OpenCL을 이용한 지문개선 가속화

고성학*, 이철*, 박능수*
*건국대학교 컴퓨터공학
e-mail:peilse@gmail.com

Fingerprint enhancement acceleration using OpenCL

Sunghak Ko*, Chul Lee*, Neungsoo Park*
*Dept of Computer Science & Engineering, Konkuk University

요 약

최근 OpenCL, CUDA와 같은 이종 병렬 컴퓨팅 프레임워크가 등장함에 따라, 많은 연산량을 요구하는 알고리즘에 대한 이종 병렬 처리 연구가 늘고 있다. 본 논문에서는 연산량이 많은 지문개선(fingerprint enhancement) 알고리즘을 OpenCL을 이용해 병렬화하고 최적화하여 연산 시간을 단축하고자 한다. 이를 위하여 2차원 FFT 및 필터링 알고리즘을 병렬화하고, Loop Unrolling 및 메모리 접근 최적화 등의 기법을 적용하였다. 실험을 통하여 CPU의 순차적 처리기법과 비교하여 개선된 가속화 기법을 이용한 지문개선 알고리즘이 최대 25배의 성능이 향상하였음을 확인하였다.

1. 서론

최근 CPU외에 GPU 등의 연산 자원을 활용할 수 있게 해주는 CUDA, OpenCL과 같은 이종 병렬 프레임워크가 등장하였다. 이로 인해 기존의 많은 연산을 요구하던 알고리즘을 GPU와 같은 특수 하드웨어로 Offloading함으로써 연산을 가속하려는 시도가 늘어나고 있다.

지문 분류 시스템 PCASYS[1]에서 사용되는 지문개선(fingerprint enhancement) 알고리즘의 경우, 다수에 걸친 2차원 FFT 및 필터링 연산을 수행하게 되어 있다. 이러한 알고리즘은 연산량과 처리해야 할 데이터가 큰 사례에 해당하며, 이를 GPGPU를 통해 가속할 경우 연산에 소비되는 시간을 크게 단축할 수 있을 것으로 기대된다.

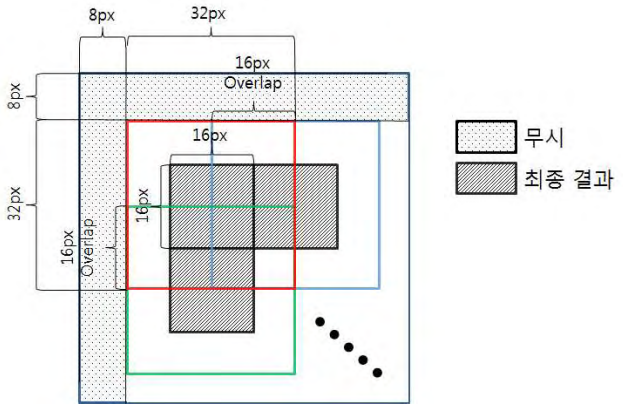
본 논문에서는 지문개선 알고리즘에 대한 OpenCL 병렬화 방법을 제안한다. 또한, 최적화 기법 적용을 통해 최종적인 성능을 향상시키고자 한다.

2. PCASYS의 지문개선 알고리즘

지문 분류 시스템인 PCASYS에는 512×480 크기의 8bit Grayscale 영상을 입력으로 받아 영상 화질을 개선하는 개선 알고리즘[2]이 적용되어 있다.

그림 1에서 보는 바와 같이 이 알고리즘은 원본 영상의 테두리 8픽셀을 무시한 영역 내에서 32×32 크기의 윈도우를 16픽셀씩 중첩하여 슬라이딩한다. 윈도우에 대해 2차원 FFT를 수행한 후 고주파, 저주파 영역의 값을 0으로 치환하여 노이즈를 제거한다. 나머지 주파수 영역의 값은 아래의 식을 연산하여 적용한다.

$$_{k} \quad i\,V_{jk} = (X_{jk}^2 + Y_{jk}^2)^{0.3}(X_{jk} + i\,Y_{jk}) \qquad (1)$$



(그림 1) 윈도우 연산

이후 2차원 IFFT를 수행하여 변환한 후, 윈도우 내에서 가장 큰 절대값 maxABS를 찾는다. 이후 모든 픽셀에 대해 127/maxABS값을 곱해준 후, 128.5를 더하여 Normalize한다. 이 후 그림 1과 같이 윈도우 중앙의 16×16영역만을 잘라내어 최종 결과에 합산한다.

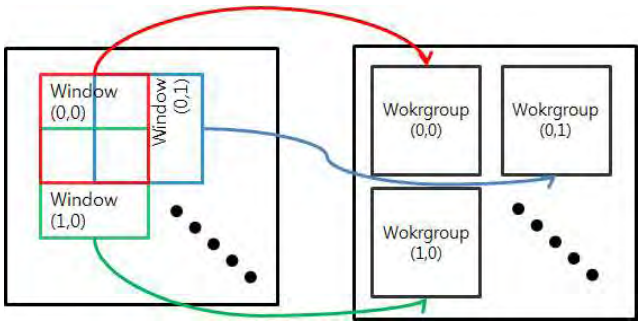
위와 같은 연산 작업을 반복하여 개선된 지문 영상을 얻어낼 수 있다.

3. 지문개선 알고리즘의 OpenCL 병렬화

위에서 설명한 알고리즘의 경우, 32×32 윈도우를 16픽셀 중첩하므로 각 윈도우당 원본 데이터가 일부 중복되고 있다. 하지만 연산 자체의 경우 각 윈도우간의 의존성이 없으므로 전체 영역을 연산하기 위해 필요한 윈도우 연산들을 병렬화하여 동시 수행할 수 있다.

3.1 윈도우 - 워크그룹 매핑

512×480 영상에서 테두리 8픽셀을 제외한 영역에서 윈도우 슬라이딩을 하므로, 최종적으로는 496×464 영역에 대한 윈도우 연산을 한다. 따라서 윈도우 연산 횟수는 (496÷16-1)×(464÷16-1)=30×28회가 된다.



(그림 2) 윈도우-워크그룹 매핑

그림 2에서와 같이 윈도우 연산을 워크그룹으로 1:1 매핑을 하므로 워크 그룹의 개수는 30×28개가 된다. 워크그룹 하나는 윈도우 하나에 대응하며, 워크그룹별로 매핑된 윈도우에 해당하는 연산을 수행하도록 한다.

3.2 워크아이템 할당

워크그룹에서는 내부적으로 32×32영역에 대한 2차원 FFT를 수행하게 된다. 본 논문에서는 버터플라이 연산의 반복을 통한 FFT를 사용한다. 버터플라이 연산은 데이터를 2등분하여 분할 연산하기 때문에 32÷2=16개의 워크아이템으로 병렬화 할 수 있다. 또한 2차원 FFT를 위해서는 버터플라이 연산을 세로 개수만큼 수행해야 하므로, 총 16×32개의 워크 아이템으로 병렬화가 가능하다. 따라서 1개의 워크 그룹에는 16×32개의 워크 아이템을 할당한다. 전체 워크 아이템의 개수는 (30×16)×(28×32)=480×896이 된다.

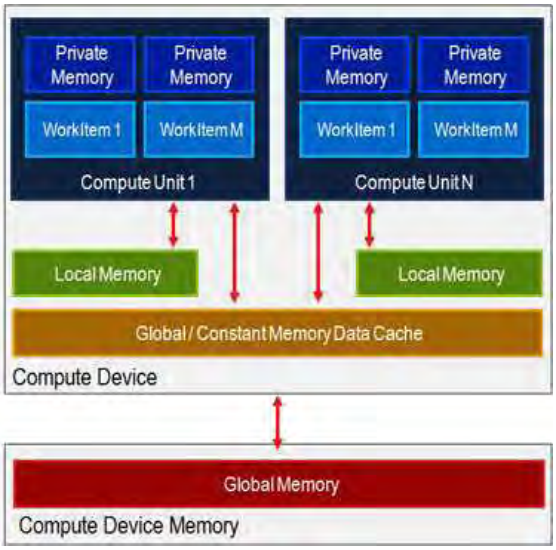
4. 성능 최적화

4.1 Loop Unrolling

FFT연산을 위해서는 버터플라이 연산을 반복할 필요가 있다. 버터플라이 반복 횟수는 $\log_2$(데이터 길이)가 되므로, 이 경우 $\log_2(32) = 5$가 된다. 이 때 5회 반복 연산을 for문과 같은 반복문을 통해 수행하는 것이 일반적인데, GPGPU의 특성상 이러한 제어문은 연산 성능을 저하시킬 수 있다[3]. 따라서 이러한 반복문을 Loop Unrolling 하여 제거하고, 코드 안에 반복 회수만큼 버터플라이 연산 코드를 삽입함으로써 연산 성능 저하를 회피할 수 있다. 본 논문에서는 총 5회의 반복이 필요한 버터플라이 연산 코드를 5번 직접 삽입하였다.

4.2 메모리 접근 최적화

OpenCL에서 Host가 되는 CPU와 통신이 가능한 메모리 영역은 Global Memory이다. 그러나 그림 3에서 보는 바와 같이, Global Memory는 OpenCL 장치의 연산 유닛에서 접근할 때 가장 시간이 많이 소요되는 영역이다[4]. 따라서 이러한 Global Memory에 대한 접근을 최소화하면 전체적인 수행 속도를 높일 수 있음을 짐작할 수 있다.



(그림 3) GPGPU 메모리 접근 도식

본 논문에서는 Global Memory로의 접근을 최소화하기 위해 32×32 Local Memory를 선언하여, 연산 도중의 중간 결과값을 저장하도록 구현하였다. 따라서 5회의 버터플라이 연산 중 Global Memory에 접근하는 횟수는 초기 원본 데이터를 읽을 때, 최종 연산 결과를 쓸 때의 2번으로 줄였다. 이를 통해 메모리 접근에 소비되는 시간을 줄일 수 있도록 최적화 하였다.

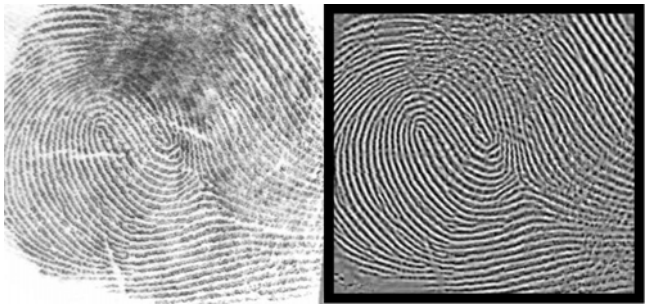
5. 실험 및 결과

실험은 표 1과 같은 환경에서 수행하였으며, 샘플 지문 데이터 2700개에 대해 지문 개선을 수행하고 그 평균 시간을 측정하였다.

CPU	Intel Core i7 975 Extreme (3.33Ghz)
GPU	Nvidia Tesla C2075

<표 1> 실험 환경

실험은 CPU에서 실행되는 NBIS 표준 Sequential 코드, 최적화 적용 전의 OpenCL코드, 최적화를 적용한 OpenCL코드를 각각 실행하여 측정하였다.

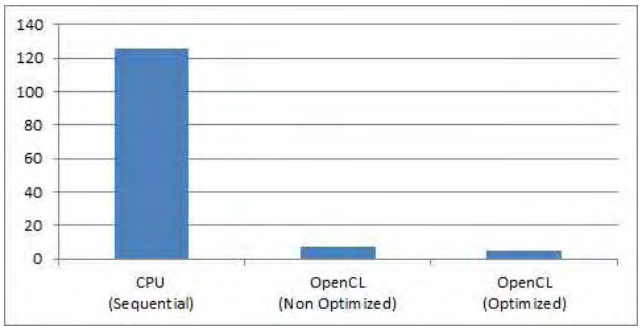


(그림 4). 지문 개선 수행 전, 수행 후

출력 결과는 셋 모두 동등함을 확인하였으며, 각각의 실행 시간은 다음과 같다.

CPU code	OpenCL code (witn non-optimized)	OpenCL code (with optimized)
125.8 ms	7.2 ms	4.9 ms

<표 2> 지문개선 알고리즘 평균 실행 시간



(그림 5) 지문개선 알고리즘 평균 실행 시간

6. 결론

실험 결과, CPU에서 실행했을 때에 비해 최적화 하지 않았을 경우 약 18배, 최적화 했을 경우 약 25배의 성능 향상이 관측되었다. 기존 지문 개선 알고리즘이 반복적으로 2차원 FFT를 호출하는 구조이기 때문에 연산량이 큰 알고리즘에 속했다. 그 때문에 OpenCL을 통해 연산을 병렬화 했을 때 성능 향상이 두드러졌다고 평가할 수 있다.

본 논문에서는 비교적 단순한 워크 아이템 분배 정책을 사용하였고, 적용한 최적화 기법이 많지 않았다. 이후에는 보다 효율적인 워크 아이템 분배 정책을 연구하고, 다양한 최적화 기법을 적용하여 성능을 높일 수 있도록 할 예정이다.

참고문헌

[1] Watson, GT Candela PJ Grother CI, R. A. Wilkinson, and C. L. Wilson, “PCASYS－A Pattern-Level Classification Automation System for Fingerprints” Vol. 4. Technical Report NISTIR 5647, 1995.

[2] Craig I. Watson, Michael D. Garris, Elham Tabassi, Charles L. Wilson, R. Michael McCabe, Stanley Janet, Kenneth Ko, “User's Guide to NIST Biometric Image Software(NBIS)", National Institute of Standards and Technology, 2004.

[3] Murthy, G. S., Ravishankar, M., Baskaran, M. M., & Sadayappan, P. "Optimal loop unrolling for GPGPU programs." Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on. IEEE, 2010.

[4] Nvidia Corp, "OpenCL Best Practice Guide", Nvidia CUDA Documentation, Nvidia Corp, 2011.