GPGPU를 이용한 영상 품질 측정 프로그램의 가속화 연구

Research of accelerating method of video quality measurement program using GPGPU

이성욱*, 변기범*, 김기수*, 홍지만**

(Seonguk Lee, Gibeom Byeon, Kisu Kim, Jiman Hong)

요 약

최근 그래픽 처리 장치(GPU)의 발전과 개발자 친화적인 GPGPU(General-Purpose computing on Graphics Processing Units)기술의 발전으로 인해 그래픽 처리 장치를 활용한 병렬 컴퓨팅의 사용이 확대되고 있다. 이를 통해 과학, 의학, 공학 등 많은 분야에 걸쳐 기존 CPU 컴퓨팅 환경보다 더 빠른 처리속도로 결과 값을 얻어 낼 수 있게 되었다. 본 논문은 CPU 기반 컴퓨팅과 GPU 기반 컴퓨팅의 연산처리 속도의 차이의 비교를 위해 기존 CPU 기반으로 구현된 영상 품질 측정 프로그램을 NVIDIA사의 GPGPU기술을 사용할 수 있도록 프로그램을 포팅한다. 포팅한 프로그램을 바탕으로 GPGPU기술을 통한 프로그램의 가속화에 대하여 연구한다. 가속화된 프로그램은 CPU 기반의 프로그램보다 약 1.83배 정도의 실행 속도를 가진다. 또한 CPU 기반의 프로그램을 GPU 기반으로 수정할 때 생기는 제약과 문제점에 대해서도 기술한다.

■ 중심어 : GPGPU; 병렬컴퓨팅; CUDA

Abstract

Recently, parallel computing using GPGPU(General-Purpose computing on Graphics Processing Units) according to the development of the graphics processing unit is expanding. This can be achieved through the processing speeds faster than traditional computing environments across many fields, including science, medicine, engineering, and analysis. However, in using the GPU technology to implement the a parallel program there are many constraints. In this paper, we port a CPU-based program(Video Quality Measurement Program) to use technology. The program ported to GPU-based show about 1.83 times the execution speed than CPU-based program. We study on the acceleration of the GPU-based program. Also we discuss the technical constraints and problems that occur when you modify the CPU to the GPU-based programs.

■ keywords : GPGPU; Parallel computing; CUDA

Ⅰ. 서 론

그래픽 처리 장치(GPU)는 과거 단순 변환기에서부터 현재 CPU 보다 더 뛰어난 성능을 보여주기까지 매년 4배 이상의 성능 향상을 보이고 있다[1]. 이에 맞추어 CPU가 담당하던 연산 기능을 GPU를 활용하여 연산하는 GPGPU(General Purpose computing on Graphics Processing Unit) 기술 또한 개발자가 사용하기 편한 방식으로 발전했다.

GPU의 하드웨어적 특징은 많은 코어를 병렬 처리한 것으로 CPU를 사용하는 방식보다 단순 연산과정을 빠르게 수행 할 수 있다. 오늘날 이러한 GPU의 하드웨어 특징을 GPGPU 기술을 통해 병렬 컴퓨팅 방식으로 활용하여 유체역학, 분자 동력학, 수치 선형대수, 의료영상, 자원 탐사 등 다양한 분야에서 활발하게 사용되고 있다[2].

그래픽카드 제조회사인 NVIDIA사(社)는 자사의 그래픽 처리 장치에 다수에 CUDA(Compute Unified Device Architecture) 코어를 내장시켰다. 이 코어들은 독립적으로 스

* 학생회원, 숭실대학교 컴퓨터학과
** 정회원, 숭실대학교 컴퓨터학부
이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단 - 차세대 정보컴퓨팅 기술개발사업의 지원을 받아 수행된 연구임 (No. 2012M3C4A7032182)

접수일자 : 2016년 08월 17일
수정일자 : 2016년 11월 24일
게재확정일 : 2016년 11월 13일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

레드를 수행하며 코어의 개수가 증가 할수록 많은 스레드들을 동시에 처리하는 것이 가능하며 이러한 하드웨어적 구조를 통해 연산 속도를 향상시켰다. 하지만 단순히 코어가 증가한다고 성능이 효율적으로 향상되는 것은 아니다. 성능 향상에서 더욱 이득을 보려면 효율적인 스레드 처리가 병행 되어야한다[3].

NVIDIA사의 GPGPU기술인 CUDA플랫폼은 이러한 하드웨어적 특성을 개발자가 쉽게 활용 할 수 있도록 C언어를 확장한 CUDA C를 개발 언어로 사용한다. 이는 기존 개발자가 쉽게 적응 할 수 있는 환경이며 CUDA 플랫폼을 통해 개발자는 그래픽 처리 장치의 스레드를 더욱 효율적으로 사용 할 수 있게 프로그램을 설계 할 수 있다[4].

본 논문에서는 CPU기반으로 설계된 영상 품질 측정 프로그램을 NVIDIA사의 CUDA 플랫폼을 통해 GPU기반으로 포팅한 후 두 프로그램의 실행시간의 차이를 평가하여 GPU를 통한 프로그램 실행 고속화에 대하여 연구한다.

본 논문의 구성은 다음과 같다. 2장에서는 GPGPU기술을 이용하여 프로그램 성능을 향상시킨 다른 연구에 대하여 소개하고, 3장에서는 CUDA C를 사용해 프로그램을 포팅 하는 방법과 GPU기반으로 포팅 된 프로그램을 기존 CPU기반 프로그램과 실행 속도를 비교하는 실험을 통해 프로그램 성능 향상률을 평가한다. 마지막으로 4장에서는 본 연구의 결론과 향후 연구방향에 대해 서술하며 끝을 맺는다.

Ⅱ. 관련 연구

1. GPU를 이용한 고속화 연구

GPU를 이용한 고속화 연구는 CPU 성능의 한계극복과 GPU 성능의 향상으로 시작되었다. 게임과 영화의 발전으로 인해 화려한 그래픽을 처리하기 위해 많은 리소스가 필요하게 되었고 GPU는 대규모 파이프라인방식과 멀티코어로 설계되었다. 그렇기 때문에 GPU는 CPU에 비해 덧셈, 뺄셈, 곱셈을 처리하는 산술 연산 장치(ALU)의 개수가 월등히 많다. 이러한 하드웨어 구조적 차이를 바탕으로 GPU를 이용한 대규모의 연산은 CPU를 이용한 연산보다 고속화에 월등히 유리하며 CPU보다 나은 성능을 보여줄 뿐만 아니라 CPU에 비해 저렴한 가격 이라는 이점도 지니고 있다.

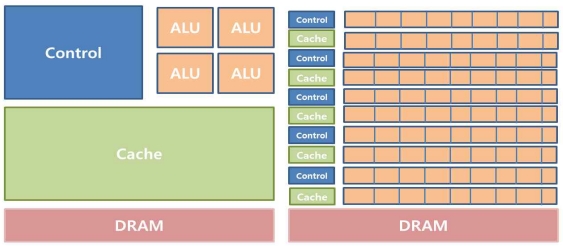


그림 1. CPU, GPU의 하드웨어적 구성 차이

GPU의 성능을 효율적 사용하기 위해 GPGPU 기술이 발전되어 왔다. 현재 프로그램 고속화에 사용되는 기술은 CUDA와 OpenCL이 주류를 이룬다. CUDA는 NVIDIA사에서 개발된 그래픽카드만 사용할 수 있지만 기존 전문가를 위한 그래픽용 API들의 복잡한 사용방법에 비해 일반 프로그래머들도 사용이 간편할 만큼 개발자 친화적이라는 장점이 있다. 또한 그래픽카드의 성능이 향상될수록 CUDA를 이용한 고속화도 향상된다. OpenCL은 개방형 범용 표준 컴퓨팅 프레임 워크로 이기종 플랫폼에서 실행되는 프로그램을 만들 수 있는 도구이다. 어떠한 장비라도 동시에 사용 할 수 있는 장점이 있지만 초기 접근의 어려움과 동시에 사용되는 여러 장비를 관리해야 하는 오버헤드가 존재한다.

2. 영상 분야의 GPU를 이용한 관련 연구

영상 인식 및 분석의 전처리과정에서 사용되는 기술인 Thresholding의 대표적 방법으로 클래스간 분산을 이용하는 Otsu 알고리즘이 있다. 이 알고리즘은 클래스가 늘어날수록 계산시간이 길어지는 단점이 존재한다. 이러한 부분을 해결하기 위해 정용한 등은 Otsu 알고리즘에 GPU프로세싱을 적용하여 고속화를 구현하였다. GPU 프로세싱을 통해 고속화한 알고리즘은 기존 CPU기반대비 5배의 처리속도 향상을 이루었다[5].

의료영상을 판독하기 위해서 중요한 기능 중에 하나인 최대휘소투영은 불륨 렌더링의 한 기법이다. 최대휘소투영 렌더링은 상당히 높은 화질의 영상을 생성하거나 많은 연산을 요구하는 광산 투사법을 이용한다. 계희원 등은 단순한 반복 연산이 필요한 최대휘소투영의 렌더링의 고속화를 위해 GPGPU기술인 CUDA기반의 고속 광선 투사법을 사용하였다. 블록 기반 공간 도약 기법을 적용하여 불필요한 부분은 무시하며, 필요한 부분만을 렌더링한다. 이 같은 이분 이동법을 통해 블록의 경계면 탐색을 고속화 처리하며, 초기 값 추정 알고리즘을 이용하여 이분 이동법의 도약 확률을 향상시킨다. CUDA를 통한 고속화된 기법을 이용해 처리의 최대휘소투영 렌더링의 가시화 속도의 고속화뿐만 아니라, 화질에 대한 손실량을 적게 만들 수 있다. 실험의 결과를 통해 CPU기반 대비 4배의 처리속도 향상을 이루었다[6].

연속 영상에서 물체 추적은 컴퓨터 비전 및 여러 실용적 응용분야에서 이슈가 되는 주제 중 하나이다. 칼만 필터 등의 필터 적용은 물체 추적의 정확도를 높이기 위해서 로보틱스 분야에서 많이 사용되었다. 이 중에서도 파티클 필터는 가장 정확도가 높은 필터이다. 파티클 필터란 여러 입자들을 무작위로 배열, 그 입자들에서 얻어낸 대상의 측정치를 이용, 추측해서 정확도를 높이는 방법을 의미한다. 영상 속에서의 여러 입자 하나하나를 계산하다 보니, 필터 처리에 많은 양의 시간이 소비된다. 김익수

등은 처리시간 고속화를 위해 파티클 필터의 계산에 GPGPU를 이용하여 병렬화 처리를 진행했다. 최신 그래픽 카드에서는 파티클 필터를 위한 적합한 난수 생성을 지원하고 있지 않기 때문에, CPU에서 생성한 난수 변수 값을 GPU에 전달한다. 하지만 많은 난수를 전달해야 하기 때문에 이 과정에서 많은 성능 저하가 발생한다. 이러한 난수 생성을 GPU에서 효율적으로 전달하여 처리하는 것에 대한 연구는 아직도 진행 중이다. 하지만, 이 같은 성능 저하보다도 성능 향상에 대한 이득이 우위를 점하고 있기 때문에 GPU를 통한 고속화는 장점을 가진다[7].

3. 기타 분야의 GPU를 이용한 관련 연구

백은태 등은 국제 표준 블록 알고리즘인 HIGHT를 GPU기반의 CUDA라이브러리를 이용하여 고속화하는 연구를 진행하였다. 자주 사용되는 데이터를 GPU의 공유 메모리로 저장하여 사용하며 전역 메모리에서 데이터를 읽는 작업은 통합접근 기법을 사용하여 최적화하였다. 이러한 작업을 통해 CPU기반 대비 약31배의 고속화를 구현하였다[8].

최근 암호기술분야에서는 효과적인 암호 알고리즘의 구현과 분석을 위해 FPGA(Field-Programmable Gate Array), DSP(Digital Signal Processor)를 이용한 암호 전용 칩을 개발하고 있다. 또한 최근 이미지를 이용한 암호학에 대한 연구가 진행되면서, 이미지처리를 위한GPU 활용에 관한 연구가 활성화 되고 있으며, CPU 이외의 프로세서에 대한 관심이 높아지고 있다. 기존의 범용적인 CPU는 문자열 처리, 단순 그래픽 처리, 다중 사용자 및 프로세스와 같이 범용적이며 복잡한 업무에는 적합하지만 특수한 목적을 가지는 단순한 반복 연산 작업에 집중된 렌더링, 복잡한 이미지처리에는 비효율적이다. 이러한 특수 목적에서 가지는 단순 연산 작업에 GPU의 연산 처리 능력을 활용하는 것은 효율적인 시간 절약이 가능하게 한다. 암호 분야에서 단순 연산 작업이 많이 소비되는 대칭키와 비대칭키 암호의 구현 및 분석에 GPU를 활용하고 있다. 최근에는, GPU 환경에서 RSA(Rivest Shamir Adleman), DSS(Digital Signature Standard), ECC(Elliptic Curve Cryptosystem)와 같이 현재 많이 사용되고 있는 비대칭 키 암호를 구현을 통해 성과를 보여주고 있다[9].

Hilbert R-tree는 데이터베이스분야에서 가장 널리 쓰이는 색인 구조 중 하나이다. 기존 CPU 기반의 Hilbert R-tree는 순차적인 접근방식으로 접근하기 때문에, 대용량의 데이터베이스 색인을 처리 할 경우 전처리에 대한 상당한 오버헤드가 요구된다. GPU를 이용해 순차적인 접근을 병렬화 처리하여, 전처리 오버헤드를 경감시켜줄 수 있다. Hilbert R-tree 전체를 GPU의 메모리상에 구성하며, 해당 MBR(Minimum bounding rectangles)당 차원의 개수만큼의 GPU 스레드를 할당한다. 기존의 GPU를 사용해서 증가하는 단순 병렬처리 속도 향상뿐만 아니라, GPU 내의 메모리 구성 또한 병렬처리에 맞게 최적화하여, 기존 성능에 비해 45배 빠른 Hilbert R-tree 구현이 가능하게 되었다[10].

IT 장비 규모의 증가에 따라, 대규모의 데이터를 저장하는 스토리지 기술과 저장한 데이터들을 효율적으로 분석하며 처리할 수 있는 기술의 필요성이 증대되었다. 이와 따라 스토리지 관리 기술 중 신뢰성 증대와 여러 노드에 데이터를 분산 저장 및 관리하는 분산 파일 시스템을 중심으로 개발이 진행되고 있다. 최근 빅 데이터에 저장 및 관리를 위한 하둡(Hadoop) 플랫폼이 각광받고 있다. 하둡은 맵리듀스 서비스 모델과 분산 파일 시스템인 HDFS(Hadoop Distributed File System)로 구성되어 있는 병렬 처리 플랫폼이다. 병렬 처리 플랫폼으로 다수의 데이터 노드에 비정형 데이터를 보내 처리하며, 이를 통해 데이터간의 분석을 실시한다. 하둡 플랫폼에서 가장 중요한 이슈는 속도이다. 병렬처리라 하더라도, 최소 테라바이트 급부터 페타바이트 급 정도의 빅 데이터를 처리하는데 상당히 시간을 소비하게 된다. 이러한 시간적 문제점을 해결하기 위해, 구본근 등은 각각의 데이터 노드에 GPU를 부착하여 고속화하는 기법을 설계했다[11].

III. 실험 및 성능 평가

1. CUDA C를 이용한 포팅

기존의 CPU기반 프로그램을 CUDA 플랫폼을 통한 GPU기반으로 포팅하는 방식에는 많은 제약이 따른다. 그 제약으로는 CPU가 하는 연산 중 단순 계산 연산만을 GPU를 이용하여 수행할 수 있다는 점과 단순 계산의 경우에도 충분히 방대한 데이터 연산의 처리가 아니라면 PCI버스의 속도한계로 인한 병목현상으로 오히려 GPU를 사용하는 것이 CPU를 사용하는 것보다 느린 현상이 발생한다. 또한 GPU를 사용할 경우에는 병렬 프로그래밍 형식으로 구현된 코드에서 최고의 성능향상률을 보이는데 이는 대부분의 프로그램이 병렬프로그래밍을 생각하며 설계된 것이 아니기 때문에 기존 CPU기반의 프로그램을 포팅하여 엄청난 성능 향상을 기대하기는 힘들다.

그림 2. CPU, GPU 처리시간의 비교

본 논문에서 포팅을 위해 사용한 영상 품질 측정 프로그램은 C언어와 영상처리를 위한 Open Source Library인 OpenCV를 사용해 구현되었다. 처음 구현 설계부터 GPU의 사용을 염두해 두지 않아 병렬 프로그래밍 형식으로 코딩되지 않았으며 OpenCV는 자체적으로 GPU 사용을 지원하는 함수들이 존재하지만 본 논문에서 사용된 프로그램은 GPU 지원을 하지 않는 함수들을 사용하였다. 그렇기 때문에 GPU를 통한 연산을 사용했을 때 프로그램의 성능을 향상시킬 수 있는 부분을 찾아 포팅을 진행하였다. 프로그램의 알고리즘 중 영상의 한 블록단위의 특징 벡터의 품질 기준 모델을 생성하여 왜곡도 측정을 위해 구현된 부분의 연산과정을 GPU를 이용하여 연산이 진행 될 수 있도록 CUDA C를 이용하여 포팅하였다. CUDA C는 GPU와 DRAM, CPU의 호환을 위한 함수들을 C언어의 함수 형태로 지원하기 때문에 기존의 프로그래머가 이질감 없이 코드를 포팅 할 수 있다.

2. 실험 방법 및 구성

본 연구에서는 영상 품질 측정 프로그램의 실행속도를 측정하기 위해 아래 표와 같은 실험 환경을 구성하였다. CUDA 플랫폼을 사용하기 위해 NVIDIA Quadro K620을 사용하였다. Quadro K620 그래픽카드의 CUDA 코어 개수는 384개 이다.

표 1. 실험 환경

CPU	Intel Pentium G3340	RAM	16GB
OS	CentOS 7 64bit	Kernel ver.	3.10
GPU	Nvidia Quadro K620	Cuda Toolkit	7.5

실행속도 측정 실험은 이미 기존에 구현된 CPU기반의 영상 품질 프로그램의 전체 실행 시간과 GPU기반의 프로그램의 전체 실행시간의 비교, CPU 및 GPU기반의 영상 품질 측정 프로그램의 핵심 알고리즘의 실행 시간 비교하는 방식으로 수행되었다. 실행 시간의 측정은 gettimeofday 유닉스 함수를 이용하여 ms(밀리 세컨드)단위 까지 측정하였다. 프로그램 전체 실행시간의 비교는 동일한 영상의 50프레임을 분석하는 시간을 기준으로 측정했으며, 핵심 알고리즘 실행시간의 비교는 1프레임을 분석하는 시간을 기준으로 측정했다. CPU, GPU기반 프로그램 두 버전 모두 각각 5번의 측정 결과의 평균값을 구하여 비교하였다. 성능 개선율 계산은 $(t_1 - t_2)/t_2$ 수식으로 구하였다.(t_1 개선 전 속도, t_2 개선 후 속도)

3. 실험 수행 결과

가. 프로그램 전체 실행시간

CPU기반의 영상 품질 측정 프로그램의 전체 실행 시간평균은 59.66초로 측정되었다. GPU기반의 프로그램의 전체 프로그램 실행 시간 평균은 32.63초로 측정되었다. CPU기반에 대비하여 GPU기반으로 포팅 된 프로그램에서 23초의 고속화를 이루었으며 수치로 나타내면 약 83%의 성능 향상을 기록하였다. 약 1프레임 당 0.46초의 실행 속도가 향상 되었다. 만일 50프레임이 아닌 더 많은 프레임을 분석하여 영상 품질을 측정 하였다면 더 높은 성능 효율을 보였을 것이다.

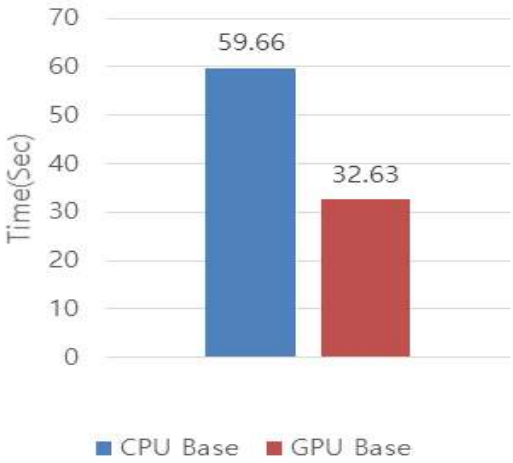


그림 3. 프로그램 전체 평균 실행시간 비교

나. 영상 품질 측정 핵심 알고리즘 실행시간

CPU기반의 영상 품질 측정 프로그램의 핵심 알고리즘의 실행 시간 평균은 1.09초로 측정되었다. GPU기반의 알고리즘의 실행 시간 평균은 0.56초로 측정되었다. CPU기반에 대비하여

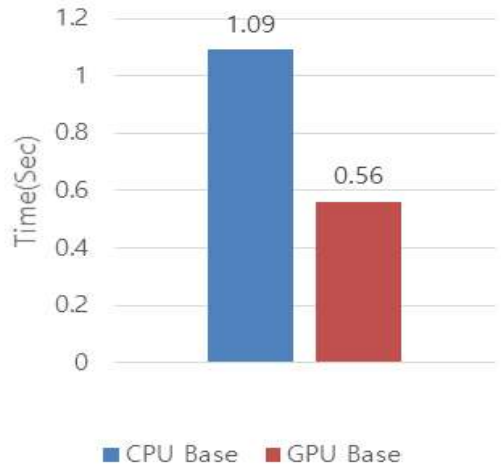


그림 4. 핵심 알고리즘 평균 실행시간 비교

GPU기반으로 포팅 된 알고리즘은 0.53초의 고속화를 이루었으며 수치로 나타내면 약 93%의 성능 향상을 기록하였다.

표 2. 알고리즘 내 핵심 함수 CPU 기반 평균 실행시간

알고리즘 내 핵심 함수명	실행시간(sec)
copydata	0.06
getMSCN	0.10 ~ 0.11
getBlockBasedFeatures	0.52 ~ 0.53
copydata	0.01
getMSCN	0.027 ~ 0.029
getBlockBasedFeatures	0.32 ~ 0.33
calFinalFeature	0.0001

표 3. 알고리즘 내 핵심 함수 GPU 기반 평균 실행시간

알고리즘 내 핵심 함수명	실행시간(sec)
copydata	0.057 ~ 0.06
getMSCN	0.07 ~ 0.08
getBlockBasedFeatures	0.24 ~ 0.25
copydata	0.01
getMSCN	0.023 ~ 0.024
getBlockBasedFeatures	0.09 ~ 0.1
calFinalFeature	0.0001

위의 표는 영상 품질 측정 프로그램 핵심 알고리즘 각 함수들의 CPU, GPU기반 실행시간 결과이다. 알고리즘은 두 번의 왜곡도 측정을 통해 마지막 계산을 하는 구조로 copydata, getMSCN, getBlockBasedFeatures 함수는 두 번씩 실행된다. copydata함수는 영상 프레임을 블록 마다의 정보를 복사하고 getMSCN함수와 getBlockBasedFeatures 두 함수로 각 블록의 왜곡도를 측정 후 calFinalFeature함수를 통해 각 프레임의 품질 값을 구해낸다. GPU를 사용하여 프로그램의 속도 향상을 기대 할 수 있는 부분인 getMSCN, getBlockBasedFeatures 두 함수를 포팅한 결과 첫 반복의 성능 향상율은 각각 45%, 110% 두 번째 반복의 성능 향상률은 각각 17%, 71%를 기록하였다.

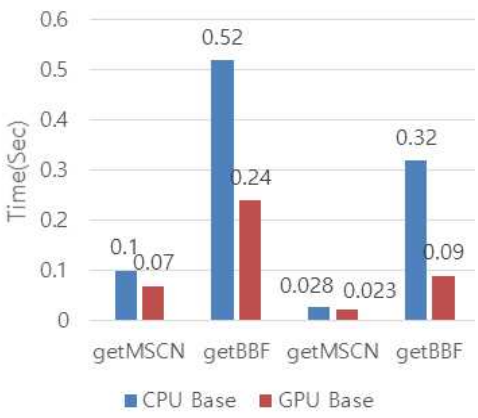


그림 5. 핵심 알고리즘 내 함수 평균 실행시간 비교

4. 성능 평가

CPU기반으로 설계된 프로그램을 GPU기반으로 바꾸는 일은 싱글코어기반으로 설계한 프로그램을 멀티코어기반으로 수정해야하는 일이다. 이러한 일은 프로그램을 초기 설계부터 다시 진행해야 하는 일이라 할 수 있다. 그렇기 때문에 본 연구에서는 GPU를 이용하여 최대의 효율을 낼 수 있는 대량 연산부분만을 타겟팅하여 대규모 반복 연산 부분을 GPU를 사용하여 계산할 수 있도록 포팅하였다. 그 결과 프로그램의 전체 실행시간의 향상률은 약 83%, 영상 품질 측정 핵심 알고리즘의 실행시간 향상률은 약 93%향상을 기록하였다.

기존 CPU기반의 프로그램을 GPU를 사용하여 프로그램의 성능을 최대한 끌어 올렸지만 제약점이 너무 많아 GPU를 사용함에도 불구하고 큰 성능향상은 이끌어 내지는 못하였다.

만약 기존 프로그램을 GPU 환경의 병렬프로그래밍을 염두해 두고 설계하였거나, OpenCV 함수에 종속되지 않게 구현되었다면 프로그램 실행시간 속도 향상률이 더욱 증가되었을 것이다.

IV. 결론 및 향후 연구

본 연구에서는 CPU 컴퓨팅 환경에서 설계된 영상 품질 측정프로그램을 GPU 컴퓨팅 환경으로 포팅하여 성능을 향상시키는 연구를 진행하였다. CPU기반에서의 분기와 분산된 반복부분을 GPU연산의 효율성 증대를 위해 분기를 최소화하고 분산된 반복은 통합하였으며 대규모 연산이 존재하는 부분을 GPU를 사용한 연산으로 수정하여 프로그램 실행시간에 대한 성능향상을 이루었다. CPU기반 대비 GPU기반으로 포팅된 프로그램은 약 1.83배 빠르며 알고리즘 대비 성능 향상률은 그보다 더 높은 수치인 1.93배의 성능 향상을 보였다. 결론으로 기존의 CPU기반으로 설계된 프로그램을 GPU기반으로 포팅 했을 경우 몇 가지의 조건만 충족한다면 고속화 성능 향상률을 보일 수 있는 것이 확인되었다. 조건은 다음과 같다. 첫째, 대량의 반복연산이 수행되는 경우이다. 둘째, 프로그램의 분기가 적은 경우이다. 기본적으로 대량 연산이 존재한다면 GPU를 사용할 경우가 CPU만을 사용하는 경우보다 속도가 빨라진다. 하지만 GPU의 하드웨어적 구조와 GPU의 특징에 대한 이해도가 높다면 더욱 최적화된 결과물을 얻어 낼 수 있을 것이다.

향후 연구로는 CPU기반 프로그램을 GPU기반으로 포팅할 시 포팅이 가능한 연산 부분을 자동화하여 찾아주는 방법의 설계 연구와 연산의 종류와 반복에 따라 GPU를 이용하는 병렬프로그래밍으로 포팅해 주었을 때 얻을 수 있는 속도 향상률에 대한 시뮬레이션 방법의 연구를 진행할 예정이다.

References

[1] M.Macedonia "The GPU enters computing's mainstream", *Computer*, vol.36, no. 10, pp106-108, Oct, 2003.

[2] 김정환, 김진수, "CUDA 기반 GPU에서 효율적인 Power Method의 구현", *한국 컴퓨터 정보학회 논문지*, 제16권, 제2호, 9-16쪽, 2011년 2월

[3] 전형규, 안진우, 김종면, 김철홍. "Memory Delay Comparison between 2D GPU and 3D GPU", *한국 컴퓨터 정보학회 논문지*, 제17권, 제7호, 1-11쪽 ,2012년 7월

[4] Tom R. Halfhill, "Parallel Processing with CUDA," Microprocessor Report, Jan, 2008.

[5] 정용한, 최학남, 박은수, 김준철, 김학일, "GPU를 이용한 고속 Multi-level thresholding 알고리즘", *한국 정보과학회 학술발표 논문집*, 제36권, 제2C호, 354-358쪽, 2009년 11월

[6] 계희원, 김준호. "GPGPU환경에서 최대휘소투영 렌더링의 고속화 방법", *멀티미디어학회논문지*, 제14권, 제8호, 981-991쪽, 2011년 8월

[7] 김익수, 이동익, 정창성, "GPU를 이용한 물체 추적을 위한 색상 기반 파티클 필터", 한국인터넷정보학회 학술발표대회 논문집, 769-774쪽, 2010년 6월

[8] 백은태, 이문규, "HIGHT 블록 암호 알고리즘의 고속화 구현", *정보보호학회논문지*, 제22권, 제3호, 495-504쪽, 2012년 6월

[9] 임지혁, 강정민, 조수민, 김현오, 김동규. "GPU용 라이브러리 CUDA를 이용한 SEED 알고리즘의 고속화 구현", *한국정보과학회 학술발표논문집*, 제37권, 제2B호, 417-421쪽, 2010년 11월

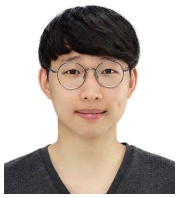
[10] 양시동, 최원익, "GPGPU를 이용한 Hilbert R-tree 벌크로딩 고속화 기법", *정보과학회논문지*, 제41권, 제10호, 792-798쪽, 2014년 10월

[11] 구본근, 최도진. "임베디드 시스템을 이용한 GPU+Hadoop 클러스터 구축", *한국정보기술학회 2014년도 하계종합학술대회*, 236-239쪽, 2014년 5월

저 자 소 개

이성욱(학생회원)

2015년 숭실대학교 컴퓨터학부 학사 졸업.
2016년 숭실대학교 컴퓨터학과 석사 과정.
<주관심분야 : 시스템 소프트웨어, 임베디드 시스템>



변기범(학생회원)

2016년 숭실대학교 컴퓨터학부 학사 졸업.
2016년 숭실대학교 컴퓨터학과 석사 과정.
<주관심분야 : 운영체제, 임베디드 시스템>

김기수(학생회원)

2015년 숭실대학교 컴퓨터학부 학사 졸업.
2016년 숭실대학교 컴퓨터학과 석사 과정.
<주관심분야 : 시스템 소프트웨어, 운영체제>

홍지만(종신회원)

2003년 서울대학교 컴퓨터공학과 박사 졸업
2003년 ~ 2007년 광운대학교 컴퓨터공학부 조교수
2007년 ~ 현재 숭실대학교 컴퓨터학부 교수
2007년 ~ 현재 ACM SIGAPP 회장
<주관심분야 : 운영체제, 시스템소프트웨어 ,임베디드 시스템>