CUDA를 활용한 스케일링 필터 및 트랜스코더의 성능향상

(Performance Enhancement of Scaling Filter and Transcoder using CUDA)

한 재 근 † 고 영 섭 ††
(JaeGeun Han) (YoungSub Ko)
서 성 한 †† 하 순 회 †††
(SungHan Suh) (Soonhoi Ha)

요 약 본 논문은 GPGPU가속을 이용한 스케일링 필터(scaling filter) 및 트랜스코딩(Transcoding)의 성능 향상 방법을 제안한다. 트랜스코딩 기술은 다양한 요구조건을 지닌 멀티미디어 기기에 적합하게 동영상을 가공하는 기술로, 오늘날 여러 분야에서 활용되는 중요한 기술이다. 그러나 트랜스코딩에는 대량의 연산이 필요하기 때문에 기존 트랜스코더(Transcoder) 사용자들은 오랜 처리시간을 감내해야만 했는데, 이는 CPU만을 이용한 트랜스코딩이 충분히 효율적이지 못하기 때문이다. 본 연구에서는 고성능의 연산이 가능한 GPGPU기술을 활용하여, 트랜스코더의 스케일링 필터를 GPU 상에서 높은 병렬성을 가지고 동작하도록 개선함으로써 트랜스코더의 전체적인 성능을 향상시켰다. 개선된 트랜스코더는 다양한 크기의 동영상과 여러 종류의 스케일링 필터 옵션들에 대해 잘 동작함이 검증되었으며, 기본 옵션에서 36%, 최대 101%의 성능향상을 보였다.

키워드 : 트랜스코더, 스케일링 필터, 범용GPU, 쿠다

Abstract In this paper, we propose to enhance the performance of software transcoder by using GPGPU for scaling filters. Video transcoding is a technique that translates a video file to another video file that has a different coding algorithm and/or a different frame size. Its demand increases as more multimedia devices with different specification coexist in our daily life. Since transcoding is computationally intensive, a software transcoder that runs on a CPU takes long processing time. In this paper, we achieve significant speed-up by parallelizing the scaling filter using a GPGPU that can provide significantly large computation power. Through extensive experiments with various video scripts of different size and with various scaling filter options, it is verified that the enhanced transcoder could achieve 36% performance improvement in the default option, and up to 101% in a certain option.

Key words : transcoder, scaling filter, GPGPU, CUDA

1. 서 론

우리는 휴대폰, PMP 등 다양한 멀티미디어 기기의 보편화와 DMB, IPTV 등 디지털 방송의 활성화에 힘입어 일상생활 속에서 멀티미디어 기술을 손쉽게 접할 수 있게 되었다. 이러한 환경에서 동영상을 가공하는 기술은 점차 중요해져 가고 있다.

트랜스코딩은 이러한 요구를 뒷받침할 수 있는 기술로 다양한 요구조건에 맞추어 동영상을 변환한다. 일반적으로 고해상도의 영상을 성능이 낮은 휴대용 기기에 재생할 수 있도록 사용하며, 영상 파일의 전송 및 스트리밍 디지털 방송에 적합하도록 변환하는 데도 사용한다. 따라서 트랜스코딩 기술은 일반 사용자에서 서비스 사업자에 이르기까지 폭넓게 사용되는 기술이다.

트랜스코더는 트랜스코딩 기술을 제공하는 프로그램을 지칭하는 것으로, 트랜스코더는 연산량이 많아 동영상을 변환하는데 긴 시간을 필요로 한다. 이는 트랜스코더 각 단계에서의 높은 데이터 병렬성(data parallelism)을 CPU의 적은 스레드 수로는 이를 효과적으로 처리하는데 한계가 있기 때문이다.

한편, 최근 이러한 CPU의 제한된 병렬성을 극복하고 많은 연산을 효율적이고 효과적으로 처리하기 위해 GPGPU(General-Purpose computing on Graphics Processing Unit) 기술이 활용되기 시작하였다. GPGPU는 PC에서 사용되는 그래픽프로세서(GPU)의 병렬연산처리 능력을 일반 연산에 활용할 수 있도록 향상된 GPU 프

· 본 연구는 BK21 프로젝트와 교육과학기술부 도약연구 지원사업(R1-2007-086-01001-0), 서울시 산학연 협력사업(JP090955)의 지원을 받아 진행되었으며, 한국전자통신연구원의 SoC 핵심설계인력양성사업의 부분적인 지원을 받았다. 또한 서울대학교 컴퓨터신기술연구소와 IDEC은 본 연구에 필요한 기자재들을 지원해 주었다.
· 이 논문은 2009 추계학술대회에서 'CUDA를 활용한 스케일링 필터 및 트랜스코더의 성능향상'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 전기컴퓨터공학부
hanjack@iris.snu.ac.kr
†† 비 회 원 : 서울대학교 전기컴퓨터공학부
kys4464@iris.snu.ac.kr
sunghan_suh@iris.snu.ac.kr
††† 정 회 원 : 서울대학교 전기컴퓨터공학부 교수
sha@iris.snu.ac.kr

논문접수 : 2009년 12월 24일
심사완료 : 2010년 2월 10일

로그래밍 기능성을 제공하는 기술을 말한다. 기존 PC환경에 GPGPU 프로그래밍이 가능한 그래픽 카드를 추가로 설치하는 것만으로 개발 및 활용이 가능하여, 연산능력이 뛰어나 앞으로 활용 가능성이 높은 기술이다.

본 논문은 GPGPU 기술, 특히 nVidia에서 지원하는 CUDA기술을 기반으로 한 트랜스코더 성능 향상을 목표로 하였으며, 그 중에서도 스케일링 필터(scaling filter)의 성능향상에 초점을 맞추었다. 스케일링 필터는 입력된 프레임의 축소 및 확대와 색차변환을 수행하는 필터로 연산량이 많아 전체 트랜스코딩 시간에서 큰 비중을 차지한다. 기존 트랜스코더들의 경우 긴 변환시간을 줄이기 위해 고급필터 대신 적당한 화질을 얻을 수 있는 필터를 선택하여 스케일링 필터를 사용해 왔다. 그러나 본 연구를 통해 개발된 트랜스코더를 사용할 경우 고화질을 얻을 수 있는 필터를 사용하면서도 높은 성능을 얻을 수 있어 그 이점이 크다 하겠다.

이 논문의 내용은 다음과 같다. 2장에서는 관련된 연구 및 기술들을 소개하고, 3장에서는 스케일링 필터의 동작을 설명하며, 4장에서는 본 연구에서 수행한 연구에 대해 다룬다. 5장에서는 실험결과를 보이고, 6장에서는 결론을 맺고자 한다.

2. 관련 연구

2.1 트랜스코더 및 스케일링 필터

본 논문에서 사용한 트랜스코더는 많은 사용자를 확보하고 있는 기존의 오픈 소스 트랜스코더인 Mencoder [1]를 기반으로 하였다. Mencoder는 그림 1과 같이 디코딩(decoding), 필터링(filtering), 인코딩(encoding)의

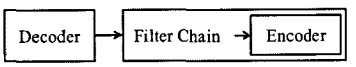

그림 1 mencoder의 내부 구성

표 1 스케일필터 옵션 별 수행시간대비 비중(1080p→720p)

필 터 종 류	변환시간(a)	스케일링필터(b)	b/a (%)
Fast bilinear	58220	16233	27.88%
Bilinear	59249	16987	28.67%
Bicubic (default)	70887	27787	39.20%
Experimental	59203	17181	29.02%
Nearest neighbor	52510	9956	18.96%
Area	57891	15982	27.61%
Luma bicubic/ chroma bilinear	67267	23846	35.45%
Gaussian	69966	27403	39.17%
sincR	166812	122121	73.21%
Lanczos	94583	51405	54.35%
Natural bicubic spline	106642	62435	58.55%

절차를 거쳐 영상을 변환한다. 디코딩은 압축된 동영상의 압축을 해제하여 프레임(frame)들을 얻어내며, 필터링은 각각의 프레임들에 대해 스케일링(scaling), 샤프닝(sharpening), 노이즈 감쇠(noise reduction) 등 부가적인 작업들을 수행한다. 그리고 인코딩은 필터에서 프레임들을 대상 규격에 맞는 영상으로 압축을 수행한다.

스케일링 필터는 프레임의 크기에 대해 확대 및 축소와 색차변환 기능을 수행하는 필터이다. 스케일링 필터는 옵션에 따라 수행시간에 차이가 있으며, 표 1과 같이 전체 수행 시간에 영향을 미친다. 이와 같은 동작 특징은 스케일링 필터의 크기가 옵션에 따라 다르며, 필터의 크기가 커질수록 샘플의 수가 많아져 연산량이 늘어나 스케일링 필터의 수행시간이 늘어나게 되기 때문이다. 따라서 큰 필터의 사용은 총 실행 시간이 증가 및 필터의 비율을 높인다. 큰 필터는 영상의 보다 우수한 품질의 결과물을 얻을 수 있는 이점이 있는데, 표 1을 보면 실행시간 비중이 최대 73%까지 이르게 되므로, 수행시간을 고려한 필터를 선택해서 사용하게 된다.

2.2 CUDA의 구조 및 특성

Common Unified Device Architecture(CUDA)는 nVidia의 GPGPU기술이다. CUDA기술이 지원되는 GPU는 그림 2와 같이 여러 프로세서(processor)를 묶어놓은 Texture Processor Cluster (TPC)가 나열된 형태를 가지고 있다. 그림 2를 통해 TPC의 내부를 볼 수 있는데, [2]에서는 이러한 CUDA의 동작을 상세히 설명하고 있다. 간략히 설명하자면 명령을 수행하는 쓰레드(thread)가 SP상에서 동작하므로, 여러 개의 쓰레드가 병렬적으로 처리하는 구조를 갖는다고 할 수 있다.

[2]에서는 또한 CUDA의 메모리에 대해서도 설명을 하고 있다. 특히 그림 2에 있는 텍스처 메모리와 상수 메모리 등은 캐시 효과가 있으며, 공유 메모리는 프로그램에서 명시적으로 데이터를 가져와 캐시와 같이 사용된다. 이들은 DRAM인 전역 메모리와 구별되는 온 칩(on-chip) 메모리이며, 낮은 지연시간을 갖는다.

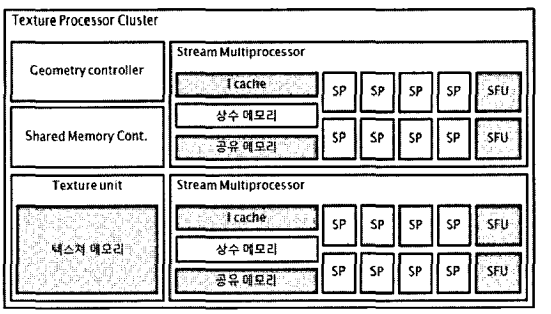

그림 2 nVidia G80계열 CUDA GPU의 구성단위인 TPC의 구성

2.3 CUDA를 적용한 트랜스코더 사례

CUDA를 적용한 트랜스코더는 현재 일부 상용 소프트웨어로 시장에 출시가 되고 있다. 대표적으로 바다붐(Badaboom)[3]과 TMPGEnc[4] 등이 있으며, CUDA를 사용하여 바다붐의 경우 최대 20배, TMPGEnc의 경우 최대 4.47배의 성능 향상을 얻었다고 주장하고 있다. 하지만 바다붐의 경우 H.264 코덱에 대해 제한적으로 적용되었으며, Mencoder를 활용한 다른 인코더 프로그램과 변환속도를 비교해 보았을 경우 실제 그리 크지 않은 성능 차이를 보였다. TMPGEnc의 경우 CUDA기술이 필터에 적용되어 코덱의 제한을 받지는 않지만, 수행속도가 CUDA를 적용하기 전보다 느렸다.

3. 스케일링 필터의 동작

스케일링 필터는 수평과 수직, 두 단계로 영상의 크기 변환을 수행한다. 샘플링을 통해 목적하는 크기에 맞는 픽셀 값을 구하며, 옵션에 따라 스케일링 필터의 종류가 달라져, 필터의 크기도 바뀌게 된다.

수평 방향 스케일링(horizontal scaling)과 수직 방향 스케일링(vertical scaling)의 동작 과정은 그림 3과 그림 4에 표현되어 있다. 각 필터는 지정된 크기대로 입력 프레임에 대해 샘플링(sampling)하고 그 결과의 합을 출력으로 하며, 수평 방향 스케일링은 중간 결과를 별도의 버퍼에 보관하고, 수직 방향 스케일링은 버퍼 내의 프레임을 입력으로 삼아 결과물을 얻게 된다. 예를 들면, 그림 3과 그림 4에서 각 스케일링의 결과인 b와 d영역은 입력 프레임의 a와 c영역이 샘플링 되어 스케일링 필터와 스칼라 곱과 같은 형태로 연산이 된 결과이다.

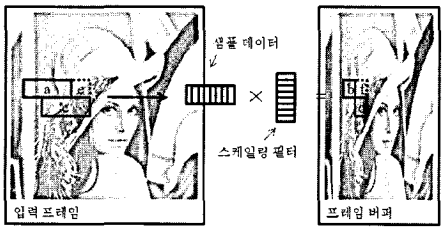

그림 3 수평 스케일링 동작

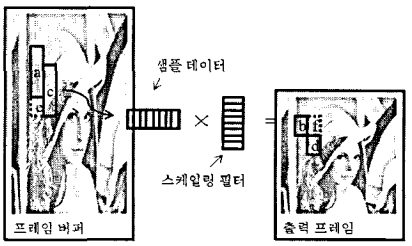

그림 4 수직 스케일링 동작

4. CUDA를 이용한 성능향상

4.1 스케일링 필터의 병렬화 적용 방안

본 연구에서는 결과 프레임의 픽셀을 하나의 스레드가 연산을 하는 형태로 병렬화 하였다. 즉, 하나의 스레드가 각 수평 및 수직 방향 스케일링 단계에서 결과물의 한 픽셀씩을 출력으로 내보내도록 하였다.

4.2 스케일링 필터의 성능향상을 위한 CUDA 최적화

4.2.1 파이프라이닝의 적용

트랜스코딩 과정의 순차적인 진행과정은 그림 5와 같이 나타낼 수 있다. 기존의 진행과정을 유지하게 될 경우 CUDA로 프레임이 전송되고, 스케일링 되고, 다시 CPU로 반환되기까지 CPU 동작은 정지된다.

본 연구는 CPU동작을 효율화하기 위해 비동기 데이터 전송(asynchronous data transfer)을 사용하고 트랜스 코딩의 각 단계가 파이프라이닝 되도록 수정하였다. 그림 6은 이 동작을 나타낸 것으로, CUDA에서 스케일링 필터를 동작시키는 것과 함께 CPU의 다른 작업(디코더, 인코더)을 동시에 수행할 수 있게 된다.

파이프라이닝은 스케일링 필터의 동작 시간을 디코더와 인코더의 동작 시간 사이에 감추는 효과를 내며, 스케일링 필터의 수행시간이 디코더와 인코더의 동작 시간 합보다 작다면, 스케일링 필터가 전체 수행시간에서 차지하는 비율만큼 성능의 향상을 이룰 수 있다.

단, 스케일링 필터의 수행시간이 디코더와 인코더의 수행시간의 합보다 크다면 성능 향상 폭이 다소 감소한다. 그렇다 하더라도 디코더와 인코더의 수행시간만큼 전체 수행시간을 단축시킬 수 있다는 이점이 있어, 이 방법은 유효하다. 파이프라이닝이 적용된 트랜스코더에서 인코더에 입력되는 첫 프레임은 무의미한 값을 가지므로 최초의 입력된 데이터는 무시하도록 수정했다.

4.2.2 중간 프레임 버퍼의 전치(transpose)

수직 스케일링은 그림 4와 같이 수직 방향으로 중간 프레임 버퍼에서 필요한 픽셀 데이터들을 샘플링 한다. 이러한 메모리 접근은 CUDA가 전역메모리에 64바이트

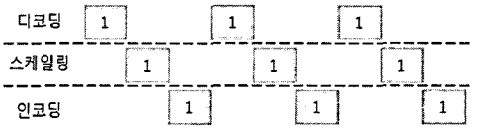

그림 5 기존 트랜스코더의 동작

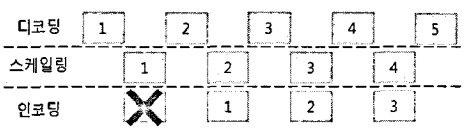

그림 6 파이프라이닝이 적용된 트랜스코더의 동작

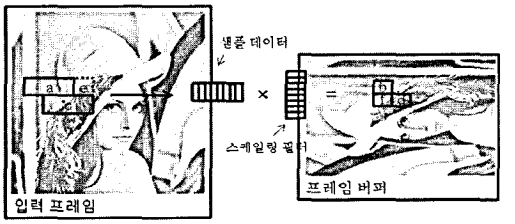

그림 7 프레임 출력을 이항한 수직방향 스케일링

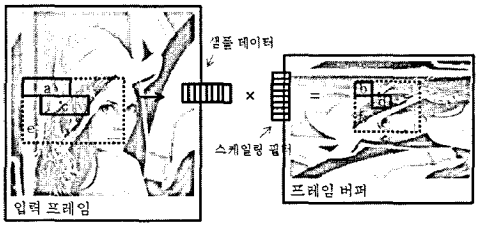

그림 9 공유메모리를 적용한 이항 수직방향 스케일링

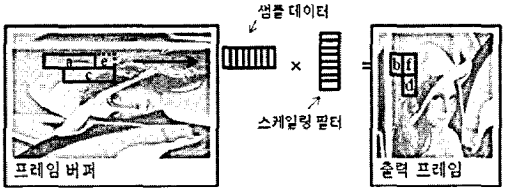

그림 8 이항한 프레임에 대한 수직 스케일링을 수평 스케일링으로 변환한 모습

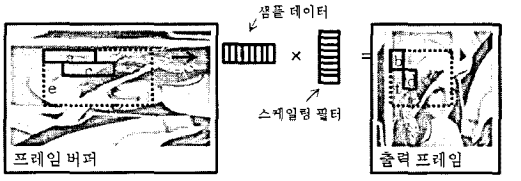

그림 10 공유메모리를 적용한 이항한 프레임에 대한 수평방향 스케일링

씩 정렬된 형태로 묶여서(coalescing) 접근[5]하므로 비효율적인 접근 방식이며, 연속적인 주소로 접근하도록 픽셀을 재배치할 필요가 있다.

따라서, 그림 7과 같이 중간 프레임 버퍼를 전치하여 수직 방향의 데이터들이 연속적인 주소공간에 배치되도록 하였다[6]. 수직스케일링이 그림 8과 같이 수평으로 바뀌어 효율적인 전역메모리사용이 되도록 했다.

4.2.3 텍스쳐 메모리를 통한 전역메모리 지연시간단축

CUDA 아키텍처에서는 읽기 전용 캐시메모리로 텍스쳐 메모리를 제공한다. 이는 프리패치(prefetch) 효과로 지연시간을 줄일 수 있다는 이점이 있다.

본 연구에서는 이러한 특징을 이용해 지역성(locality)이 높은, 즉 연속적이고 중복적으로 접근하는 프레임 데이터와 수평, 수직 스케일링 필터를 텍스쳐 메모리를 통해 접근하여 적중률(hit rate)을 높였다.

4.2.4 공유메모리를 커널의 입출력 버퍼로 활용

그림 7을 통해 스케일링 필터의 동작을 살펴보면, 하나의 출력 픽셀은 주변의 픽셀을 참조하는데, 입력 픽셀을 중복적으로 사용하는 것을 알 수 있다. 또한 개별적인 출력 픽셀의 기록으로 독립적인 전역메모리 접근으로 인해 지연시간이 길어지게 되는 문제가 있다.

따라서, 전역메모리 접근에 의한 지연시간의 단축을 위해 공유메모리에 입력 프레임을 복사하여 전역메모리에 대한 중복 접근을 줄이는 한편 [7] 결과 픽셀의 값을 모아두었다가 한 번에 기록(colesced access)할 필요가 있다[8]. 그림 9와 그림 10은 이러한 동작을 나타낸 것이며, 구현상 공유메모리의 제한된 크기를 고려해 필터 및 블록의 크기 등을 고려해 한 블록에서 처리할 수 있는 크기만큼을 공유메모리로 가져온다. 즉, 오른쪽의 f영역을 만드는데 필요한 왼쪽의 e영역을 공유메모리에 가져와, 전역메모리가 아닌 공유메모리를 대상으로 샘플링을 하도록 하였다. 입력 픽셀 지역성으로 텍스처 메모리에 의존하는 것은 텍스터 메모리의 크기 제한[2]으로 공유메모리로 픽셀을 복사할 때의 용도로 제한했다.

4.2.5 상수메모리의 변환표를 통한 색차변환연산 가속

Mencoder의 필터 체인은 필터의 종류에 따라 각기 다른 색차정보를 필요로 한다. 만약 다른 색차정보를 활용하는 필터가 결합되면 색차 변환을 해야 하며, 이 과정은 본 연구를 수행한 스케일필터에서 이루어진다. 색차 변환은 연산량을 줄이기 위해 정수색차변환 및 변환표를 활용하는데, 이 방법은 CUDA에서도 유효하였다.

단, 색차 변환표를 전역메모리에 두는 방법과 상수 메모리에 두는 방법 및 직접 변환 값을 연산하는 방법 등이 있을 수 있으며, 실험결과 상수 메모리에 변환표를 두어 변환을 하는 방법이 가장 좋은 결과를 얻었다. 이는 전역 메모리의 긴 지연시간과 변환 연산에 따른 레지스터 초과사용에 비해 상수 메모리 접근에 의한 방법이 적은 자원을 사용하기 때문으로 보인다.

5. 실험결과

본 연구에서는 멀티 코어 CPU와 CUDA 지원 그래픽 카드를 탑재한 환경에서 실험을 수행하였다. 구체적인 실험 환경은 다음 표와 같다.

CPU	Intel Q9400 2.66Ghz
RAM	DDR2-6400 4.0GB
GPU	nVidia GeForce 9800GT 512MB
예제영상	1080p, 720p, 23fps MPEG4, 2분 24초 재생

실험은 크게 두 가지로, 첫 번째 실험은 필터의 복잡도에 따르는 트랜스코더의 성능 향상을 확인하고, 두 번째 실험에서는 색차 변환을 수행한 유무에 대한 성능향상폭을 얻었다. 성능향상비율을 구하는 식은 새로운 트랜스코딩 시간 대비 기존 트랜스코딩 시간에 대한 비율로 식 (1)과 같다. 실험은 1080p에서 720p로 축소하는 것으로 비교하였다.

$$\varphi = \frac{1}{1-\frac{t_{prev}-t_{CUDA}}{t_{prev}}} - 1 = \frac{t_{prev}}{t_{CUDA}} - 1 \qquad (1)$$

그림 11은 첫 번째 내용을 확인하기 위해 mencoder에서 제공하는 11가지 종류의 스케일링 필터의 옵션 및 색차 변환에 따른 동작시간을 측정한 것으로 전체 수행시간을 비교한 그래프이다. 그래프의 오른쪽으로 갈수록 크기가 큰 필터가 사용된 경우이며, 필터의 크기는 x축의 필터옵션에 대한 이름 중 괄호 안 숫자(수평 스케일링 필터, 수직 스케일링 필터)에 표시되어 있다. 이 그림을 통해 크기가 큰 필터를 사용할수록 수행시간이 증가하며, CUDA를 사용할 경우 모든 옵션과 색차변환의 경우에 대해서 고루 성능 향상을 얻었음을 알 수 있다. 특히 파이프라인 효과로 sincR 필터를 제외한 모든 필터에서 고른 동작 시간을 보일 수 있었다.

그림 12는 두 번째 내용을 확인하기 위해 측정한 결과의 그래프이며, 필터 크기별 성능 향상 정도 및 색차

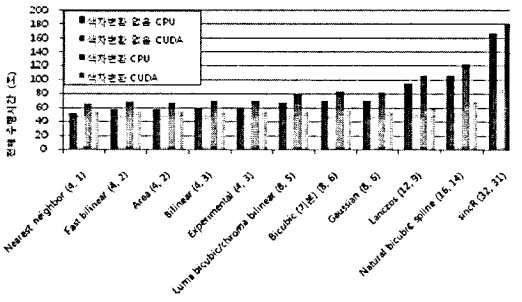

그림 11 스케일링 필터의 옵션 별, 색차변환 유무에 따른 수행시간 비교

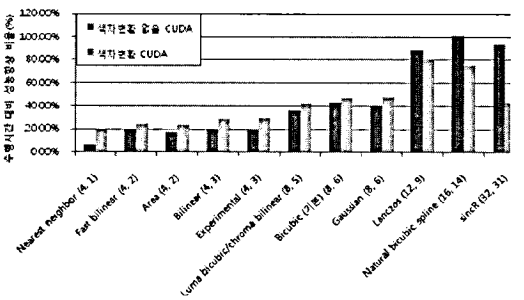

그림 12 스케일링 필터의 옵션 별, 색차변환의 유무에 따른 성능 향상 비율 비교

변환의 영향을 실험한 것이다. 실험 결과, 적게는 6%에서 많게는 101%에 이르는 성능 향상을 확인할 수 있었으며, 일반적으로 복잡한 필터를 사용한 경우일수록 가속화한 트랜스코더의 성능 향상 폭이 높음을 확인할 수 있다. 또한 그림 11을 통해 색차변환이 존재하는 경우 트랜스코더의 동작시간이 다소 길어지지만, 그림 12를 통해 성능 향상 비율은 전반적으로 비슷하여, 어떠한 경우에서든 균일한 성능향상을 이루었음을 확인하였다.

6. 결 론

본 연구에서는 트랜스코더의 스케일링 필터를 GPGPU 기술을 적용하여 가속하는 방법을 연구하였으며 최적화를 위해 파이프라이닝(pipelining)의 적용, 중간 프레임 버퍼의 전치(transpose), 텍스처 메모리의 활용, 공유 메모리 활용, 상수 메모리 상의 변환 표(conversion table)를 활용한 색차 변환 연산 가속 등의 기술을 적용하였다.

실험을 통해 얻은 성능 향상은 기본 옵션에서 30%, 최대 50%에 이르렀으며, 고화질의 결과를 내는 복잡한 필터일수록 보다 높은 성능 향상을 보여 본 연구의 결과를 활용할 경우 트랜스코딩 과정에서 고급 필터들을 보다 빠른 속도로 활용할 수 있을 것으로 보인다.

향후 본 연구는 nVidia CUDA 기술 외에 향후 표준화될 OpenCL 기술 적용 등으로 확장될 수 있을 것이며, 이 경우 보다 많은 기기 등에 적용이 가능해질 것으로 본다.

참 고 문 헌

[1] mencoder, http://www.mplayerhq.hu.
[2] nVidia CUDA Programming Guide 2.3, nVidia Corp. 2009.
[3] Badaboom, http://www.badaboomit.com
[4] TMPGEnc, http://www.tmpgenc.net
[5] nVidia CUDA Best Practices Guide 2.3, nVidia Corp. 2009.
[6] Eric Yung and Frank Jargstorff, Image Processing and Video Algorithms with CUDA, nVision, 60p, September 17, 2008.
[7] Shane Ryoo, Christopher I. Rodrigues, Sara S. Baghsorkhi, Sam S. Stone, David B. Kirk, Wen-mei W. Hwu, Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA, PPoPP '08, 10p, February 20-23, 2008.
[8] Paulius Micikevicius, Optimizing CUDA, SC08, 44p, November 15-21, 2008.