2014년 10월 전자공학회 논문지 제 51권 제 10 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 10, October 2014 http://dx.doi.org/10.5573/ieie.2014.51.10.190

논문 2014-51-10-24

OpenCL을 이용한 모바일 ADAS : 보행자 검출

(Mobile Advanced Driver Assistance System using OpenCL : Pedestrian Detection)

김 종 희*, 이 충 수*, 김 학 일**

(Jong-Hee Kim, Chung-Su Lee, and Hakil Kim©)

요 약

본 논문에서는 상용 스마트폰에서의 첨단운전자보조시스템(ADAS)을 위해 모바일 플랫폼에 최적화된 cascade 방식의 HOG 특징을 이용한 보행자 검출 방법을 제안한다. 제한된 모바일 플랫폼 자원을 효율적으로 사용하기 위해 OpenCL 병렬처리 라이브러리를 이용하였고 크게 두 가지 방법으로 수행속도를 향상시켰다. 첫째, 호스트 코드에서 OpenCL 프로그램 빌드 옵션을 특정하고 작업 그룹 크기를 조절하였다. 둘째, 커널 코드에서 지역 메모리와 LUT 등을 사용하여 가속화하였다. 성능 평가를 위하여 널리 알려진 영상처리 라이브러리인 OpenCV for Android 함수의 모바일 CPU 수행 결과와 비교하였으며 실험 결과, OpenCV의 hogcascade 함수보다 25% 향상된 처리속도를 보였다.

Abstract

This paper proposes a mobile-optimized pedestrian detection method using Cascade of HOG(Histograms of Oriented Gradients) for ADAS(Advanced Driver Assistance System) on smartphones. In order to use the limited resource of mobile platforms efficiently, the method is implemented by the OpenCL(Open Computing Language) library, and its processing time is reduced in the following two aspects. Firstly, the method sets a program build option specifically and adjusts work group sizes as variety of kernels in the host code. Secondly, it utilizes local memory and a LUT(Look-Up Table) in the kernel code to accelerate the program. For performance evaluation, the developed algorithm is compared with the mobile CPU-based OpenCV(Open Computer Vision) for Android function. The experimental results show that the processing speed is 25% faster than the OpenCV hogcascade.

Keywords : Pedestrian detection, OpenCL, Embedded GPGPU, ADAS, HOG

Ⅰ. 서 론

현대 사회를 살아가는 바쁜 운전자들에게 각광받고 있는 자동차 관련 산업 중의 하나가 첨단운전자보조시스템 즉, ADAS(Advanced Driver Assistance System)이다. ADAS 중에는 운전자와 동승자, 차량을 보호하기 위한 기능도 존재하지만 유사시에 큰 부상을 당할 수 있는 보행자를 보호하기 위한 기능도 존재한다. 그 중 하나가 보행자 검출 기능이다. 보행자 검출 기능을 포함한 PCW(Pedestrian Collision Warning) 장치는 1차 시장(primary market)에서 뿐만 아니라, 2차 시장(after market)에서도 점차 활성화되고 있다.

상대적으로 값 비싼 고급 자동차를 위한 옵션인 1차 시장보다는 많은 운전자들을 위한 2차 시장을 통한

* 학생회원, ** 평생회원, 인하대학교 정보통신공학부
(School of Information and Communication Engineering, Inha University)
© Corresponding Author(E-mail: hikim@inha.ac.kr)
※ 본 논문은 산업통상자원부 산업융합원천기술개발사업으로 지원된 연구결과임. [10041664, 멀티 Shader GPU 통합형 멀티 코어 퓨전 프로세서 원천기술 개발]
접수일자: 2014년08월20일, 수정일자: 2014년09월29일
게재확정: 2014년10월06일

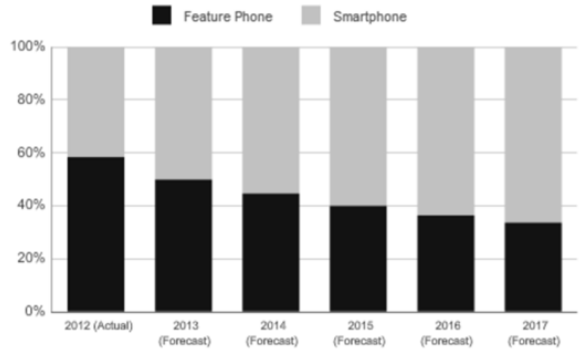

그림 1. 세계 모바일폰 기기 유형 변화 예상 (IDC[1])
Fig. 1. Worldwide mobile phone forecast by device type (% Units).

ADAS 장치의 보급화는 교통사고 및 그에 따른 인사사고를 줄이는 데 도움이 될 것이다. 또한 현대인들이 널리 사용하고 있는 스마트폰이 이러한 ADAS를 위한 장치로서 활용될 수 있다면 그 효과는 더욱 빠르게 현실화될 수 있을 것이다. 그림 1에서 볼 수 있듯이 스마트폰은 전 세대인 피처폰에 비해 더 많이 보급되어 있을 것으로 예상되며 이러한 양상은 점점 심화될 것으로 보인다. 게다가 그림 2에 나타나있듯이 모바일 프로세서의 성능이 점차 좋아지면서 과거 데스크톱 CPU의 성능을 능가하고 있다. 따라서 스마트폰에서의 영상 취득 및 고등의 알고리즘 처리가 가능해졌고 본 논문에서는 보행자 검출 알고리즘을 스마트폰으로 구현하여 이를 증명하고자 한다.

보행자 검출을 위한 특징 추출 방법에는 N. Dalal과 B. Triggs가 제안한 Histograms of Oriented Gradients(HOG) 방법[2]이 널리 쓰이고 있고, P. Viola와 M. Jones가 제안한 Haar-like 기반 방법[3]도 많이 쓰이고 있다. 또한, 분류를 위해서는 Support Vector Machine(SVM)[4]과 AdaBoost[5] 알고리듬에 따른 특징 선택 및 cascade 방식 분류가 많이 쓰이고 있다. 특히, Q. Zhu는 기존의 HOG+SVM 방식을 벗어나 HOG와 cascade 방식을 결합한 Cascade of HOG 보행자 검출 방법[6]을 제안하였다. 최근에는 수행시간이 적은 간단한 특징이나 함수를 2단계 이상의 연이은 구조로 그 검출 성능도 높이는 연구[7~8]도 진행되고 있다.

한편, 최근 들어 모바일 GPU에서 OpenGL ES와 OpenCL 라이브러리를 이용하여 기본적인 병렬영상처리 함수를 구현[9]하거나 Viola-Jones 알고리듬을 GPU에서 가속화하여 얼굴 검출을 하는 연구[10]도 진행되고

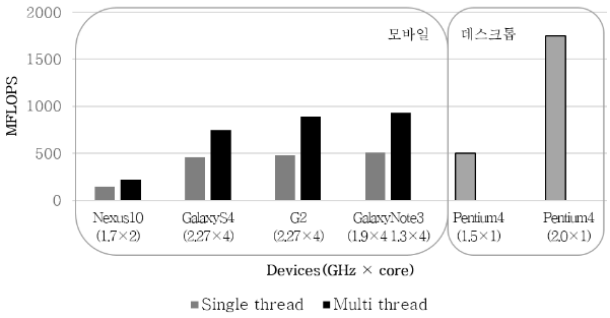

그림 2. 모바일/데스크톱 CPU 성능 추이
Fig. 2. Mobile/Desktop CPU performance changes.

있다.

본 논문에서는 Cascade of HOG 방식을 OpenCL로 병렬화하여 스마트폰에서 구현하여 보행자를 검출하는 방법을 제안한다. 모바일 플랫폼에서는 프로세서 클럭 주파수나 메모리 자원에 제약사항이 비교적 많이 따르므로 병렬처리를 함에 있어서도 데스크톱보다 더 많은 노력을 기울여야 성능 향상을 가져올 수 있다. 본 논문에서는 OpenCL로 Cascade of HOG를 구현하는데 이러한 제약사항을 극복할 수 있도록 지역 메모리 활용, LUT를 통한 근사화, 빌드 옵션 설정 등으로 해결하는 과정을 기술할 것이다.

제Ⅱ장 본론에서는 Cascade of HOG 특징 추출 및 분류 방법과 함께 스마트폰에서의 OpenCL 병렬화 프레임워크를 소개한다. 제Ⅲ장에서는 제Ⅱ장의 내용을 구현 및 실험한 내용을 소개하며 마지막 제Ⅳ장에서 이에 대한 결론 및 향후 연구 방향을 제시한다.

Ⅱ. 본 론

1. Cascade of HOG

본래의 HOG와는 다르게 Cascade of HOG에서의 특징 추출은 검출 윈도우 내에서 가능한 많은 특징(블록)을 후보군으로 둔 뒤 AdaBoost 방법을 통해 cascade 분류를 위한 특징들을 선택한다. 이는 본래의 HOG보다 더 많은 정보를 추출할 수 있고 블록의 위치나 크기에 따라 머리나 다리 같은 부분적 의미도 부여할 수 있는 장점이 있다. Cascade of HOG 특징 추출의 상세한 과정은 다음과 같다.

1) 색 공간 변환, 평탄화, 감마 정규화 등 필요한 이미

지 전처리 과정을 진행한다.

2) HOG 특징 추출을 위하여 각 픽셀의 기울기 정보를 계산한다. 이미지 좌표 $I(x,y)$에서의 기울기 크기 $m(x,y)$는 식 (1)과 같이, 기울기 방향 $\theta(x,y)$는 식 (2)와 같이 구할 수 있다.

3) 하나의 검출 윈도우 내에서 총 5,031개의 특징 추출 가능한 블록을 각각의 특징 후보군으로 둔다. 이 블록들은 최소 12×12 픽셀 크기부터 최대 64×128(검출 윈도우 크기) 픽셀 크기까지 (1:1), (1:2), (2:1)의 비율로 {4, 6, 8}의 샘플링 간격을 두어 다양한 크기 및 위치를 모두 추출한 것이다.

$$m(x,y) = \sqrt{d_x(x,y)^2 + d_y(x,y)^2} \tag{1}$$

$$\theta(x,y) = \tan^{-1}\frac{d_y(x,y)}{d_x(x,y)} \tag{2}$$

$where,$

$$d_x(x,y) = I(x+1,y) - I(x-1,y)$$

$$d_y(x,y) = I(x,y+1) - I(x,y-1)$$

특징 추출 후 훈련 과정에서는 AdaBoost 알고리듬을 통해 이들 중 미리 설정한 최소 검출률 허용 수치 및 최대 오검출률 허용 수치에 따라 약분류기들을 조합하여 최종적으로 강분류기를 생성한다. 검출 단계에서는 이 강분류기를 이용하여 보행자와 보행자가 아닌 검출 윈도우 영역을 분류해내게 된다.

2. 스마트폰에서의 OpenCL 병렬화 프레임워크

스마트폰에서의 OpenCL을 이용한 병렬영상처리 알고리듬 구현은 PC와는 다소 다른 구조를 갖기에 이를 위한 프레임워크를 참고하였고[9] 이는 그림 3과 같다. 이 프레임워크에 따르면 OpenCL 라이브러리는 계산 관련 API이므로 영상처리 부분만을 위하여 동작하고 입력과 출력은 OpenGL ES 파이프라인을 이용한다. OpenCL을 이용한 병렬영상처리는 다음과 같은 과정을 거친다.

1) OpenGL 타입의 텍스처로 메모리에 저장된 입력영상을 GL_Sharing 관련 함수로 OpenCL 메모리 객체로 받아온다.

2) 이를 OpenCL 커널에 구현된 일련의 처리 과정을 통하여 병렬영상처리를 수행한다.

3) 결과 출력을 위하여 OpenCL 메모리 객체를 다시 OpenGL 타입의 텍스처로 변환한 뒤에 바로 스마트폰 화면에 출력한다.

일반적으로 모바일 환경에서는 CPU와 GPU가 동일한 하나의 메모리를 서로 다른 구역으로 구분하여 각각 활용하지만, OpenCL 1.1에서는 위 1)의 메모리 객체를 CPU와 GPU에서 모두 접근이 가능하도록 도와준다. 따라서 CPU와 GPU 간 자료이동에 대한 제약에서 벗어날 수 있다.

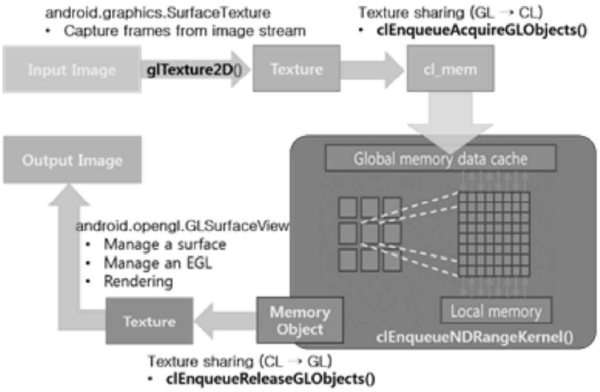

그림 3. OpenCL 1.1 기반의 병렬영상처리 프레임워크 [9]

Fig. 3. The framework of parallel image processing based on OpenCL 1.1 version.

3. 스마트폰에서의 OpenCL 보행자 검출 알고리듬 구조

C/C++ 언어 기반의 소스 코드를 Java 언어 기반의 Android 스마트폰으로 이식하기 위해 Android NDK(Native Development Kit)를 사용하였고, NDK 영역에서 실제 보행자 검출을 위한 프로그램을 작성하였다. 분류기를 불러오는 과정을 포함한 보행자 검출 알고리듬 구조는 그림 4와 같다. Android SDK(Software Development Kit)를 이용하여 Java 언어로 작성되는 Java 영역에서는 주로 분류기 정보가 담겨있는 xml 파일을 불러와 분석하여 필요에 맞는 메모리에 저장하여 주고, C/C++ 언어로 작성되는 NDK 영역에서는 주로 보행자 검출을 위한 OpenCL 호스트 코드 및 커널 코드

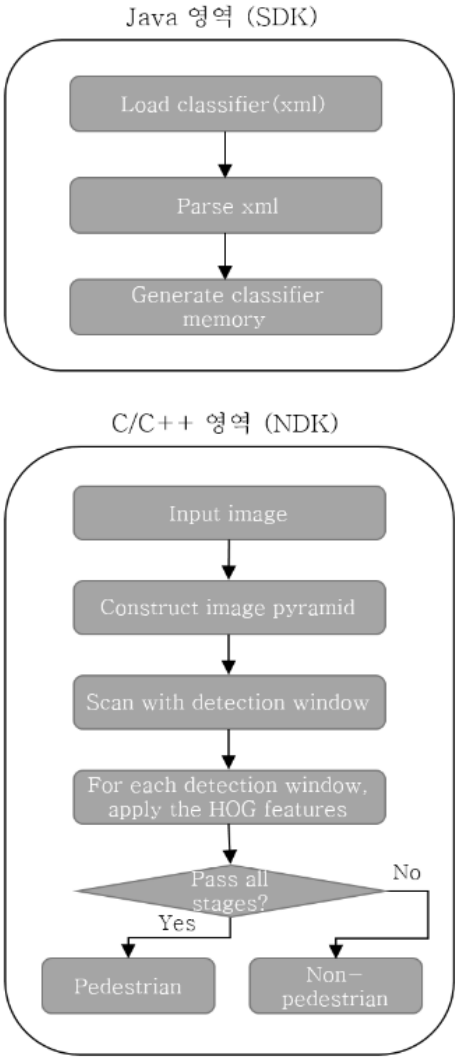

그림 4. 스마트폰 보행자 검출 알고리듬 구조
Fig. 4. Structure of pedestrian detection algorithm for smartphone.

로 프로그램을 생성한다.

보행자 검출 알고리듬을 구현하기 위하여 총 6가지의 커널을 구현하였다. 커널의 구조와 역할 및 흐름도는 그림 5와 같고 각각의 커널의 결과는 그림 6에 나타내었으며 커널을 나누어 구현한 기준은 다음과 같다.

1) 커널 이식성을 극대화하기 위하여 의미 있거나 자주 쓰이는 기능을 최소 단위로 하여 분리한다. 예를 들어, RGB2GRAY나 Gaussian 같은 커널은 어느 영상처리 프로그램에서나 자주 쓰이지만 꼭 함께 쓰이는 것은 아니므로 이식성 향상을 위하여 분리하여 구현한다.

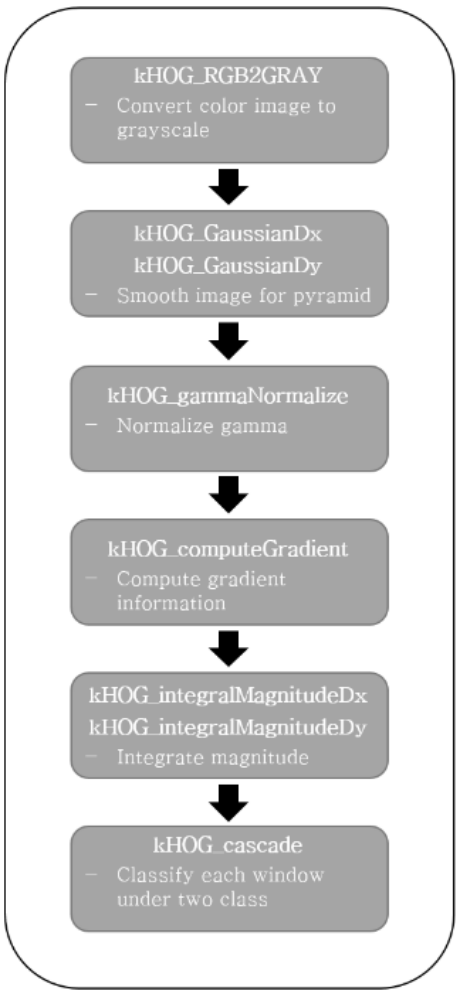

그림 5. 보행자 검출 OpenCL 커널 구조 및 역할
Fig. 5. Structure of OpenCL kernel and its function for pedestrian detection.

그림 6. 보행자 검출 OpenCL 각 커널의 결과 (왼쪽 위부터 시계방향으로 RGB2GRAY, gammaNormalize, computeGradient, cascade)
Fig. 6. Result of each kernel of OpenCL for pedestrian detection (Clockwise from top left : RGB2GRAY, gammaNormalize, computeGradient, cascade).

2) 지역 메모리 사용 등으로 작업 그룹 크기를 다른 커널들과 다르게 적용하는 경우 분리한다. 특히, 수직 방향과 수평 방향으로 나누어 구현할 경우 각 방향에 대해서도 작업 그룹 크기가 다르기 때문에 커널을 분리할 수밖에 없다.

이밖에도 커널의 적절한 분리는 커널 역할의 이해와 커널 소스코드 관리에도 긍정적인 영향을 줄 수 있다.

4. 최적화 및 근사화 요소

GPU 기반의 OpenCL 병렬처리는 기본적으로 pixel 별 동시 연산을 하지만, 더 효율적인 연산을 할 수 있도록 알고리듬에 따라 최적화 및 근사화를 진행해주어야 한다. 본 연구에서는 다음과 같은 최적화와 근사화를 진행하였다.

1) 반복문 unrolling, 조건문 분리 분기 및 커널 최적화 빌트인 함수 및 데이터 타입 활용
- 이 항목은 가장 기본적인 최적화 방법에 해당한다. 하지만 커널 내에서의 반복문 unrolling의 경우 하나의 반복문을 네 부분으로 나눈 것만으로도 50% 수준의 수행 시간 감소를 가져올 수 있었다. GPU는 부동소수점 연산에 최적화되어 있고 벡터 데이터 타입은 연산 자동 병렬화에 큰 도움을 준다. OpenCL 커널 내에서는 이러한 장점을 살린 여러 빌트인 함수들과 데이터 타입이 사용가능하다.

2) atan 함수 대신 look-up table 작성
- 이 항목은 기본적인 근사화 방법이라고 할 수 있다. HOG 특징 추출 중 기울기 정보를 구할 때 atan이나 atan2 함수를 쓰는 것이 일반적이지만 모바일 GPU에서 이 함수들은 각각 30ms와 500ms의 처리 시간을 소모한다. 하지만 LUT를 사용하면 10ms대로 처리 시간을 낮출 수 있으니 약간의 정확도를 희생하는 것도 필요하다.

3) 최적화된 작업 그룹 크기 적용
- OpenCL을 이용할 때 가장 중요한 것 중의 하나가 바로 작업 그룹 크기 지정이라 할 수 있다. 이 작업 그룹 크기가 동시에 커널을 실행시키는 작업 그룹의 개수이므로 병렬 프로그램의 처리 속도와 직결된다. 이 작업 그룹의 크기는 무한대로 지정할 수 있는 것이 아니고 최대 지원하는 작업 그룹의 크기는 기기마다 다르므로 OpenCL 관련 정보를 먼저 확인해야 한다. 커널에 관련된 질의 중 'clGetKernelWorkGroupInfo'의 'CL_KERNEL_PREFERRED_WORK_GROUP_ SIZE_MULTIPLE' 파라미터를 사용하면 커널마다 성능을 최대로 낼 수 있는 작업 그룹 크기의 최소치를 알려준다. 이 수의 배수로 작업 그룹의 크기를 지정해 주는 것이 좋다.

4) 지역 메모리를 이용한 integral magnitude 연산
- 지역 메모리는 동일 작업 그룹 내에 있는 작업 아이템들만이 공유할 수 있는 작은 메모리인데, 이는 전역 메모리보다 작은 대신에 더 빠르다. HOG 특징 추출 중에 특정 영역의 합을 구해야하는 연산이 반복적으로 필요한데, integral image와 같은 방식으로 지역 메모리를 활용하여 magnitude의 누적 합들을 미리 구해놓았다.

5) 반복 사용하는 메모리를 상수로 지정
- LUT처럼 반복 사용하는 메모리는 호스트에서 생성하여 커널로 복사하기 보다는 커널 내에서 상수 타입으로 생성하는 것이 빠르다. 452Byte 크기의 메모리라도 이 방식으로 25ms의 처리 시간을 아낄 수 있었다. 다만 상수 메모리는 최대 8개, 4096byte (Qualcommm Adreno330 GPU의 경우)까지만 생성할 수 있는 등 제약 조건이 따르므로 기기마다 메모리 정보를 확인하여야 한다.

6) '-cl-fast-relaxed-math' 등 OpenCL 프로그램 빌드 최적화 옵션 사용
- 프로그램 빌드 옵션에는 여러 가지가 존재한다. 그 중 수학관련 옵션들이 있는데, 'cl-fast-relaxed-math' 옵션을 사용하였다. 이 옵션은 '-cl-finite-math-only'와 '-cl-unsafe-math-optimizations' 옵션이 결합된 것인데, 각각 사용되는 실수와 연산 결과가 NaN이나 ±∞가 아니라고 가정, 실수 연산의 에러 체크를 없애는 등으로 최적화하는 것이다. 이러한 옵션을 사용할 때에는 해당 오류가 발생하지 않도록 주의하여 프로그램을 작성해주어야 한다.

Ⅲ. 실험 결과

1 실험 환경

구현에 사용한 스마트폰은 Android 4.2.2 기반의 Samsung Galaxy S4 LTE-A 모델이며, 해당 기기는

2.3GHz의 Qualcomm Krait 400 CPU 및 Adreno 330 GPU 그리고 2GB의 메인 메모리와 13MP의 카메라 및 1920×1080 해상도의 디스플레이를 내장하고 있다. 실험을 위한 카메라 동작 및 동영상 재생은 Java 언어 기반의 Android SDK를 이용하였고, OpenGL ES 2.0 기반의 영상 입출력 및 OpenCL 1.1 기반의 병렬영상처리 애플리케이션을 구현하기 위하여 C/C++ 언어 기반의 Android NDK를 이용하였다. 또한 desktop 개발 환경은 i7-3770 CPU (3.40GHz) 및 8GB 메인메모리, OS는 Windows 7 (64bit)이다.

2. 성능 평가

성능 평가를 위하여 USCPedestrianSet을 사용하였고, 객관적인 비교 평가를 위하여 OpenCV4Android의 hogcascade 함수 결과와 비교하였다. 스마트폰에서의 보행자 검출 결과는 그림 7과 같고 성능 비교 평가 결과는 표 1과 같다.

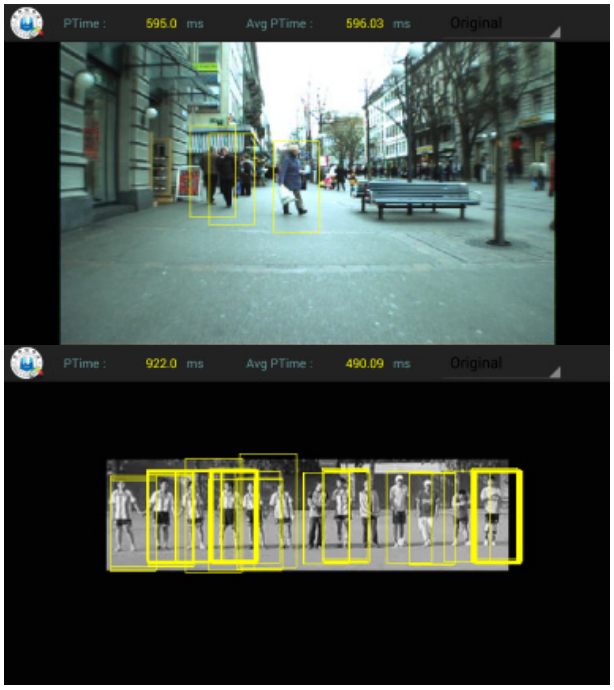

그림 7. 모바일 GPU OpenCL을 이용한 보행자 검출 결과
Fig. 7. Result of pedestrian detection using OpenCL based on mobile GPU.

표 1. 보행자 검출 CPU 및 GPU 성능 비교 평가표
Table 1. Performance comparative table between CPU and GPU about pedestrian detection.

분류	Device	Accuracy (%)	Time (msec)
OpenCV	Desktop CPU	44.40	19.17
	Mobile CPU	44.40	639.22
OpenCL	Mobile GPU	40.95	482.16

Ⅳ. 결론 및 향후 연구 방향

본 논문에서는 ADAS의 보급화를 위해 상용 모바일 CPU 및 GPU를 장착한 스마트폰을 이용한 보행자 검출을 제안하였다. GPU의 연산능력을 적극 활용하기 위하여 OpenCL 병렬처리 라이브러리를 이용하였고 병렬연산의 효율성 극대화를 위하여 최적화를 진행하였다. 그 결과 OpenCV에 비해 3.45% 낮은 정확도를 보였지만 모바일 CPU 대비 약 25%의 속도 향상을 보였다. 이는 정확히 같은 알고리듬으로 비교한 것이 아니기 때문에 정확한 비교라고 보기엔 어렵다. 또한, 본 연구의 실험에서 사용한 해상도에서는 월등한 속도 차이를 보이진 않았지만 앞으로 해상도가 커질수록 CPU와 GPU 연산의 속도 차이가 커질 것이라 기대할 수 있다. 향후에는 GPU만이 아닌 CPU의 멀티 코어를 함께 사용하여 수행 속도를 향상시키고 다중 특징을 사용하는 연구를 진행할 것이다.

REFERENCES

[1] International Data Corporation, http://www.idc.com/getdoc.jsp?containerId=prUS23982813
[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 886-893, San Diego, United States; 2005/6.
[3] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, I-511-I-518, vol.1, Kauai, United States; 2001/12
[4] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach Learn*, vol. 20, no. 3, pp. 273-297, 1995.
[5] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J*

Comput Syst Sci, vol. 55, no. 1, pp. 119-139, 1997.

[6] P. Geismann and G. Schneider, "A two-staged approach to vision-based pedestrian recognition using Haar and HOG features," *Proc. Intelligent Vehicles Symposium*, IEEE554-559, Eindhoven, Netherlands; 2008/6.

[7] Q. Zhu, S. Avidan, M. C. Yeh, and K. T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1491-1498, New York, United States; 2006/6.

[8] S. Kim, D. G. Yoo, and Y. H. Kim, "High Performance Pedestrian Detection System Using A Cascade Algorithm Structure," *Proc. IEEK Conference on System-on-Chip*, 91-94; 2011/4.

[9] J. Lee, S. H. Kang, M. H. Lee, S. Li, H. Kim, and I. K. Park, "Real-Time Parallel Image Processing Library using Mobile GPU," *Journal of KIISE : Computing Practices and Letters*, vol. 20, no. 2, pp. 96-100; 2014/2.

[10] H. Jia, Y. Zhang, W. Wang, and J. Xu, "Accelerating Viola-Jones facce detection algorithm on GPUs," *Proc. the 14th IEEE International Conference on High Performance Computing and Communications*, HPCC-2012 - 9th IEEE International Conference on Embedded Software and Systems, ICESS-2012396-403, Liverpool, United Kingdom; 2012/6.

[11] I. K. Park, M. H. Lee, and Y. K. Choi, "Technical Development Trend of Computer Vision on Embedded Platform," *The Magazine of the IEEK*, vol. 39, no. 2, pp. 85-92; 2012/2.

저 자 소 개

김 종 희(학생회원)

2013년 인하대학교 정보통신 공학부 학사 졸업.

2013년~현재 인하대학교 정보통신공학부 석사 과정.

<주관심분야 : 컴퓨터비전, 자동차비전, 병렬영상처리, 임베디드영상처리>

이 충 수(학생회원)

2013년 인하대학교 정보통신 공학부 학사 졸업.

2013년~현재 인하대학교 정보통신공학부 석사 과정.

<주관심분야 : 컴퓨터비전, 자동차비전, 병렬영상처리, 모바일영상처리>

김 학 일(평생회원)

1983년 서울대학교 제어계측 공학과 졸업.

1985년 Purdue Univ. 전기/컴퓨터 공학과 석사 졸업.

1990년 Purdue Univ. 전기/컴퓨터 공학과 박사 졸업.

1990년~현재 인하대학교 정보통신공학부 교수.

<주관심분야 : 패턴인식, 컴퓨터비전, 바이오인식, 로봇비전, 의료영상처리>