모바일 GPU 를 이용한 SURF 알고리즘의 병렬화

유진우*, 최광철**, 박인규*
인하대학교 정보공학과*, 삼성전자 DMC R&D Center**
jinwoo1028@gmail.com*, choikc@samsung.com**, pik@inha.ac.kr*

요 약

본 논문에서는 특징추출의 대표적 기법인 SURF(Speeded-Up Robust Features)[1, 2] 알고리즘의 병렬화 기법을 제안한다. SURF 알고리즘을 구성하는 적분영상 생성, Hessian 행렬식과 non-maximum suppression 계산, 기술자 추출의 전 과정을 OpenGL ES 2.0 에서 효율적으로 병렬화 하고, 최신 스마트폰 단말기에 탑재된 GPU 인 POWERVR SGX540[4]에서 구동한다. 실험결과 같은 단말기의 CPU 에서의 수행에 비해 약 1.8 배 성능향상을 보인다.

1. 서론

최근 개발되는 스마트폰은 고성능 GPU(graphics processing units)를 탑재하고 있으며 이를 3 차원 그래픽 렌더링 외의 일반적인 분야에 활용하기 위한 GPGPU (general purpose GPU)의 필요성이 증대되고 있다. 또한 모바일 그래픽 API 의 표준인 OpenGL ES 2.0 에서는 shader 를 이용한 범용 프로그래밍이 가능하다[3]. 따라서 기존의 모바일 CPU 에서 실시간으로 처리하지 못했던 다양한 알고리즘이 GPU 에서의 병렬처리를 통해 실용화 될 것으로 예측된다.

본 논문에서는 컴퓨터비젼 분야의 대표적인 특징점 추출 알고리즘 중 하나인 SURF 알고리즘을 효율적으로 병렬화하기 위한 기법을 제안한다. 또한 최신 스마트폰 단말기상에서의 여러 응용 분야에 사용이 가능하도록 모바일 GPU 를 이용하여 병렬화 된 알고리즘을 구현하였다.

2. 특징점의 위치와 기술자 추출

SURF 는 적분영상 생성, Hessian 행렬식 계산, non-maximum suppression 의 3 단계를 통해 특징점을 추출하고, 각 특징점의 방향과 기술자를 계산하여 저장한다.

모바일 기기의 센서를 이용하여 장면의 방향 계산이 가능하기 때문에 본 논문에서는 알고리즘의 간략화를 위해서 특징점의 방향을 계산하는 과정은 제외하였다.

2.1 적분영상 생성

CPU 연산과 달리 병렬 연산인 OpenGL ES 에서 CPU 와 동일하게 적분 영상을 생성하게 되면, 하나의 thread 에서 많은 텍스처의 접근이 필요하게 되어 속도 저하에 영향을 미친다. 따라서 속도 향상을 위해 2D-reduction 알고리즘을 적용하여 적분영상을 생성한다. 하지만 $\log n$ 번의 렌더링이 필요하다. 그림 1 은 2D-reduction 의 계산 과정을 보인다.

OpenGL ES 2.0 에서는 최대 한 채널에 1 바이트의 텍스처가 지원된다. 따라서 계산된 적분영상의 값을 저장하기 위해서는 추가적인 연산을 통해 4 개의 채널(RGBA)을 이용한다.

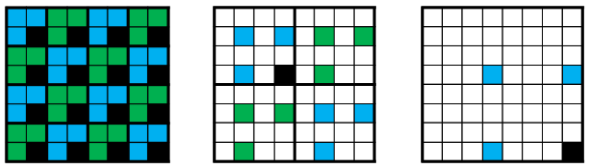
2.2 Hessian 행렬식 계산 및 Non-Maximum Suppression 수행

Hessian 행렬식은 영상의 모든 점에서 동일하게 계산된다. 하지만 계산 량을 줄이고 속도의 향상을 위해 octave 에 따라서 2^{octave} 만큼 샘플링된 텍스처를 필터의 개수만큼 생성하고 Hessian 행렬식을 계산한다.

Hessian 행렬식을 계산한 뒤 각 octave 에서 3 개의 필터 크기를 차례대로 선택하여 non-maximum suppression 과정을 수행한다. 이 결과 주위 픽셀에 비해서 큰 Hessian 행렬식 값을 갖는 위치가 특징점의 위치가 된다.

2.3 특징점 테이블 생성

Non-maximum suppression 의 결과 특징점에 해당하는 점들이 텍스처에 공간적으로 분포되어 있다. 이를 그대로 사용하게 되면 기술자를 추출하는 단계에서 특징점이 아닌 화소에도 thread 가 발생하게 되어 속도 저하의 원인이 된다. 따라서 추출된 특징점을 하나의 테이블로 통합하여 저장하고, 테이블에 기록된 특징점만을 대상으로 기술자를 계산한다. 특징점 테이블 생성 과정은 GPU 에서 병렬처리가 용이하지 않으므로, 텍스처를 CPU 메모리로 복사하여 수행하고, 생성된 테이블을 다시 텍스처로 옮겨서 기술자 추출 단계를 수행하기 위한 입력 텍스처



(a) (b) (C)

그림 1. 8X8 영역의 2-D reduction 알고리즘의 예, 부분영역 안의 초록색 또는 파란색의 픽셀과 검은색 픽셀의 값을 더하여 검은색 픽셀에 출력. 모든 픽셀에서 동일한 연산이 수행된다. (a) 첫 번째 렌더링. (b)두 번째 렌더링. (c)세 번째 렌더링.

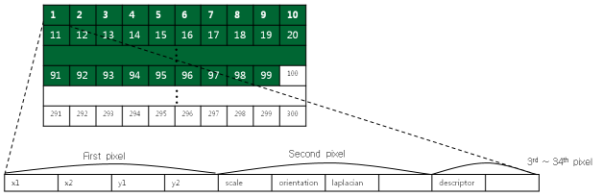


그림 2. 특징점 테이블의 구조

를 생성한다. 그림 2는 특징점 테이블을 보여준다.

2.4 기술자 추출

병렬연산을 위해 기술자를 추출하는 과정은 총 6단계로 구성된다.

먼저 각 특징점에 대해 20×20의 영역을 생성하고 haar wavelet 을 계산한다. 그 뒤 각 영역을 4×4 의 부분영역으로 나눠서 초기의 기술자 $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$를 계산한다.

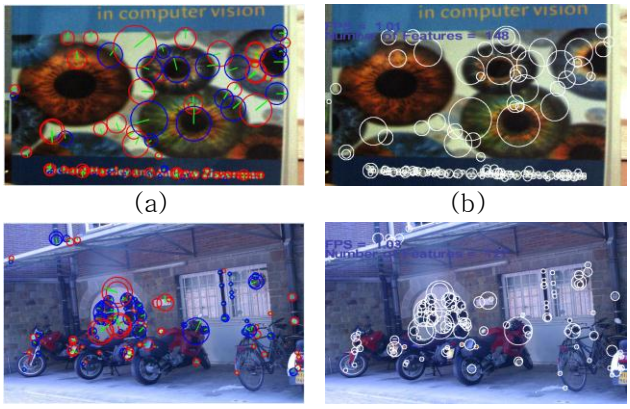
Normalize 를 위해 세 번째 과정에서 각 특징점 영역 안의 모든 $\sum d_x, \sum d_y$ 와 $\sum |d_x|, \sum |d_y|$를 각각 더하고, $\sqrt{\sum D_x^2 + \sum D_y^2 + \sum |D|_x^2 + \sum |D|_y^2}$ ($D_x = \sum d_x$) 를 계산한다. 이 결과를 초기 기술자에 나눠주는 계산을 수행하여 최종적인 기술자를 추출하고, 마지막 과정에서 특징점 테이블에 추출된 기술자를 저장한다.

3. 실험 결과 및 분석

본 논문에서는 1GHz A8 Cortex CPU 와 POWERVR SGX 540 의 GPU 가 내장된 응용 프로세서인 S5PC110 을 장착한 안드로이드 2.1 기반의 스마트폰인 SHW-M110S 를 플랫폼으로 개발된 병렬 알고리즘을 구현하였다.

먼저 추출된 특징점 위치의 정확도를 확인하기 위해서 PC 기반의 오픈 소스인 OpenSURF[5]와 비교하였고, 그림 3 에서 유사한 결과가 나오는 것을 확인할 수 있다.

다음 표 1 은 SHW-M110S 의 안드로이드 NDK 를 이용한 CPU 구현과 GPU 구현의 수행속도를 비교한 결과를 보여준다. GPU 의 수행시간이 CPU 를 이용한 수행시간에 비해 약 1.8 배 향상되었다.



(a) (b)

(c) (d)

그림 3. 특징점 추출 결과. 원의 중심과 반지름은 각각 특징점의 위치와 scale 을 나타낸다. (a), (c) OpenSURF 수행결과. (b), (d) 모바일 GPU 의 수행 결과
(threshold : 0.0023)

표 1. 모바일 CPU, GPU 의 수행시간 비교.
(단위: 초, 영상 크기 : 800×480)

	CPU	GPU
특징점 개수	209	148
수행시간	1.703	0.943

4. 결론

결과에서 보듯이 제안된 병렬 알고리즘과 기존 SURF 의 결과가 유사한 것을 확인할 수 있다. 아직 OpenGL ES 2.0 의 제약사항으로 인해서 CPU 의 결과와 비교하여 많은 성능 향상을 보이지 못하지만 앞으로 제약사항이 보완되면 CPU 에 비해 많은 성능 향상을 확인할 수 있을 것으로 기대된다.

감사의 글

본 연구는 삼성전자㈜의 지원을 받아 수행된 연구임

참고문헌

[1] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. ‘Speeded-Up Robust Features(SURF)’. Computer Vision and Image Understanding 110(3):346-359, 2008.

[2] N. Cornelis and L. Van Gool. ‘Fast scale invariant feature detection and matching on programmable graphics hardware’. pages 1-8, June 2008.

[3] A. Munshi, D. Ginsburg, and D. Shreiner, OpenGL ES 2.0 Programming Guide, Addison Wesley, 2009.

[4] Imagination Technologies Ltd, *POWERVR SGX Factsheet*, 2008.

[5] http://www.chrisevansdev.com/