DOI : 10.5392/JKCA.2011.11.3.009

CUDA를 기반한 볼륨데이터의 집적영상 생성을 위한 고속화 기법

Acceleration Method for Integral Imaging Generation of Volume Data based on CUDA

박찬*, 정지성*, 박재형**, 권기철**, 김남**, 류관희*
충북대학교 정보산업공학과*, 충북대학교 정보통신공학부**

Chan Park(szell@cbnu.ac.kr)*, Ji-Seong Jeong(farland83@cbnu.ac.kr)*,
Jae-Hyeung Park(jh.park@cbnu.ac.kr)**, Ki-Chul Kwon(jshajsha@yahoo.co.kr)**,
Nam Kim(namkim@cbnu.ac.kr)**, Kwan-Hee Yoo(khyoo@cbnu.ac.kr)*

요약

최근 들어, 안경식 3D TV 등장으로 3D 입체 콘텐츠의 활성화가 기대된다. 안경식의 불편함을 해소하기 위해 무안경식 3차원 입체 영상 디스플레이에 대한 연구가 활발히 진행되고 있다. 이 연구에서 렌즈 어레이(lens array)로부터 만들어지는 기초영상(elemental images)을 생성하는 것이 필수적이다. 그러나 렌즈 어레이를 구성하는 렌즈의 개수가 증가함에 따라 기초영상을 생성하는데 많은 시간이 소요되고 있으며, 고용량의 볼륨데이터에 대해서는 더 많은 시간이 소요되고 있다. 본 논문에서는 이러한 문제를 좀 더 효율적으로 개선하기 위해 CUDA 기반의 OpenCL를 사용하여 집적영상을 생성하는 기법을 제시한다. 제안된 방법을 세 종류인 Tesla C1060, Geforce 9800GT와 Quadro FX 3800 그래픽 카드를 갖는 PC 환경에서 실험하였으며, 실험 결과 최근 연구 결과[11] 보다 약 20배 정도 성능 개선이 있었다.

■ 중심어 : | 3D 디스플레이 | 집적영상 | 볼륨 데이터 | GPU | CUDA | OpenCL |

Abstract

Recently, with the advent of stereoscopic 3D TV, the activation of 3D stereoscopic content is expected. Research on 3D auto stereoscopic display has been carried out to relieve discomfort of 3D stereoscopic display. In this research, it is necessary to generate the elemental image from a lens array. As the number of lens in a lens array is increased, it takes a lot of time to generate the elemental image, and it will take more time for a large volume data. In order to improve the problem, in this paper, we propose a method to generate the elemental image by using OpenCL based on CUDA. We perform our proposed method on PC environment with one of Tesla C1060, Geforce 9800GT and Quadro FX 3800 graphics cards. Experimental results show that the proposed method can obtain almost 20 times better performance than recent research result[11].

■ keyword : | 3D Display | Integral Imaging | Volume Data | GPU | CUDA | OpenCL |

* 본 연구는 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(2010-0021853)과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업의 지원을 받아 수행된 연구결과임

접수번호 : #101206-017
접수일자 : 2010년 12월 06일
심사완료일 : 2011년 01월 14일
교신저자 : 류관희, e-mail : khyoo@cbnu.ac.kr

I. 서 론

3D 디스플레이 기술은 일반 디스플레이 장치가 제공하는 2D 평면 영상과 달리 실제로 3D 공간상에 물체가 있는 것과 같이 관찰되도록 3D 입체영상을 표시하는 기술을 말한다. 이는 시각 정보의 수준을 한 차원 높여주는 새로운 개념의 실감 영상 미디어로서 차세대 디스플레이를 주도할 것으로 인지 되고 있으며, 여러 산업계, 학계를 중심으로 활발한 연구가 진행되고 있다.

3D 디스플레이 기술은 입체 영상을 표현하는 방법에 따라 양안 시차 디스플레이 방식, 체적형 디스플레이 방식, 홀로그래피 방식 등으로 나눌 수 있다. 그 중 양안 시차 디스플레이 방식은 특수한 안경의 착용 여부에 따라 안경식과 무안경식으로 나눌 수 있다. 이 방식은 깊이감이 다른 방식들에 비해 크다는 중요한 장점을 갖는 반면 특수 안경을 착용해야 하는 불편함 또는 몇 개의 정해진 위치에서만 3D 영상을 관측해야 하는 단점 등의 문제점들 때문에 차세대 3D 디스플레이 기술로 사용하기 위해 해결해야할 문제점들 많이 있다[1]. 이와 비교하여 집적영상(integral imaging)[2] 3D 디스플레이 기술은 특수 안경을 착용해야 하는 불편함 없이 자연스러운 3D 영상을 관찰자에게 제공 할 수 있다는 장점을 가지고 있어, 완전한 3D 디스플레이 기술로서 많은 발전 가능성을 가지고 있다.

집적영상의 장점은 무안경 방식이며, 일정한 시야각 내에서 연속적인 시점을 제공하며, 수평뿐만 아니라 수직 시차를 제공하며, 기존 2D 디스플레이 시스템을 사용하여 구현할 수 있다는 것이다. 집적영상 시스템은 p [그림 1]과 같이 여러 개의 기초렌즈(elemental lens)들로 구성된 렌즈어레이(lens array)를 이용하여 대상물의 3D 객체를 기초영상(elemental image)의 형태로 저장하고, 그 기초 영상을 다시 렌즈 어레이를 통하여 3D 영상으로 보여주는 시스템이다. [그림 1]은 주어진 3D 객체에 대해 집적 영상 시스템의 개념도를 설명한 것으로, 렌즈어레이로부터 보이는 집적영상을 생성하는 픽업(pick up) 과정을 거쳐 생성된 집적영상은 기초영상 형태로 저장되어 3D 디스플레이 화면(display panel)에 출력된다.

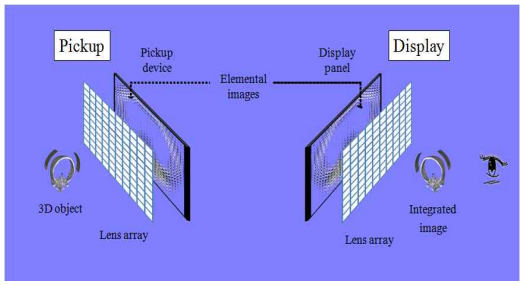


그림 1. 집적영상을 이용한 3D 디스플레이 시스템[9]

이러한 장점으로 인해 무안경식 3D 디스플레이에 대한 연구가 진행되고 있으며, 특히 깊이 영역과 시야각 개선 등의 다양한 시뮬레이션을 처리하기 위해 집적영상을 효율적으로 픽업하기 위한 알고리즘의 개발이 진행되고 있다. 이 과정에서 매우 중요하게 다루는 내용으로는 주어진 3D 객체에 대해 NxN 개의 렌즈로 구성된 렌즈어레이를 통해 기초영상을 얼마나 빠르고 질 높게 만들 수 있는가이다[2].

본 논문에서는 대용량의 볼륨데이터(volume data)에 대해 NxN개의 렌즈들로 구성된 렌즈어레이로부터 집적영상을 효율적으로 생성할 수 있는 기법을 제안한다. 제 II 장에서는 집적영상을 생성하기 위한 선행 연구 결과를 소개하고, 제 III 장에서는 본 논문에서 제안하는 NxN 렌즈들로 구성된 렌즈어레이로부터 집적영상을 효율적으로 생성하기 위한 CUDA(compute unified device architecture)[13] 기반의 API (application programmer's interface)인 OpenCL(open computing language)[14]를 이용하여 처리한 기법을 소개하고, 제 IV 장에서는 실험 및 결과를 제시한다. 마지막으로 제 V 장에서는 연구결과를 요약하고 향후 연구 방향을 제시한다.

II. 선행연구

집적 영상을 생성하기 위한 대표적인 기법으로는 PRR(point retracing rendering)[3], MVR(multiple viewpoint rendering)[4], PGR(parallel group rendering)[5][6], VVR(viewpoint vector rendering) [7]

등이 있으며, 이들 방법을 개선한 다른 방법 등도 있다[8][9].

PRR 방식은 렌즈를 통해서 디스플레이에 상이 맺히는 부분을 하나하나 계산하여 찾는 방법이다. 기초 영상의 모든 픽셀에 대해서 계산이 이뤄져야 하기 때문에 실시간으로 얻어야 하는 부분에서 사용하기에 적합하지 않다. MVR 방식은 컴퓨터 그래픽인 OpenGL(Open Graphics Library)을 사용하여 가상으로 실제와 같은 기초영상을 얻을 수 있는 방식이다. 컴퓨터 그래픽을 이용하기 때문에 간단하면서도, 정확한 기초영상을 얻을 수 있으며, 단일 기초영상의 크기에 전혀 영향을 받지 않는다. 하지만, 렌즈 어레이의 증가에 따른 처리시간이 증가하는 문제점이 있으며, 이는 사용되는 3D 물체의 해상도에도 영향을 받게 된다. PGR 방식은 디스플레이에서 렌즈를 통하는 벡터는 한정되어 있으며, 포커스 모드에서 이 벡터 방향의 이미지를 얻게 되며, 이 이미지는 렌즈를 통해 생성되는 벡터의 개수만큼 만들게 된다. 이 영상에서 각각의 기초영상에 대응하는 정보를 매칭 시키게 되는데, 렌즈의 수가 증가하더라도 처리의 시간적 문제는 생기지 않게 된다. 하지만, 포커스 모드에서만 사용해야 하며, 단일 기초영상의 크기가 증가하게 되면, 이 역시 시간적 문제점이 생기게 된다. VVR 방식은 PGR 방식을 이용하였으며, 렌즈를 통해 생성되는 벡터를 일정하게 그룹화 하여, 벡터 방향의 이미지를 적게 얻어 시간을 단축시킬 수 있다. 하지만, 정밀도를 원하는 부분에서는 왜곡 현상을 겪을 수 있다. 지금까지 소개된 기법은 두 가지 측면에서 한계가 있다. 첫 번째 NxN 렌즈로 구성된 렌즈어레이에서의 N의 개수가 증가함에 따라 집적영상을 생성하는 시간이 N^2배로 증가한다는 것이며, 두 번째로 볼륨데이터와 같은 대량의 3D 객체에 대해서는 집적영상을 생성하는 시간이 길어질 수 있다. 이러한 문제점을 해결하기 위해 Jang, etc[9] 등은 쓰레드(thread)를 이용한 병렬 처리 기법을 제시하였으며, 또한 그들은 볼륨데이터에 대해 옥트리(octree)를 구성하여 GPU(graphics processing unit)[12]로 렌더링 하는 기법을 제시하였다[10]. 하지만 512x512x79 바이트 크기의 볼륨데이터에 대해 10x10개의 렌즈로 구성된 렌즈어레이로부터 집적영상을 생성하는데 약 45.08초가 소요되었다. 본 논문에서는 이러한 한계점을 극복할 수 있는 CUDA 기반 기초영상을 생성하는 기법을 제안하고자 한다.

III. CUDA 기반 집적영상의 생성 고속화 기법

이번 장에서는 볼륨데이터(volume data)에 대해 NxN렌즈들로 구성된 렌즈어레이로부터 기초영상을 CUDA 환경에서 생성하기 위한 기법에 대해 소개한다. [그림 2]는 본 논문에서 제안한 기초영상을 생성하기 위한 전체적인 흐름도를 나타낸다.

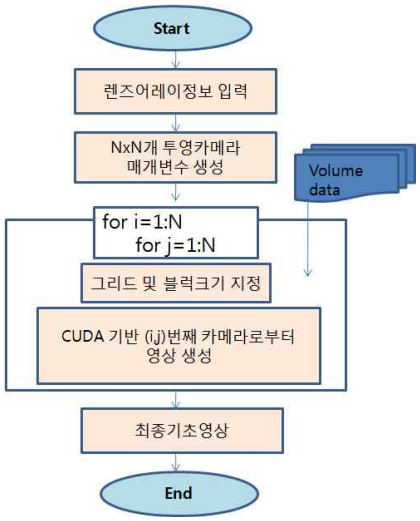
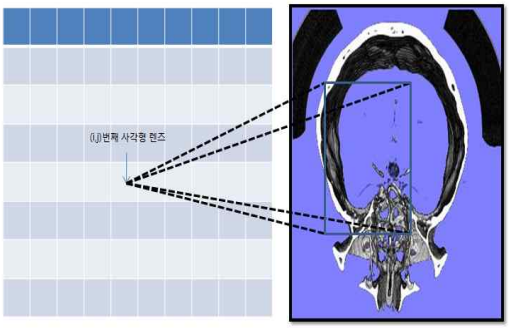


그림 2. 볼륨데이터에 대한 NxN 렌즈로 구성된 렌즈어레이로부터 기초영상 생성 흐름도

Jang, etc[10]에서 논의한 바와 같이, 렌즈어레이를 구성하는 렌즈의 개수(NxN), 기초영상의 너비와 높이, 렌즈의 초점 거리 정보와 렌즈와 물체사이의 거리 정보를 이용하여 렌즈와 상이 매치는 디스플레이와의 거리를 구할 수 있으며, 이들 정보로부터 투영카메라의 위치, 보는 방향과 시야각에 대한 정보를 구한다. [그림 3]에서와 같이 사용자가 [그림 3](a)와 같은 10x10 렌즈어레이, 즉 100개의 렌즈로부터 [그림 3](b)와 같은 볼륨데이터에 대해 보이는 영상을 생성한다. 특정 렌즈로부

터 보이는 영상은 설정된 렌즈 정보로부터 투영카메라를 얻은 후, 이 투영카메라로부터 보이는 볼륨데이터를 구한다. 최종 기초영상은 100개의 투영카메라로부터 얻어진 영상을 합하여 얻어진다.



(a) 10x10 렌즈어레이 (b) 볼륨데이터

그림 3. 10x10렌즈어레이와 볼륨데이터

이제 특정한 투영카메라로부터 보이는 볼륨데이터 영상을 CUDA를 이용하여 효율적으로 계산하는 방법을 논의한다. CUDA의 프로그래밍 모델[13]에서 호스트(host)는 여러 개의 커널을 가질 수 있고, 각 커널은 하나의 그리드(grid)를 배치할 수 있고, 하나의 그리드는 여러 개의 블록(block)을 구성할 수 있다. 마지막 단계에서 하나의 블록은 여러 개의 쓰레드를 할당할 수 있다. 쓰레드 블록은 동기화할 수 있는 쓰레드 그룹으로 하나 이상의 쓰레드 블록은 멀티프로세서(multiprocessor)로 옮겨져 실행될 수 있으며, 국부공유 메모리를 통해 데이터를 효율적으로 공유할 수 있다.

본 논문에서는 CUDA 프로그램을 구현하기 위해 OpenCL을 사용한다. OpenCL은 nVidia GPU 시스템인 CUDA 병렬 계산 구조를 활용할 수 있는 API이다[14]. OpenCL을 이용하는 사용자는 응용 프로그램에서 커널을 설정한 후, 그리드 크기와 블록 크기를 지정하여 원하는 작업에 대한 병렬 처리를 수행할 수 있다.

특정한 카메라로부터 보이는 3D 볼륨데이터에 대한 기초 영상을 구하기 위해 하나의 그리드에 할당된 블록 수와 특정 블록에 설정될 쓰레드의 수를 명시하여야 해야 한다. [그림 4]은 (i, j) 번째 카메라로부터 보이는 볼륨데이터 영상을 생성하기 위해 커널에 의해 수행되는 그리드, 블록과 쓰레드를 설정하는 방법을 보여주고 있다. 본 논문에서는 전체 기초영상의 크기와 렌즈어레이의 크기가 각각 (width, height)와 (N, N)으로 주어질 때, 그리드의 블록 크기를 (width/N, height/N)으로 주었고, 한 블록 내 쓰레드 크기를 (16,16)으로 주었다.

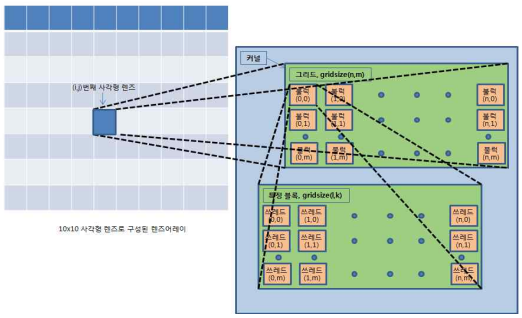


그림 4. (i, j)번째 카메라로부터 보이는 볼륨데이터를 구하기 위한 블록과 쓰레드 설정 과정

위와 같이 설정된 그리드 크기와 블록의 크기에 따라 설정된 (i,j)번째 카메라 정보를 이용하여 볼륨데이터에 대한 보이는 부분을 렌더링 하여, 그 결과는 (i,j)번째에 해당하는 출력 버퍼에 할당된다. 아래의 커널 프로그램 d_kernel이 (i,j)번째 카메라로부터 볼륨데이터를 렌더링 하는 과정이다.

```
__kernel void
d_kernel(outputbuffer, i_cameraNo, j_cameraNo,
imageWidth, imageHeight, cameraMatrix, volumeData)
//outputbuffer: 출력 버퍼
//(i_cameraNo, j_cameraNo): (i,j)번째 카메라
//(imageWidth, imageHeight): 생성될 영상 크기
//cameraMatrix: (i,j)번째 카메라의 변환 행렬
//volumeData: 볼륨데이터
{
  -global_id(0), global_id(1)를 얻어 (i,j)번째
   카메라와 생성될 영상크기 정보를 이용하여
   출력될 버퍼 위치로 설정한다.
  -(i,j)번째 렌더링을 위한 출력버퍼위치설정
  -cameraMatrix로부터 Ray 정보를 얻음
  -얻어진 Ray와 volumeData의 교차점을 구하고
   해당 색깔을 구한다.
  -구해진 색깔 정보를 출력 결과에 저장한다.
}
```

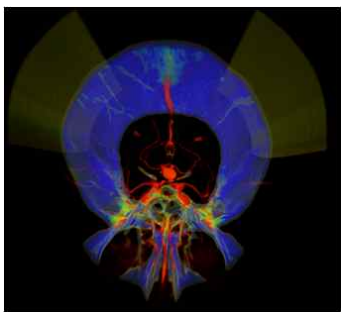
Ⅳ. 실험 및 결과

본 논문에서 제안하는 기법을 구현하기 위해 PC환경에서 개발 도구로 MS Visual Studio 2008를 사용하였고, 3D 그래픽 라이브러리로 OpenGL를, GPU 프로그램을 위한 nVidia의 CUDA 기반 OpenCL을 사용하였다. 성능 실험을 위한 PC로서는 Intel(R) Xeon 2,40GHz CPU와 메인 메모리 4GB를 사용하고 그래픽 카드는 Tesla C1060을 사용하였다. 특히, Tesla C1060의 GPU core 수는 240개 이다. 이러한 PC 시험 환경에서 제안한 방법을 이용하여 집적영상을 생성 성능 평가 실험을 수행하기 위해 [표 1]과 같이 4 가지 볼륨데이터를 이용하였다.

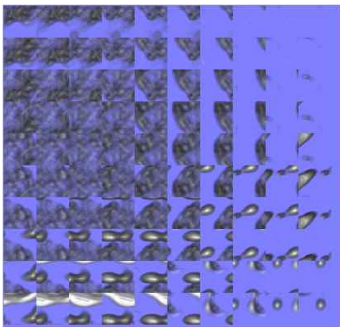
표 1. 성능평가 실험을 위해 사용한 볼륨데이터

번호	데이터크기(바이트)	영상의미
1	32x32x32	Bucky
2	128x128x128	Mummy
3	128x256x256	남자 머리
4	512x512x79	뇌

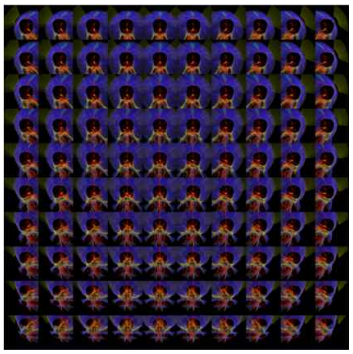
본 논문에서는 제안한 방법과 기존에 제시된 Jang, etc[10]에 의해 제시된 gl3DTexture, GPU와 옥트리를 이용한 기법과의 성능 비교를 수행하였다. 성능 시험 환경은 Tesla C1060그래픽 카드와 512x512x79 크기의 볼륨 데이터로 [그림 5](a)와 같고, GPU와 옥트리 사용 기법과 본 논문에서 제안한 기법을 이용하여 10x10 렌즈어레이로부터 생성한 기초영상은 각각 [그림 5](b)와 (c)와 같다.



(a) 512x512x79 크기의 볼륨데이터



(b) Jang, etc[10]에 의해 생성된 기초영상



(c) 본 논문에서 제안한 기법에 의해 생성된 기초영상

그림 5. 제안한 기법에 따른 기초영상 생성 결과

[표 2]는 사용한 렌즈크기의 렌즈어레이로부터 볼륨데이터에 대한 한 장의 기초영상을 생성한 시간(단위: 초)을 대표적인 세 가지 기법에 따라 나타낸다. 표에서 보는 바와 같이 본 논문에서 제안된 방법이 4x4 렌즈어레이에 대해서는 GPU와 옥트리 사용 기법에 비해 4.94배 빨라졌으며, 10x10과 30x30 렌즈어레이에 대해서는 각각 약 13.89배와 19.53배 빨라졌음을 알 수 있다.

표 2. 제시된 방법에 따른 기초영상 생성 시간(단위: 초) (성능비교는 제안한 기법과 현재까지 가장 빠른 GPU와 옥트리 기법과의 비교임)

기법 / 렌즈크기	gl3DTexture [10]	GPU와 옥트리[10]	제안한 기법	성능 비교
1x1	0.930	0.046	0.063	0.73
4x4	0.880	0.420	0.085	4.94
10x10	5.150	2.390	0.172	13.89
30x30	46.150	22.963	1.176	19.53

본 논문에서는 제안한 기법을 Tesla C1060이외에 Geforce 9800GT와 Quadro FX 3800 그래픽 카드에서 성능 시험을 실시하였다. 성능 시험에 사용된 Tesla C1060는 30개 Multiprocessors와 240개 CUDA Cores를, Quadro FX 3800는 24개 Multiprocessors와 192개 CUDA Cores를, GeForce 9800GT는 14개 Multiprocessors와 112개 CUDA Cores를 사용하였다. [표 1]에 제시된 4종류의 볼륨데이터에 대해 1x1, 5x5, 10x10, 20x20, 30x30, 50x50와 100x100 크기의 렌즈로 구성된 렌즈어레이로부터 기초영상을 생성하는 시험을 수행하였다. [그림 6]이 [표 1]에 나타난 4 종류의 볼륨데이터에 대해 30x30 크기의 렌즈로 구성된 렌즈어레이로부터 보이는 기초영상을 구한 결과를 보여주고 있다.

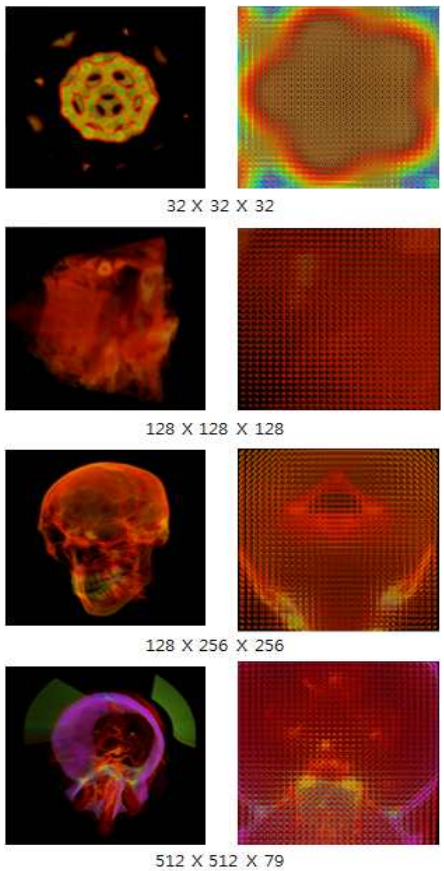


그림 6. 볼륨데이터에 대해 30x30 렌즈어레이로부터 생성한 집적 영상

표 3. 그래픽 카드 종류와 렌즈어레이의 크기와 볼륨데이터 크기에 따른 직접 영상 생성 시간 (단위:초)

GeForce 9800GT				
렌즈크기	32X32X32	128X128X128	128X256X256	512X512X79
1X1	0.094	0.098	0.103	0.102
5X5	0.136	0.147	0.155	0.161
10X10	0.274	0.325	0.366	0.338
20X20	0.904	0.999	0.991	1.144
30X30	1.439	1.610	2.100	1.998
50X50	4.470	5.081	5.452	5.863
100X100	15.699	18.444	23.232	27.817

Quadro FX 3800				
렌즈크기	32X32X32	128X128X128	128X256X256	512X512X79
1X1	0.048	0.062	0.050	0.052
5X5	0.101	0.109	0.113	0.108
10X10	0.253	0.264	0.259	0.320
20X20	0.866	0.865	1.007	1.113
30X30	1.912	2.040	2.256	2.178
50X50	4.937	5.223	5.776	6.030
100X100	20.882	22.231	22.811	21.851

Tesla C1060				
렌즈크기	32X32X32	128X128X128	128X256X256	512X512X79
1X1	0.078	0.062	0.064	0.063
5X5	0.085	0.099	0.101	0.091
10X10	0.144	0.140	0.151	0.172
20X20	0.396	0.412	0.480	0.543
30X30	0.866	0.927	1.048	1.176
50X50	2.486	2.579	3.105	3.684
100X100	8.960	9.698	11.527	11.843

[표 3]은 3종류 그래픽 카드 환경에서 [표 1]의 4 종류의 볼륨데이터에 대해 다양한 크기의 렌즈어레이로부터 기초영상을 생성한 시간(단위: 초)을 나태내고 있다. 시험 결과 Geforce 980GT와 Quadro FX3800 그래픽 카드를 사용하였을 경우, 렌즈의 크기와 볼륨데이터의 크기에 따른 렌즈어레이로부터 직접영상을 생성하는 시간은 거의 유사하게 나타났다. 그에 반해 Tesla C1060 그래픽 카드를 사용하였을 경우, 렌즈의 크기와 볼륨데이터의 크기가 작으면 위 두 종류의 그래픽 카드를 사용하였을 때와 유사한 결과를 얻었다. 그러나 렌즈의 크기와 볼륨데이터의 크기가 증가함에 따라 Tesla C1060 그래픽 카드를 사용하여 렌즈어레이로부터 직접영상을 생성하는 시간은 다른 그래픽 카드를 사용하였을 때 보다 거의 2~3배 정도 차이가 있음을 알 수 있다. 특히, 30x30 크기의 렌즈어레이로부터 512x512x79 볼륨데이터에 대한 기초영상이 Geforce 9800GT와 Quadro FX 3800 그래픽 카드에서는 약 2초 정도 소요되어 생성되었지만, Tesla C1060 그래픽 카드를 사용한 경우에서는 1.176초 소요됨을 알 수 있었다.

이상의 결과를 종합하여 그래프로 표현하면 [그림 7]와 같다. [그림 7]의 그래프에서 가로축은 렌즈의 개수를 나타내고, 세로축은 기초영상을 생성하는데 소요되는 시간(단위:초)을 나타낸다. 그래프에서 보는 바와 같이 세 종류의 그래픽 카드를 사용하여 집적영상을 생성하는 시간은 Tesla C1060이 가장 빠르며, 다음으로 Quadro FX 3800이며, 그 다음으로 Geforce 9800GT 임을 알 수 있다.

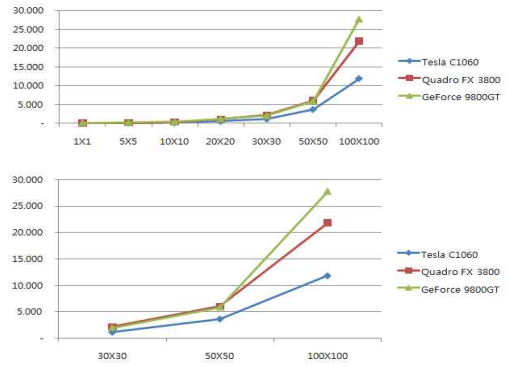


그림 7. 세 종류의 그래픽 카드에 따른 기초 영상 생성 시간 (단위:초)

V. 결론 및 향후 과제

CPU와 GPU 환경에서 NxN 크기의 렌즈어레이로부터 볼륨데이터에 대해 집적영상을 생성할 때, N이 증가함에 따라 집적영상을 효율적으로 생성하기란 불가능하였다[10]. 본 논문에서는 이를 개선하기 위해 CUDA 기반의 OpenCL을 활용하였고, 그 결과 515x512x79 크기의 볼륨데이터에 대해 30x30 크기의 렌즈어레이로부터 하나의 집적영상을 1.176초에 생성할 수 있었다.

그러나 본 논문에서 제안한 기법은 아직도 많은 문제점이 있다. 우선 고해상도 영상을 제작하기 위해서는 렌즈 어레이를 구성하는 렌즈의 수가 많아야 한다. 현재 본 논문의 실험 데이터의 렌즈 어레이 수는 100개로 하였지만 실제 고해상도의 입체 영상을 제작하기 위해서는 적어도 지금의 실험보다는 많은 개수의 렌즈 어레이가 필요하게 되며, 그에 따라 연산 시간이 늘어나게 되어 실시간으로 집적영상을 생성하는데 문제점이 아직도 남아 있다.

향후 연구로서는 렌즈 어레이의 특성상 인접한 카메라간의 매우 유사한 정보가 많으므로 군집화를 통한 인접한 카메라 NxN 연산 처리를 줄여 렌즈 어레이의 수와 상관없이 집적영상을 생성하는 기법이다. 이를 위해서는 군집화 한 렌즈 어레이 간의 정보를 공유하거나 공간 응집도, 시간 응집도를 활용하여 군집화된 렌즈 어레이의 카메라 위치를 산출할 필요가 있다.

참 고 문 헌

[1] "3D Display Technology and Market Forecast Report," *DisplaySearch*, 2010.

[2] G. Lippmann, "La photographie integrale," *C.R Academic Science*. Vol.146, pp.446-451, 1908.

[3] Y. Igarashi, H. Murata, and M. Ueda, "3D Display System using a Computer Generated Integral Photography," *Jpn. J. Appl. Phys.* Vol.17, pp.1683-1684, 1978.

[4] M. Halle, "Multiple Viewpoint Rendering," *SIGGRAPH 98*, pp.243 - 254, 1998.

[5] S. W. Min, "Three-dimensional Image Processing using Integral Imaging Method," *Optical Society of Korea summer Meeting* 2005(7.14~15, 500).

[6] Jang-Il Ser, "A Study on the Properties of an Elemental Image depending on the Shape of Elemental Lens and the pick-up Method in the Integral Imaging," *Journal of Telecommunication and information*, Vol.10, pp.33-39, 2006.

[7] S. W. Min, "Enhanced Image Mapping Algorithm for Computer-Generated Integral Imaging System," *Japanese Journal of Applied Physics*, Vol.45, No.28, pp.L744-L747, 2006.

[8] J. Y. Son, Vladmir V. Saveljev, J. S. Kim,

Sung-Sik, and Bahram Javidi, "Viewing Zones in Three-dimensional Imaging Systems based on Lenticular, Parallax-barrier, and Microlens-array Plates," *Applied Optics*, Vol.43, pp.4985-4992, 2004.

[9] J. I. Ser and S. H. Shin, "Elemental Image Resizing and the Analysis of the Reconstructed Three dimensional Image in the Integral Imaging System," Journal of Korean Optics, Vol.16, No.3, pp.225-233, 2005.

[10] Y. H. Jang, C. Park, J. H. Park, N. Kim, and K. H. Yoo, "Parallel Processing for Integral Imaging Pickup using Mutliple Threads," *International Journal of Contents*, Vol.5, No.4, pp.30-34, 2009.

[11] Y. H. Jang, C. Park, J. S. Jung, J. H. Park, N. Kim, and K. H. Yoo, "Integral Imaging Pickup Method of Bio-Medical Data using GPU and Octree," Journal of Korea Contents, Vol.10, No.9, pp.1-9, 2010.

[12] Fernado, *GPU Gems*, Addison Wesley. 2004.

[13] nVidia, CUDA C programming guide, version 3.1.1, 2010.

[14] nVidia, OpenCL programming guide for the CUDA architecture, version 2.3, 2009.

저 자 소 개

박 찬(Chan Park) 정회원

- 2003년 2월 : 충북대학교 컴퓨터교육과(공학사)
- 2007년 2월 : 충북대학교 컴퓨터교육과(교육학석사)
- 2008년 3월 ~ 현재 : 충북대학교 정보산업공학과 박사과정

<관심분야> : LMS, LCMS, 이러닝, 유러닝, 멀티미디어, 컴퓨터 그래픽스

정 지 성(Ji-Seong Jeong) 준회원

- 2009년 2월 : 충북대학교 컴퓨터교육과(공학사)
- 2009년 3월 ~ 현재 : 충북대학교 정보산업공학과 석사과정

<관심분야> : 컴퓨터 그래픽스, LCMS, 학습추론, 유러닝

박 재 형(Jae-Hyeung Park) 정회원

- 2000년 2월 : 서울대학교 전기공학부(공학사)
- 2002년 2월 : 서울대학교 전기컴퓨터공학부(공학석사)
- 2005년 8월 : 서울대학교 전기컴퓨터공학부(공학박사)
- 2005년 9월 ~ 2007년 8월 : 삼성전자 책임연구원
- 2007년 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 조교수

<관심분야> : 3D 디스플레이, 광정보처리

권 기 철(Ki-Chul Kwon) 정회원

- 1996년 2월 : 상주대학교 전기전자공학과(공학사)
- 2000년 2월 : 충남대학교 전자공학과 (공학석사)
- 2005년 2월 : 충북대학교 정보통신공학과 (공학박사)
- 2002년 ~ 2008년 : 프리즘테크 부설연구소 연구원
- 2008년 ~ 현재 : 충북대학교 연구교수

<관심분야> : 디지털 영상처리 및 의료영상처리, 입체영상 처리 시스템

김 남(Nam Kim) 종신회원

- 1981년 2월 : 연세대학교 전자공학과(공학사)
- 1983년 2월 : 연세대학교 전자공학과(공학석사)
- 1988년 8월 : 연세대학교 전자공학과(공학박사)
- 1992년 8월 ~ 1993년 8월 : 미국 Stanford 대학교 방문교수
- 2000년 3월 ~ 2001년 2월 : 미국 California Technology Institute 방문교수
- 1989년 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 교수

<관심분야> : 광정보처리, 광통신, 이동 통신 및 전파 전파, 마이크로파 전송 선로 해석, EMI/EMC 및 전자파 인체보호 규격

류 관 희(Kwan-Hee Yoo) 종신회원

- 1985년 8월 : 전북대학교 전산통계학과(이학사)
- 1988년 2월 : 한국과학기술원 전산학과(공학학사)
- 1995년 8월 : 한국과학기술원 전산학과(공학박사)
- 1988년 1월 ~ 1997년 8월 : 데이콤 선임연구원
- 2003년 7월 ~ 2005년 2월 : 카네기멜론대학교 로보틱스연구소 교환교수
- 1997년 8월 ~ 현재 : 충북대학교 컴퓨터교육과 및 정보산업공학과 교수

<관심분야> : 컴퓨터그래픽스, 인공치아모델링, 3D게임, 메디컬그래픽스