,y,z) \cos [2\pi(\theta_{xz} + \theta_y)] \quad (2)$$

III. 제안한 가속화 방법

본 논문에서는 GPGPU를 이용하여 식 (2)를 구현하기 위해 가로세로가 N_t 개, 즉 $N_t \times N_t$ 개의 스트레드로 하나의 블록을 구성하고 스트레드당 모든 광원을 입력 받아 홀로그래프 화소 하나를 생성하는 홀로그래프 화소 기반의 병렬처리 기법을 이용한다. 가속화하기 위해 GPU의 L1 캐시 내에 존재하는 공유 메모리를 이용하여 2가지 단계로 구성하여 구현하고, nVidia사의 Visual Profiler를 이용하여 최적화 하였다.

1. 광원 입력을 위한 메모리 접근

홀로그래프 화소 기반의 병렬처리 기법은 모든 스트레드에서 모든 광원에 대하여 연산하는데, 광원 정보는 호스트 프로

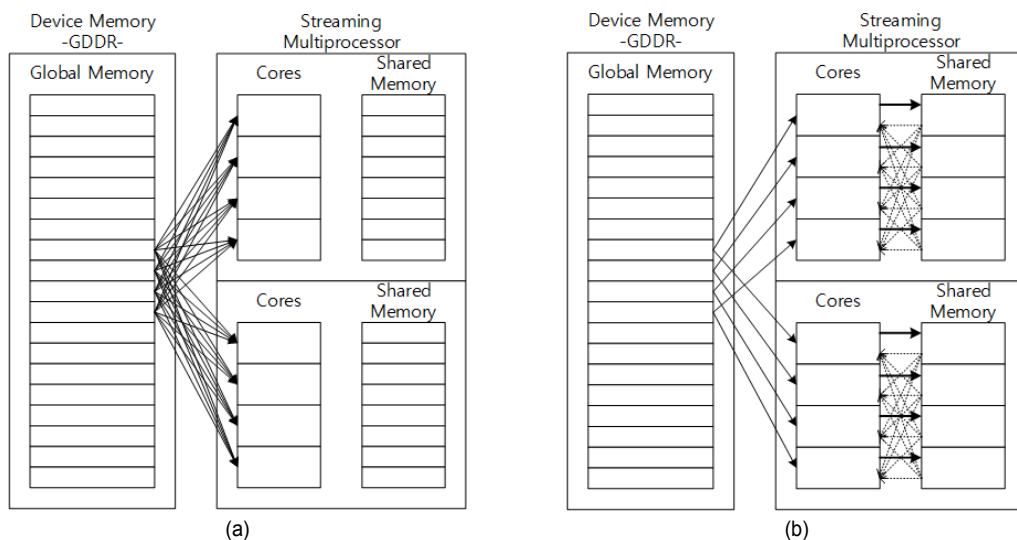


그림 2. 메모리 접근 방법 : (a)글로벌 메모리만 이용할 경우, (b)공유메모리를 사용할 경우

Fig. 2. Memory access method: (a) using only global memory, (b) using shared memory

세서에서 GPU의 외부 메모리에 존재하는 전역 메모리(Global Memory)에 저장한다. 각 스레드에서 모든 광원에 접근하기 때문에 전역 메모리와 GPU 코어 간의 메모리 접근에서 병목현상이 발생한다. 병목 현상을 줄이기 위해 스트리밍 프로세서(Streaming Processor, SM)내 L1 캐시 영역에 존재하는 공유메모리를 이용하는데 이 메모리는 같은 블록 내에서 서로 공유 가능하다.

이를 구현하기 위해 한 블록 내에서 스레드의 행의 좌표가 0인 스레드($\text{threadIdx.x} = 0$)에서 전역 메모리에 접근하여 이를 공유 메모리로 전송한다. 즉, N_t 개 광원정보를 공유메모리에 잠시 저장한 뒤 이를 이용하여 $N_t \times N_t$ 의 홀로그램을 구한 뒤 다시 N_t 개씩 불러와 반복 수행한다. 이 경우 그림 2와 같이 전역 메모리와 코어간의 메모리 접근을 공유 메모리 접근으로 이전하여 병목 현상을 줄일 수 있다.

표 1은 Visual Profiler를 통하여 메모리 접근에 따른 메모리 효율을 나타내었다. 공유 메모리를 사용할 경우 입력 효율이 7.8%에서 100%로 상승한 것을 볼 수 있다.

표 1. Visual Profiler를 이용한 메모리 효율 측정

Table 1. Memory efficiency measurement with Visual Profiler

Efficiency	Fig2.(a)	Fig2.(b)
Global Load Efficiency	7.8 %	100 %
Global Store Efficiency	100 %	100 %
Shared Efficiency	n/a	41.7%

2. 공통항을 이용한 가속화

공유메모리에 저장된 N_t 개의 광원 정보를 이용하여 식 (2)의 홀로그램 화소의 u, v 좌표에 따라 서로 공유 가능한 값(θ_{xz}, θ_y)을 계산하여 다시 공유 메모리에 저장한다. 그림 3은 $N_t \times N_t$ 의 스레드에서 광원정보와 스레드의 좌표를 이용하여 공통항을 계산하는 방법을 도식화 하였다. 그림 3(a)는 N_t 개의 광원정보와 각 스레드의 열 좌표(threadIdx.x)를 이용하여 u 좌표의 공통항(θ_{xz})을 계산 방법이고, 그림 3(b)는 행 좌표(threadIdx.y)를 이용하여 v 좌표의 공통항(θ_y)을 계산 방법을 나타내었다. 이 두 가지의 계산이 종료되면 각 스레드의 해당하는 좌표의 공유 메모

리로 저장한다.

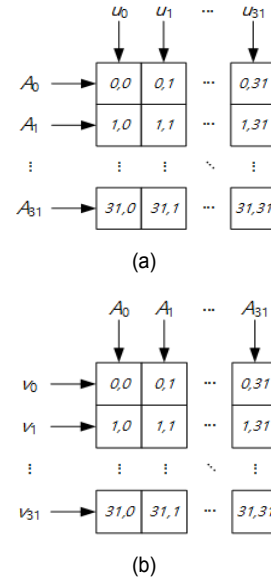


그림 3. 공통항 계산을 위한 스레드 구성

Fig. 3. Thread configuration for common term calculation

다음으로 위에서 미리 구하여 공유메모리에 저장된 공통항과 광원정보를 이용하여 그림 4와 같이 $N_t \times N_t$ 블록의 홀로그램을 계산한다. 각 스레드는 공유메모리에 저장된 u

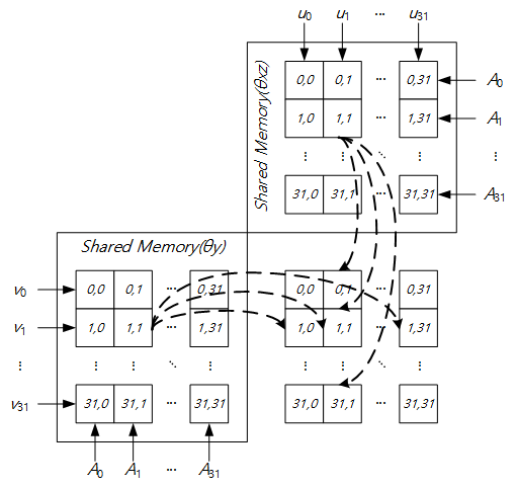


그림 4. 32×32블록의 홀로그램 계산

Fig. 4. Hologram calculation of 32×32 blocks

좌표 해당하는 모든 열의 θ_{xz} 들과 v 좌표에 해당하는 모든 행의 θ_y 를 불러와 식 (2)의 $I(u,v)$ 를 누적하여 구한다. 이러한 과정을 모든 광원에 대하여 반복수행하여 최종 홀로그램을 구한다.

3. Visual Profiler를 이용한 최적화

쓰레드의 개수를 정하기 위해 nVidia사에서 CUDA 툴킷과 함께 제공하는 Visual Profiler를 이용하여 최적화된 N_t 의 수를 구한다. 표 2는 최적의 N_t 의 개수를 정하기 위하여 Visual Profiler를 이용하여 스트리밍 프로세서당 점유율과 메모리 효율을 나타내었다. N_t 가 8일 경우 워프, 레지스터, 공유메모리에 따른 SM당 점유할 수 있는 블록 개수가 GPU (GTX 680)의 제한보다 높아 실제 49.8%의 점유를 한다. 반면 N_t 가 16 또는 32 일 경우 99% 이상의 점유가 가능한 것을 확인 할 수 있다. 하지만 메모리 효율 측면에서 N_t 가 16일 경우 83.3%로 떨어지기 때문에 N_t 가 32일 경우 현재 GPU에서의 최적이 된다.

IV. 구현 결과

구현환경은 Intel사의 i7-3770 CPU를 사용하고 DDR3 16GB의 호스트 메모리를 사용하였다. GPU는 nVidia사의 Geforce GTX680를 사용하였고, GPU는 2GB의 256bit-GDDR5 장치 메모리를 탑재하고 있다. 또한 C/C++과 CUDA 8.0 툴 킷을 이용하여 구현하였다. 또한 테스트를 위해서 SLM을 사용하지 않고 Fresnel 변환을 이용하여 소프트웨어 방식으로 홀로그램을 복원하였다. 이를 위해 표 3와 같은 파라미터들을 이용하여 실험하였다.

표 3. 실험을 위한 파라미터

Table 3. Parameters for experiments

Parameter		Specification
S/W	Distance	100cm
	Hologram Size	1,920×1,080
	Pixel Pitch (p)	6.4 μ m
	Wavelength (λ)	532nm (Green Laser)

표 4는 제안하는 가속화 기법에 따른 계산 속도를 비교하

표 2. Visual Profiler를 이용한 N_t 에 따른 점유율 및 메모리 효율

Table 2. Share and memory efficiency by N_t with Visual Profiler

	Device Limit	$N_t = 8$		$N_t = 16$		$N_t = 32$	
		Achieved	Theoretical	Achieved	Theoretical	Achieved	Theoretical
Occupancy Per SM							
Active Blocks	16		16		8		2
Active Warp	64	31.89	32	63.65	64	63.77	64
Active Threads	2,048		1024		2048		2,048
Occupancy	100%	49.8	50%	99.5	100%	99.6	100%
Warps							
Threads/Block	1,024		64		256		1,024
Warps/Block	32		2		8		32
Block Limit	16		32		8		2
Registers							
Registers/Thread	65,536		32		32		32
Registers/Block	65,536		2,048		8,192		32,768
Block Limit	16		32		8		2
Shared Memory							
Memory/Block	49,152		592		2,208		8,512
Block Limit	16		64		21		5
Efficiency							
Global Load Efficiency			62.5%		83.3%		100%
Global Store Efficiency			100%		100%		100%
Shared Efficiency			40.7%		43.7%		44.5%

였다. 표 4의 4열에 3장 1절에서 설명한 광원에 대한 메모리 접근을 전역메모리에서 공유메모리로 이전하였을 경우 평균 58% 정도 감소하였다. 또한 표 4의 5열에 3장 2절과 3절에서 설명한 공통항 및 Visual Profiler를 이용한 최적화를 적용하였을 경우 평균 63% 정도 감소하였다.

표 4. 가속화 기법에 따른 계산속도 비교

Table 4. Comparing calculation speed according to acceleration method

Source	Valid Pixels	Origin	3.1	3.2 / 3.3
Rabbit	6,123	250.0 ms	103.0 ms	93.0 ms
Sujin	11,250	429.5 ms	206.0 ms	169.0 ms
Billiard	24,297	899.4 ms	407.1 ms	353.2 ms
Baby	15,554	590.4 ms	291.4 ms	229.4 ms

표 5는 기존방법^[16]과 제안하는 방법을 구현하여 비교한

결과이다. 실험에 사용한 영상은 4쌍의 밝기영상과 깊이영상을 이용하였다. 영상에 따라 유효한 광원의 수(밝기가 0이 아닌 화소의 개수)가 다르기 때문에 각 영상별 유효 광원의 수와 기존 방법을 이용하여 구현한 결과, 그리고 제안하는 방법을 이용하여 구현할 때의 계산시간을 나타내었다. 제안하는 방법으로 구현할 때 약 40%의 계산시간을 줄일 수 있다. 그림 5는 실험에 사용한 4개 객체의 영상쌍(밝기

표 5. 입력 영상별 기존 연구와의 비교

Table 5. Comparison with existing research by input image

Source	Valid Pixels	[16]	Proposal Method
Rabbit	6,123	150.7 ms	93.0 ms
Sujin	11,250	305.7 ms	169.0 ms
Billiard	24,297	571.5 ms	353.2 ms
Baby	15,554	382.8 ms	229.4 ms

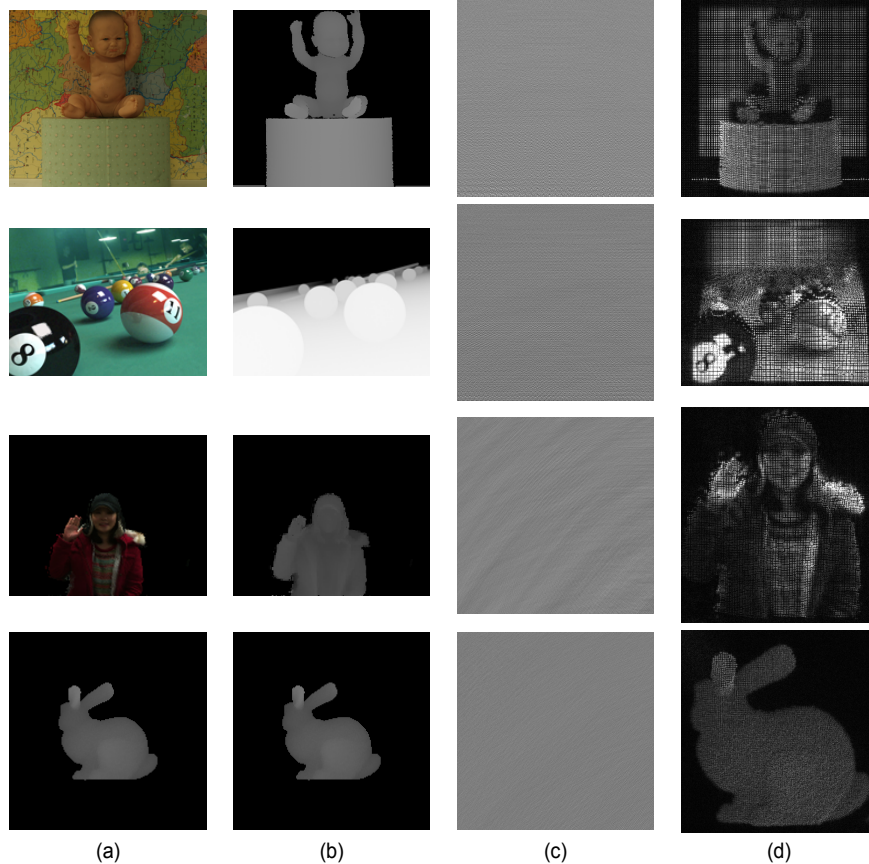


그림 5. 실험에 사용된 (a) 밝기영상, (b) 깊이 영상, (c) 생성된 홀로그램, (d) S/W 복원 영상

Fig. 5. (A) brightness image, (b) depth image, (c) generated hologram, (d) S/W reconstruction image

영상과 깊이영상)과 생성한 홀로그램 및 컴퓨터를 이용한 복원영상을 나타내었다.

V. 결 론

본 논문에서는 공유 메모리를 이용한 홀로그램 화소 기반의 병렬처리 기법에서 생기는 전역 메모리와 코어간 병목현상을 줄이는 방법과 공통항을 이용한 가속화 방법, 그리고 Visual Profiler를 이용한 최적화 방법을 제안하였다. 병목 현상을 줄이는 방법은 행의 좌표가 0인 쓰레드에서 N_t 개의 광원정보를 불러와 공유메모리에 저장하여 이를 이용하여 계산하여 줄였고, $N_t \times N_t$ 개의 쓰레드를 이용하여 N_t 개의 광원정보와 각 행과 열의 좌표에 따른 공통항을 계산하여 이를 블록 내에서 서로 공유하여 각 쓰레드에서 홀로그램 화소를 계산하였다. 또한 최적의 N_t 를 구하기 위하여 Visual Profiler를 이용하였다. 제안하는 방법을 이용할 때 기존 방법에 비하여 약 40%의 속도 개선이 있는 것을 확인하였다.

참 고 문 헌 (References)

- [1] T. Motoki, H. Isono, and I. Yuyama, "Present Status of Three-Dimensional Television Research," Proceedings of the IEEE, Vol.83, No.7, pp.1009-1021, July 1995.
- [2] T. Ito, N. Masuda, K. Yoshimura, A. Shiraki, T. Shimobaba, and T. Sugie, "Special-Purpose computer HORN-5 for a real-time electroholography," Optics Express, Vol.13, No.6, pp.1923-1932, March 2005.
- [3] Y. Ichihashi, H. Nakayama, T. Ito, N. Masuda, T. Shimobaba, A. Shiraki, and T. Sugie, "HORN-6 special-purpose clustered computing system for electroholography," Optics Express, Vol.17, No.16, pp.13895-13903, August 2009.
- [4] Y. H. Seo, H. J. Choi, J. S. Yoo, and D. W. Kim, "An architecture of a high-speed digital hologram generator based on FPGA," Journal of Systems Architecture, Vol.56, No.1, pp.27-37, January 2010.
- [5] Y. H. Seo, H. J. Choi, J. S. Yoo, and D. W. Kim, "A New Parallelizing Algorithm and Cell-based Hardware Architecture for High-speed Generation of Digital Hologram," Journal of Broadcast Engineering, Vol.16, No.1, pp.54-63, January 2011.
- [6] Y. H. Lee, Y. H. Seo, J. S. Yoo, and D. W. Kim, "Hardware architecture of high-performance digital hologram generator on the basis of a pixel-by-pixel calculation scheme," Applied Optics, Vol.51, No.18, pp.4003-4012, June 2012.
- [7] N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, "Computer generated holography using a graphics processing unit," Optics Express, Vol.14, No.2, pp.603-608, January 2006.
- [8] L. Ahrenberg, P. Benzie, M. Magnor, and J. Watson, "Computer generated holography using parallel commodity graphics hardware," Optics Express, Vol.14, No.17, pp.7636-7641, August 2006.
- [9] Y. Pan, X. Xu, S. Solanki, X. Liang, R. Bin, A. Tanjung, C. Tan, and T.-C. Chong, "Fast CGH computation using S-LUT on GPU," Optics Express, Vol.17, No.21, pp.18543-18555, October 2009.
- [10] Y. Z. Liu, J. W. Dong, Y. Y. Pu, B. C. Chen, H. X. He, and H. Z. Wang, "High-speed full analytical holographic computations for true-life scenes," Optics Express, Vol.18, No.4, pp.3345-3351, February 2010.
- [11] T. Shimobaba, T. Ito, N. Masuda, Y. Ichihashi, and N. Takada, "Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL," Optics Express, Vol.18, No.10, pp.9955-9960, May 2010.
- [12] Y. H. Lee, D. W. Kim, and Y. H. Seo, "High-Speed Generation Technique of Digital holographic Contents based on GPGPU," Journal of the Korea Society of Digital Industry and Information Management, Vol.9, No.1, pp.151-163, March 2013.
- [13] J. S. Song, J. S. Park, Y. H. Seo, and J. I. Park, "Fast Generation of Digital Hologram Based on Multi-GPU," Journal of Broadcast Engineering, Vol.16, No.6, pp.1009-1017, November 2011.
- [14] Y. H. Seo, Y. H. Lee, J. M. Goo, Y. Y. Kim, B. R. Kim, and D. W. Kim, "A New System Implementation for Generating Holographic Video using Natural Color Scene," Journal of Broadcast Engineering, Vol.18, No.2, pp.149-158, March 2013.
- [15] Y. H. Lee, Y. H. Seo, and D. W. Kim, "Fast Hologram Generation Method Using Scheduling of Multi-GPGPUs," The Korean Institute of Broadcast and Media Engineers Summer Conference, Jeju, Korea, pp.389-390, 2016.
- [16] Y. H. Seo, Y. H. Lee, and D. W. Kim, "Implementation of Parallel Computer Generated Hologram Using Multi-GPGPU," Journal of the Korea Institute of Information and Communication Engineering, Vol.18, No.5, pp.1177-1186, May 2014.
- [17] CUDA Toolkit Documentation, <http://docs.nvidia.com/cuda/>
- [18] NVIDIA's Next Generation CUDA Compute Architecture : Kepler GK110, <http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf>

저 자 소 개



이 윤 혁

- 2012년 : 광운대학교 전자재료 공학과 졸업(공학사)
- 2014년 : 광운대학교 일반대학원 졸업(공학석사)
- 2014년 ~ 현재 : 광운대학교 일반대학원 박사과정
- ORCID : <http://orcid.org/0000-0001-7184-6896>
- 주관심분야 : 디지털 홀로그램, SoC 설계



김 동 욱

- 1983년 : 한양대학교 전자공학과 졸업(공학사)
- 1985년 : 한양대학교 전자공학과 졸업(공학석사)
- 1991년 : Georgia 공과대학 전기공학과 졸업(공학박사)
- 1992년 ~ 현재 : 광운대학교 전자재료공학과 교수
- 1996년 ~ 2001년 : 반도체 설계 교육센터(IDEC)광운대지역 운영위원
- 1997년 ~ 2000년 : 대학산업기술자문단 자문위원
- 1998년 ~ 2001년 : IEEE SSCS/EDS Joint Chapter of Korea Section 학술위원장
- 1999년 ~ 2001년 : AP-ASIC 학술위원장
- 2001년 : 대한전자공학회 교육연구위원회 부위원장
- 2005년 ~ 2006년 : 대한전자공학회 협동이사
- 2006년 ~ 2008년 : 3차원영상협회 이사
- 2009년 : 실감미디어산업협회 이사
- 2011년 ~ 2013년 : 한국방송공학회 상임이사
- ORCID : <http://orcid.org/0000-0001-6106-9894>
- 주관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication



서 영 호

- 1992년 : 광운대학교 전자재료 공학과 졸업(공학사)
- 2001년 : 광운대학교 일반대학원 졸업(공학석사)
- 2004년 : 광운대학교 일반대학원 졸업(공학박사)
- 2003년 ~ 2004년 : 한국전기연구소 연구원
- 2005년 ~ 2008년 : 한성대학교 조교수
- 2008년 ~ 현재 : 광운대학교 인제니움학부대학 교수
- ORCID : <http://orcid.org/0000-0003-1046-395X>
- 주관심분야 : 실감미디어, 2D/3D영상 신호처리, 디지털 홀로그램