임베디드 GPU 기반 영상처리 고속화 방법

이강운 백아람 최해철

국립한밭대학교 정보통신전문대학원 멀티미디어공학과

kkawoons@naver.com, aram98123@naver.com, choihc@hanbat.ac.kr

Embedded GPU based Fast Image Processing for Mobile Device

Kang-Woon Lee A-Ram Beak Haechul Choi

Hatbat National University, Graduate School of Information and Communications, Dept. of Multimedia Engineering

요약

카메라를 갖춘 모바일 기기가 보편화되면서 모바일 환경에서 영상처리를 이용한 다양한 응용이 확산되고 있다. 영상은 다른 정보에 비해 데이터의 양이 비교적 방대하기 때문에 모바일 환경에서 영상처리를 수행하기 위해서는 처리속도, 전력, 발열 등의 물리적 제약조건이 존재할 수 있다. 본 논문에서는 이러한 문제를 극복하기 위해 모바일 기기에서 코프로세서인 임베디드 GPU(Graphic Processing Unit)를 이용한 영상처리의 고속화 방법을 제시한다. 실험에서는 보편적으로 활용되는 영상처리 알고리즘에 대해 CPU(Central Processing Unit) 및 GPU 각각에서의 성능을 비교함으로써 고속화 방법의 우수성을 검증하고 특징을 분석하였다.

1. 서론

영상처리 알고리즘은 많은 데이터 연산을 요구하지만 작은 크기와 적은 전력을 기준으로 설계된 임베디드 CPU를 사용하는 모바일 기기에서 고속으로 처리하는데 물리적 한계가 존재한다. 이러한 문제를 위해 그래픽 렌더링을 위한 GPU를 CPU 연산에 수행하는 기술인 GPGPU(General Purpose Graphics Processing Unit)을 활용하여 모바일에서도 영상처리 알고리즘을 고속화 하는 연구가 진행되고 있다.[1] 하지만 모바일 환경에서 GPGPU 응용은 임베디드 CPU와 GPU 사이의 낮은 메모리 대역폭과 클럭 속도, 데스크탑 환경에 비해 적은 코어 수와 같은 구조적 한계가 존재하기 때문에 환경을 고려한 효율적 분산처리가 필요하다.

따라서 본 논문에서는 모바일 환경에서 영상처리를 위한 GPU 활용의 효율성을 높이고자 임베디드 GPU에서 다중 알고리즘 처리 구조를 구현하고 성능을 비교 및 분석한다. 본 논문의 2장에서는 임베디드 GPU를 이용한 단일 알고리즘 처리 구조에 대해 설명하고, 3절에서는 임베디드 GPU를 이용한 다중 알고리즘 처리 구조에 대해 설명한다. 4절에서는 단일 및 다중 알고리즘을 임베디드 CPU 혹은 임베디드 GPU에서 처리했을 때 실험 결과를 비교하고 5절에서 결론을 맺는다.

2. 임베디드 GPU를 이용한 단일 알고리즘 처리 구조

영상처리를 위한 GPGPU 활용을 위해 그림 1과 같이 프로세서 간의 명령 또는 데이터 전송이 가능해야 한다. 모바일 환경에서 GPU 명령을 처리하기 위한 대표적인 API로 OpenGL ES 2.0이 있다. OpenGL ES 2.0을 이용한 데이터 처리는 그림 2의 구조를 따르며 GPU가 영상처리 반복문을 수행하기 위해 GLSL(OpenGL Shading Language)을 사용하여 픽셀(Pixel) 당 반복문을 처리한다.

본 연구는 교육부와 한국연구재단의 지역혁신인력양성사업(2012H1B8A2025982) 및 미래창조과학부와 한국연구재단의 신진연구지원사업(NRF-2013R1A1A1010344)으로 수행된 연구결과임.

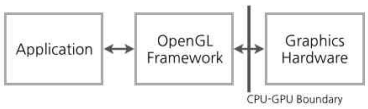

그림 1. CPU-GPU 간 데이터 전송

GPU를 이용한 영상처리를 위해 OpenGL ES 2.0은 전체 화면에 데이터 크기만큼의 사각형과 OpenGL의 뷰(View)를 동일하게 설정하여 픽셀과 텍셀(Texel)이 일대일 사상이 되도록 한다. 그리고 파이프라인에서 사용될 버퍼들을 생성한 후 GLSL인 정점 셰이더(Vertex Shader)와 프래그먼트 셰이더(Fragment Shader)를 이용해 작성된 영상처리 명령을 수행한다. 논문에서 구현된 영상처리 구조는 H.264 디코더를 이용해 추출된 참조 영상을 파이프라인의 입력으로 설정한다. 참조 영상의 모든 프레임 데이터는 텍스쳐 형태로 변환되어 파이프라인을 통과하고 프레임 버퍼에 저장된 후 화면에 출력된다.

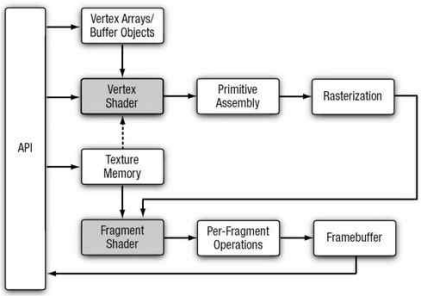

그림 2. OpenGL ES 2.0 그래픽 파이프라인

3. 임베디드 GPU를 이용한 다중 알고리즘 처리 구조

보통의 영상처리는 여러 단계의 알고리즘으로 완성되지만 단일 알고리즘 처리 구조는 한 번의 알고리즘만 처리되어 화면에 출력하게 된다. 이러한 구조를 이용하여 여러 알고리즘을 처리하기 위해서는 프레임 버퍼에 저장된 영상을 주 메모리로 가져와 다시 GPU 메모리로 전송하는 처리 방법을 사용해야 한다. 하지만 프로세서 사이의 낮은 메

모리 대역폭 때문에 처리 속도와 효율 면에서 단점이 있다. 이러한 단점을 해결하고 영상처리 알고리즘을 효율적으로 처리하기 위해 OpenGL의 프레임 버퍼 오브젝트(Frame Buffer Obejct)를 사용한다.

GLSL을 통해 처리되어 프레임 버퍼로 전송되는 영상을 동일한 크기와 영역을 만들어 텍스쳐 메모리의 프레임 버퍼 오브젝트에 저장하고 다음 알고리즘에서 텍스쳐로 다시 사용한다. 프레임버퍼로 전송된 영상은 오프-스크린 렌더링(Off-Screen Rendering) 명령을 이용하여 화면에 출력하지 않고 소비되도록 한다. 프레임 버퍼 오브젝트에서 텍스쳐로 변환된 데이터는 다시 파이프라인에 통과시켜 GLSL 연산을 수행하며 이러한 방법을 이용하여 CPU에서 GPU로 데이터 전송 후 그림 3과 같이 원하는 단계의 영상처리 알고리즘을 수행하게 된다. 그림 3에서 순차적으로 처리되는 영상처리 알고리즘들은 각각 연결된 정점 셰이더와 프래그먼트 셰이더에 작성된 명령을 수행하고 마지막 프레임 버퍼에 저장된 데이터를 모바일 기기 화면에 출력하게 된다.

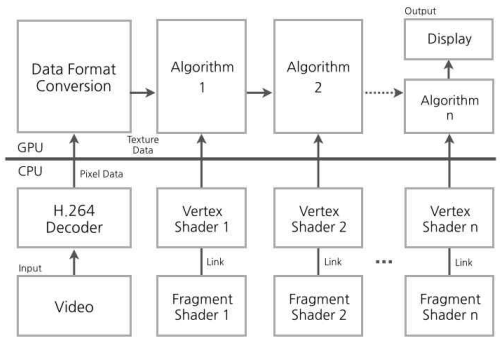

그림 3. 임베디드 GPU를 활용한 다중 알고리즘 처리 구조

4. 실험 결과 및 분석

실험은 ARM v7 기반의 듀얼코어 1.2Ghz CPU와 트리플코어로 구성된 SGX543MP3 GPU, 그리고 1G 메모리로 구성된 모바일 기기에서 진행되었고 영상추출은 60fps로 H.264 디코더에 의해 수행되었다. 참조 영상은 에지 검출을 위해 사각형 객체를 촬영하였고 3가지 해상도로 변환되어 실험에 사용되었다. 영상처리 알고리즘은 에지 검출을 위한 Grayscale, Sobel Edge Detector, Dilate, 그리고 노이즈 제거를 위해 Gauusian Blur와 Birateral Filter가 이용되었다. 실험은 단일 알고리즘을 임베디드 CPU와 GPU에서 각각 OpenCV(Open Computer Vision Library)와 OpenGL로 처리했을 때 성능을 비교하였고 다중 알고리즘도 마찬가지로 임베디드 CPU와 GPU에서 OpenCV와 OpenGL로 처리했을 때 성능을 비교하였다.

표 1. 단일 알고리즘을 임베디드 CPU 혹은 GPU에서 처리했을 때 성능 비교

Algorithms	*Resolutions*	*ⓐ : CPU Processing (ms)*	*GPU(ms)*			*Speed-Up (ⓐ/ⓓ)*
			ⓑ : Processing	*ⓒ : Data Transmission*	*ⓓ : ⓑ + ⓒ*	
Grayscale	1080p	13.7	8.9	25.8	34.7	x0.39
	720p	7.6	5.8	12.2	18.0	x0.42
	480p	4.3	4.5	5.2	9.7	x0.44
Gaussian Blur	1080p	210.4	9.7	25.7	35.4	x5.94
	720p	108.3	6.7	12.2	18.9	x5.73
	480p	47.8	5.2	5.1	10.3	x4.62
Sobel Edge Detector	1080p	215.5	10.1	26.1	36.2	x6.94
	720p	123.5	6.9	12.2	19.1	x6.46
	480p	57.2	5.8	5.1	10.9	x5.24
Dilate	1080p	277.8	9.7	25.8	35.5	x7.82
	720p	131.5	6.7	12.2	18.9	x6.94
	480p	62.0	5.3	5.1	10.3	x5.99
Birateral Filter	1080p	167.8	60.2	27.0	87.2	x1.92
	720p	82.7	28.7	12.7	41.4	x1.99
	480p	37.5	14.1	5.6	19.7	x1.89

표 1은 단일 알고리즘을 임베디드 CPU와 GPU에서 처리했을 때 비교한 성능이다. GPU를 이용할 경우 CPU에서 GPU로 데이터 전송 시간이 반드시 발생하기 때문에 전송 시간을 포함하여 성능을 비교하였다. Graysclae의 경우 데이터 전송 시간의 발생으로 인해 CPU에서 처리가 빠른 것으로 나타났으며 나머지 알고리즘의 경우엔 전송 시간을 포함해도 GPU에서 크게 빠른 것으로 나타났다. Sobel Edge Detector와 Dilate의 결과는 GPU를 활용하여 영상처리를 할 경우에 많은 데이터를 처리할수록 임베디드 CPU 처리에 비해 크게 고속화 되는 결과를 나타냈다.

표 2는 다중 알고리즘을 임베디드 CPU와 GPU에서 처리했을 때 성능을 보여준다. 실험 결과는 임베디드 GPU에서 많은 데이터를 처리하거나 GPU의 수행 연산이 증가할수록 고속화 되는 결과를 보였다. 단일 알고리즘 처리 구조에 비해 GPU의 사용량도 크게 증가하였으며 데이터 전송 시간으로 인한 손실이 단일 알고리즘 처리에 비해 줄어들기 때문에 분산처리에 더 효율적인 장점도 보였다. 반면에 GPU의 연산이 증가할수록 순차처리의 문제점인 프로세서간의 대기시간이 더 늘어나는 문제를 보였다. GPU의 사용량이 일정 수준으로 증가하게 되면 CPU의 사용량이 크게 감소되는 결과를 보였는데 이러한 특징은 GPU에서 영상처리 연산을 수행하는 동안 CPU가 대기상태로 길게 지속되기 때문이다. 처리장치간 대기시간 문제는 CPU와 GPU의 병렬처리를 이용하여 해결이 가능하다.[2]

표 2. 다중 알고리즘을 임베디드 CPU 혹은 GPU에서 처리했을 때 성능 비교

Algorithms	*Resolutions*	*ⓐ : CPU Processing (ms)*	*GPU(ms)*			*Speed-Up (ⓐ/ⓓ)*
			ⓑ : Processing	*ⓒ : Data Transmission*	*ⓓ : ⓑ + ⓒ*	
(1) : Grayscale + Gaussian Blur	1080p	76.0	22.8	26.3	49.1	x1.54
	720p	35.7	12.3	12.4	24.7	x1.44
	480p	15.7	7.3	5.2	12.5	x1.25
(2) : (1) + Sobel Edge Detector	1080p	157.0	38.3	26.5	64.8	x2.42
	720p	73.9	19.2	12.5	31.7	x2.32
	480p	35.1	10.5	5.2	15.7	x2.23
(3) : (2) +Dilate	1080p	211.8	52.6	26.6	79.2	x2.67
	720p	103.8	25.7	12.6	38.3	x2.71
	480p	48.3	13.6	5.3	18.9	x2.55
(4) : (3) + Birateral Filter	1080p	384.4	110.9	27.2	138.1	x2.78
	720p	188.5	52.7	12.7	65.4	x2.88
	480p	86.3	26.0	5.6	31.6	x2.73

5. 결론

본 논문에서는 임베디드 GPU의 효율적 분산처리 활용을 위해 단일 및 다중 영상처리 알고리즘을 임베디드 CPU 혹은 GPU에서 처리했을 때 성능을 분석하였다. 임베디드 CPU를 이용하는 구조에 비교하여 임베디드 GPU에서 처리할 데이터와 수행 연산이 많을수록 활용 효율이 증가하는 결과를 보였다. 따라서 모바일 환경에서 임베디드 GPU를 이용하여 영상처리 알고리즘을 구현하는 경우 프로세서 간의 분산처리는 전체적인 시스템 성능 향상으로 이어지므로 구조적 요소를 고려한 최적화가 필요하다는 결론을 내릴 수 있었다.

참 고 문 헌

[1] N. Singhal, J. W. Yoo, H. Y. Choi, and I. K. Park, "Implementation and optimization of image processing algorithms on embedded GPU", IEICE Trans. on Information and Systems, vol. 95 no. 5, pp. 1475-1484, 2012.

[2] A-Ram Baek, Kangwoon Lee, and Haechul Choi "CPU and GPU Parallel Processing for Mobile Augmented Reality," Image and Signal Processing(CISP), 6th International Congress, pp. 133-137, Dec, 2013.