ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 14-01-12
106

Journal of IKEEE.Vol.18,No.1,106~113,March 2014
http://dx.doi.org/10.7471/ikeee.2014.18.1.106

Advanced SIMD 아키텍처에서의 HOG 보행자 검출기 고속화 방법

A Speed-up Method of HOG Pedestrian Detector in Advanced SIMD Architecture

권 기 표*, 이 재 흥*★

Ki-Pyo Kwon*, Jae-Heung Lee*★

Abstract

A pedestrian detector can be applied for various purposes such as monitoring or counting the number of people in some place, or detecting the people plunging in the driveway. There was a lot of related research. But, the detection speed is slow in embedded system because of the limited computing power. An algorithm for fast pedestrian detector using HOG in ARM SIMD architecture is presented in this paper. There is a way to quickly remove the background of image and to improve the detection speed using NEON parallel technique. When we tested with INRIA Person Dataset, the proposed pedestrian detector improves the speed by 3.01 times than previous one.

요 약

보행자 검출기는 보안이 필요한 곳에서 모니터링을 하거나 특정 장소를 드나드는 사람의 수를 셀 때, 운전 중 차도에 뛰어드는 사람을 감지할 때 등 상황에 따라 여러 목적으로 응용될 수 있다. 이와 관련한 연구는 많이 진행되어 왔지만, 임베디드 시스템에서는 제한된 컴퓨팅 능력으로 인해 검출 속도가 느리다는 문제가 있다. 본 논문에서는 입력 영상에서 배경 부분을 빠르게 제거하여 검출 속도를 향상하는 방법과 ARM SIMD 아키텍처에서 NEON 병렬화 기법을 이용하여 검출 속도를 향상하는 방법을 제시한다. 제시한 방법으로 구현한 검출기는 INRIA Person Dataset을 이용하여 테스트한 결과 기존에 비해 3.01배의 향상된 속도를 나타냈다.

Key words : HOG(Histograms of Oriented Gradients), Pedestrian Detector, ARM, SIMD, NEON

Ⅰ. 서론

최근 사람의 얼굴이나 물체를 인식하고 움직임을 추적하는 컴퓨터 비전(Computer Vision) 기술이 급속히 발달하고 있으며, 보행자 검출 기술 또한 많은 연구가 진행되고 있다. 보행자 검출기는 보안이 필요한 곳에서 모니터링을 할 때, 특정 장소를 드나드는 사람의 수를 셀 때, 스마트 자동차에 탑재되어 운전 중 차도에 뛰어드는 사람을 감지할 때 등 상황에 따라 유용하게 응용될 수 있다.

* Dept. of Computer Engineering, Hanbat University
010-6325-4957, codek210@gmail.com

★ Corresponding author 042-821-1208 jhlee@hanbat.ac.kr

※ Acknowledgment
This research was financially supported by the Ministry of Education (MOE) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation (No. 2012H1B8A2026119).
Manuscript received Jan. 27, 2014; revised Mar. 03, 2014; accepted Mar. 04. 2014

이에 따라, 검출의 정확도를 높이기 위해 많은 연구들이 진행되어 왔다. 현재 템플릿 기반, 통계/분류 기반, 특징 기반의 다양한 검출 방법이 존재한다. 기계 학습 방식은 특징 기반 혹은 템플릿 기반 방식과 결합하여 높은 인식률을 보여주고 있다.

하지만 검출 시스템은 많은 연산이 필요하기 때문에 정확도뿐만 아니라 검출 속도도 중요한 이슈가 된다. 최근에는 PC 환경에서 이와 관련한 실시간 처리가 가능해졌다. 하지만, 임베디드 시스템에서는 적은 메모리, 낮은 클록 수, 제한적인 부동 소수점 지원이라는 하드웨어 특징 때문에 여전히 느리다. 부동 소수점 연산량이 많으면서 많은 메모리를 사용하고 실시간 처리를 요구하는 상반된 특징의 컴퓨터비전 기술은 임베디드 시스템에서 도전적인 분야이다.

본 논문에서는 임베디드 시스템에서 가장 많이 사용되고 있는 ARM 아키텍처에서 고속 보행자 검출을 위한 방법을 제시한다. 제시하는 방법은 영상 내 배경 부분을 매우 간단한 연산으로 제거해내면서 빠르게 보행자를 검출한다. 또한, ARM SIMD 아키텍처에서 NEON 병렬화 기법을 적용하여 보행자 검출 속도를 향상시킨다.

II. 관련 기술

1. HOG

HOG(Histograms of Oriented Gradients)는 현재의 영상 보행자 검출기 기술에 있어 가장 널리 사용되고 있는 특징 벡터 중 하나이다. 어떤 물체 영상이 갖는 지역적인 그래디언트 분포 특성을 이용하여 물체를 식별하는 방법인데, 보행자뿐만 아니라 승용차, 트럭 등 다양한 물체 검출에 적용할 수 있는 일반적인 방법론이다.[1]

HOG는 그래디언트 크기와 에지 방향의 분포를 특징으로 하는 히스토그램이다. 그림 1은 HOG를 이용한 보행자 검출 과정 전체를 나타낸다. 표준 방법에 따르면, '윈도우(Window, 64×128픽셀)' 내에서 그래디언트의 크기와 방향을 이용하여 '셀(Cell, 8×8픽셀)' 단위로 히스토그램을 만들고, '블록(Block, 2×2셀)' 단위로 히스토그램을 정규화 하여, 이들의 집합을 특징 벡터로써 사용한다. HOG는 다른 특징 정보들에 비해 사람 검출 능력이 뛰어나지만, 영상 전체를 좁은 간격의 슬라이딩 윈도우 방식으로 검색하면 매우 많은 시간이 걸린다는 단점이 있다.

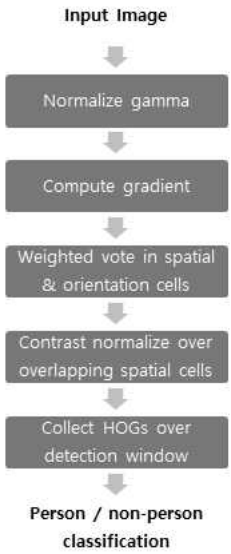

Fig. 1. Pedestrian detection process using HOG
그림 1. HOG를 이용한 보행자 검출 과정

가. 그래디언트 계산

HOG는 기본적으로 가로 64픽셀 × 세로 128픽셀의 윈도우를 사용한다. 그래디언트는 윈도우 내 가장자리를 제외한 모든 픽셀에서 계산된다. 그래디언트를 계산하기 위해 uncentered, centered, cubic-corrected, 3×3 Sobel masks, 2×2 diagonal mask 등이 사용될 수 있지만, 단순한 centered (-1, 0, 1) 마스크가 가장 성능이 좋다. 그래디언트는 아래 수식과 같이 계산할 수 있다. d_x와 d_y는 각각 가로와 세로 방향의 그래디언트이며, $I(x,y)$는 좌표 (x,y)에서의 픽셀 값이다.

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

$$d_y(x,y) = I(x,y+1) - I(x,y-1) \quad (2)$$

좌표(x,y)에서 그래디언트 크기 $m(x,y)$와 방향 $\theta(x,y)$은 다음과 같이 계산할 수 있다.

$$m(x,y) = \sqrt{d_x(x,y)^2 + d_y(x,y)^2} \quad (3)$$

$$\theta(x,y) = \tan^{-1}\frac{d_y(x,y)}{d_x(x,y)} \quad (4)$$

RGB, LAB와 같이 다채널 영상을 사용하는 경우, 각 픽셀에서 모든 채널에 대해 그래디언트를 계산한 후 가장 큰 그래디언트 값을 가지는 것을 해당 픽셀의 그래디언트로 사용한다.

나. 히스토그램 생성

히스토그램은 윈도우 내 가로 8픽셀 × 세로 8픽셀의 셀마다 하나씩 만든다. 그림 2와 같이 0°~180°를 부호 구분 없이 9개의 각도로 균등하게 나눈 bins에, 셀 내의 각 픽셀들이 그래디언트 각도에 따라 투표한다. 각 투표는 그래디언트 크기에 의해 가중된다.

검출기의 정확도를 위해, 각도와 위치에 대해 인접 구간들의 중심점들 사이에서 보간법을 적용한다. 만약 어떤 픽셀에서 그래디언트 각도가 85°이고 그래디언트 크기가 100이라면, 80°~100° bin에 75만큼 가중 투표하고, 60°~80° bin에 25만큼 가중 투표한다.

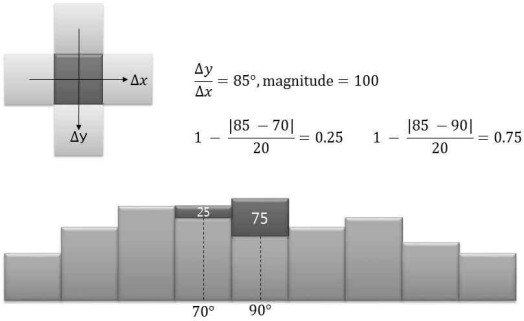

Fig. 2. Histograms with interpolation
그림 2. 보간법을 적용한 히스토그램

다. 히스토그램 정규화

히스토그램은 그림 3에 나타나는 블록 단위로 정규화 한다. 각 블록은 가로 2셀 × 세로 2셀이 슬라이딩 방식으로 이루어진다. 따라서 각 블록은 서로 겹치게 되며, 한 블록 당 36개(4셀 × 9bins)의 특징 정보를 가진다. 가로 64픽셀 × 세로 128픽셀을 가지는 윈도우는 총 7×15개의 블록으로 이루어진다.

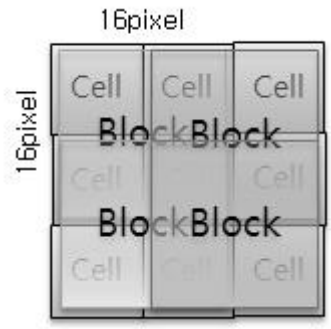

Fig. 3. Block (2×2cell, overlapping)
그림 3. 블록 (2×2셀, 오버래핑)

히스토그램을 정규화하지 않을 경우, 같은 장면의 영상이 대비(Contrast)에 따라 다른 그래디언트 크기를 갖게 된다. 따라서 정규화 과정이 반드시 필요한데, L2-norm, L2-Hys, L1-norm, L1-sqrt 등을 이용하여 정규화 할 수 있다. L2-norm을 이용할 경우 다음과 같이 정규화 할 수 있다. $\|v\|_2$는 v의 L2-norm을 나타낸다.

$$v \rightarrow \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \qquad (5)$$

라. 특징 벡터 수집

윈도우 내 모든 블록의 특징 정보를 합치면 특징 벡터가 된다. 그림 4는 특징 벡터 수집 과정을 보여준다. 블록 수가 7×15이고, 블록 당 특징 정보의 수가 4(블록 내 셀의 수) × 9(bin의 수)이면, 총 특징 정보의 수는 3780개가 된다. 이 특징 벡터는 학습기와 분류기의 입력으로 사용되어, 보행자를 검출하는데 사용된다.

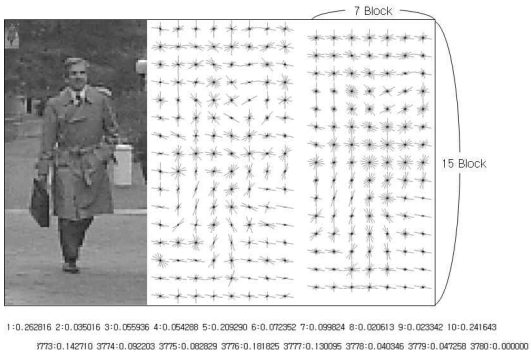

Fig. 4. HOG feature vector collection
그림 4. HOG 특징 벡터 수집

2. Integral Image

빠른 사람 검출을 위해 HOG에 적분 이미지가 결합될 수 있다.[2] 기존에는 윈도우를 슬라이딩함으로써 같은 픽셀에 대한 중복된 연산이 불가피하였으나, 이 방법은 초기에 적분 이미지를 생성해 놓으면 각 셀에 대해 단 4×9회의 데이터 접근으로 히스토그램을 만들며, 중복 연산을 피할 수 있다. 적분 이미지는 그래디언트의 방향에 따라 각각 만든다. 즉, 0°~20°의 그래디언트 방향성을 갖는 픽셀들 간의 적분 이미지를 생성하고, 마찬가지로 20°~40°, 40°~60°…, 160°~180°의 그래디언트 방향성을 갖는 픽셀들 간의 적분 이미지를 생성한다. 이후 셀에서 각 방향에 대한

그래디언트 강도는 4회의 데이터 접근으로 계산될 수 있다. 초기에 적분 이미지를 생성하는데 시간이 다소 걸리지만, 그 이후에 중복된 연산을 다수 피함으로써 결과적으로 윈도우 전체를 스캔하는 속도는 더 빨라지게 된다.

3. SVM

SVM(Support Vector Machine)은 두 부류 사이에 존재하는 여백을 최대화하여 일반화 능력을 극대화하는 분류기이다. SVM은 크게 학습과 분류로 나눌 수 있다.[3]

가. SVM 학습

SVM 학습은 두 부류를 분류하기 위한 결정초평면을 구하는 과정이다. SVM에서 결정 초평면은 다음과 같은 형태로 표현된다.

$$d(x) = w^T x + b = 0 \qquad (6)$$

결정초평면으로부터 가장 가까이 위치한 샘플들을 서포트 벡터(Support Vector)라 하며, N개의 서포트 벡터를 가진 결정초평면을 구하는 문제는 수학적으로 다음과 같이 표현할 수 있다.

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j x_i^T x_j \qquad (7)$$

나. SVM 분류

SVM 분류는 학습을 통해 얻은 결정초평면을 이용하여, 입력 데이터를 두 부류 중 하나로 알맞게 인식하는 과정이다. 그림 5는 SVM 분류를 나타낸다. w는 SVM 학습의 결과 벡터이고, x는 입력영상으로부터 얻은 특징벡터이다. 결정초평면의 식에 두 벡터를 대입하여 $d(x) > 0$이면 보행자로 판별하며, $d(x) < 0$이면 보행자가 아닌 것으로 판별한다.

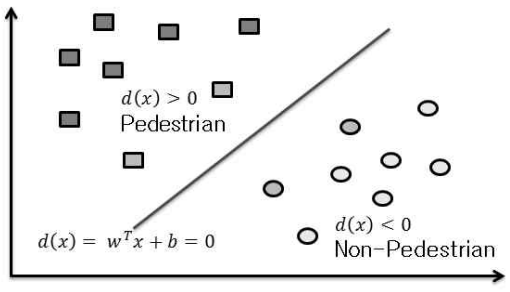

Fig. 5. SVM Classification
그림 5. SVM 분류

4. NEON

NEON은 Cortex-A8 프로세서의 핵심 기술 중 하나로써, 64·128비트 SIMD(Single Instruction Multiple Data) 명령어 집합이다.[4] 오디오, 비디오, 이미지 및 기타 데이터 처리를 다루는 8비트, 16비트, 32비트, 64비트 정수 및 단정밀도 부동 소수점(float) SIMD 연산을 지원한다. NEON은 ARM 정수 파이프라인과 독립된 별도 레지스터와 파이프라인을 사용한다.

많은 양의 데이터에 동일한 연산을 수행하는 애플리케이션에서 NEON의 효율성이 극대화 된다. NEON이 멀티미디어 코덱에 적용된 경우, 코덱의 종류에 따라 최소 35%에서 최대 100%의 성능 향상을 보여준다.

하지만 NEON 기술을 실제로 사용하기 위해서는 일반적인 프로그래밍 방법과는 차별화된 방법이 필요하다. NEON 기술을 사용하는 방법은 NEON 어셈블리 명령어를 사용하는 방법, NEON C언어 Extension을 사용하는 방법으로 나눌 수 있다. NEON 어셈블리 명령어를 사용하는 방법은 가장 효과적으로 NEON 기술을 사용할 수 있지만 구현이 매우 어렵다는 단점이 있다. NEON C언어 Extension은 NEON 어셈블리 명령어들을 사용하도록 만들어진 C언어의 부가적 요소라고 할 수 있으며, ARM에서 직접 제공하는 방법이기 때문에 신뢰할 수 있는 방법이다. 이 방법은 C/C++ 환경에 익숙한 루프와 조건문, 코드의 가독성, 보다 나은 디버깅 환경을 갖는다.

NEON은 스칼라 형식의 연산이 아닌 벡터 형식의 연산을 사용한다. 따라서 이를 위한 벡터 데이터 유형과 내장 함수가 제공된다.

Table 1. NEON 벡터 데이터 유형
표 1. NEON Vector Data Type

int8x8_t	int8x16_t
int16x4_t	int16x8_t
int32x2_t	int32x4_t
int64x1_t	int64x2_t
uint8x8_t	uint8x16_t
uint16x4_t	uint16x8_t
uint32x2_t	uint32x4_t
uint64x1_t	uint64x2_t
float16x4_t	float16x8_t
float32x2_t	float32x4_t
poly8x8_t	poly8x16_t
poly16x4_t	poly16x8_t

III 본 논문의 보행자 검출기 고속화 방법

1. 신속한 배경 윈도우의 제거

HOG를 이용한 보행자 검출은 많은 시간이 걸리는데, 주된 이유 중 하나는 스캔해야 할 윈도우의 수가 많기 때문이다. 입력 영상의 크기가 가로 320픽셀 × 세로 240픽셀, 윈도우의 크기가 가로 64픽셀 × 세로 128픽셀이라고 할 때, 가로·세로 8픽셀씩 윈도우를 슬라이딩 한다고 가정하면, 가로 방향으로 33개의 윈도우, 세로 방향으로 15개의 윈도우, 총 495개의 윈도우에서 보행자가 있는지 없는지를 판별해야 한다.

HOG는 지역적인 그래디언트 강도와 에지 방향의 분포를 특징으로 삼는다. 여기서 중요한 사실은, 사람은 기본적으로 그래디언트 강도가 매우 큰 Object라는 것이다. 만약 어떤 한 윈도우가 가진 전체 픽셀의 그래디언트 강도 I_w가 일정 수준 이하로 낮다면, 그 윈도우는 배경으로 생각할 수 있다. 배경을 미리 가려낼 수 있다면, 해당 윈도우에서 특징 추출 및 SVM 분류의 과정을 거치지 않음으로써, 전체 영상의 스캔 속도를 빠르게 할 수 있다.

I_w는 적분 이미지를 사용한다면, 간단하게 계산할 수 있다. 윈도우 내 셀들에 대해서 히스토그램을 생성하기 전에, 그림 6과 같이 히스토그램 생성 연산을 셀 범위가 아닌 '윈도우 범위'에 대해 1회 수행한다. 하나의 에지 방향 적분 이미지로부터 윈도우 내 해당 에지 방향을 가진 모든 픽셀들의 그래디언트 강도의 합을 구할 수 있다. 전체 9방향, 즉 적분 이미지 9개를 동일한 방식으로 구할 수 있고, 그 합이 I_w가 된다. 단 36회의 데이터 접근만으로 I_w가 계산된다.

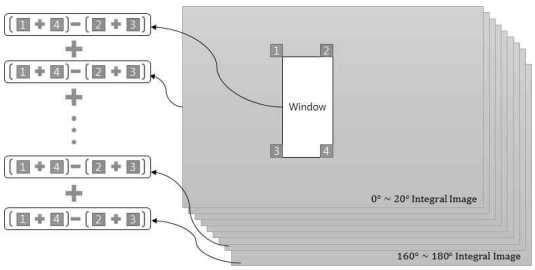

Fig. 6. Window's gradient magnitude computation
그림 6. 윈도우 그래디언트 강도 계산

SVM 학습 시 Positive Data Set에서 계산된 모든 윈도우의 I_w 중 최소값을 T_{train}으로 둔다. 그리고 T_{train}보다 약간 작은 값 T_w를 설정한다. T_w는 검출기가 윈도우를 스캔할 때, I_w와 비교되는 값이다. 여기에서 T_w 값이 너무 작다면 속도 향상은 거의 일어나지 않는다. 반대로, T_w가 T_{train}과 거의 같다면 학습시의 영상에 대해서는 보행자를 완벽하게 검출하지만, 이후에 취득하는 새로운 영상에서는 실제 보행자가 있음에도 검출을 하지 못하는 경우가 발생할 수 있다. 본 연구에서는 T_w를 T_{train}의 0.9배로 설정하였고, 이 값이 검출기의 정확도에 영향을 주지 않으면서 속도를 향상시킬 수 있음을 확인하였다. T_w는 SVM 학습 결과인 결정초평면을 기록하는 파일에 함께 저장된다.

보행자 검출기가 윈도우를 스캔할 때, 각각의 윈도우에서 I_w와 T_w가 비교되는데, 이 과정을 Coarse Detection이라 한다. I_w가 T_w보다 작을 시, 해당 윈도우는 배경으로 생각하고 다음 윈도우로 슬라이딩할 수 있다. I_w가 T_w보다 크다면, 특징 추출 및 SVM 분류 과정을 거쳐 보행자 검출을 진행하는데, 이 과정을 Fine Detection이라 한다.

제안하는 방법은 셀에 대해 수행되던 연산을 윈도우에 대해 한 번 수행함으로써 배경 윈도우를 걸러내는 것이다. 즉 Coarse Detection을 통해 배경 윈도우를 걸러내면서, 배경 부분에 대해서는 Fine Detection을 수행하지 않는다.

2. NEON 병렬화 기법을 이용한 고속화

보행자 검출 알고리즘에는 여러 픽셀에 대해 동일한 연산을 하는 작업이 매우 많기 때문에, NEON을 적용할 경우 상당한 속도 향상을 가져올 수 있다. NEON 병렬화 기법은 입력 영상의 전처리 과정이나 그래디언트 이미지를 만드는 과정, HOG 특징을 추출하는 과정, SVM 분류를 하는 과정에 각각 적용이 가능하다.

그래디언트 G_x와 G_y는 영상 내 모든 픽셀에 대해 각각 가로 방향과 세로 방향으로 (-1, 0, 1) 마스크를 적용하여 구한다. 이와 같은 연산은 그림 7과 같이 NEON을 이용하여 데이터 수준의 병렬화가 가능하다. 먼저, 메모리 상 연속된 픽셀 데이터 8개를 한번에 Load 연산으로 NEON 구조체에 담는다. 그 후 두 개의 구조체를 Subtract 연산으로 뺄셈을 하면, 각 성분 8개 각각의 뺄셈이 동시에 수행된다. 마지막으로 결과 구조체를 Save 연산으로 일반 변수에 저장한다.

병렬화를 위해 필요한 구조체에 데이터를 적재하는 과정과 연산 결과를 원래의 형태로 변환하는 과정 등 부수 작업이 추가되기는 하지만, 병렬화로 인한 연산

속도는 이를 상쇄하고도 더 빠르다. 이를 위해서는 연산에 필요한 데이터가 메모리상에 연속되도록 유의할 필요가 있다.

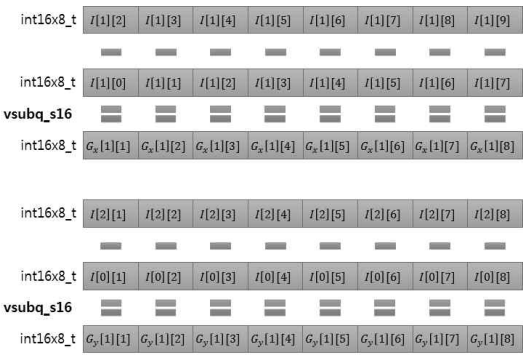

Fig. 7. Gradient computation using NEON
그림 7. NEON을 이용한 그래디언트 계산

그래디언트의 방향의 크기와 방향 또한 위와 유사한 방식으로 구할 수 있다. 방향을 구하기 위해서는 아크탄젠트 값을 계산하는 부동소수점 연산이 필요한데, NEON은 32개의 하드웨어 FPU 64비트 레지스터를 지원하기 때문에 빠른 연산이 가능하다.

SVM 분류는 $d(x)=w^t x+b$ 를 구하는 과정으로 두 벡터의 내적을 구하는 연산이 포함되어 있다. w와 x의 성분은 32비트 부동소수점을 갖는 데이터이고 NEON은 최대 128비트까지 연산이 가능하므로, 각 성분을 동시에 네 개씩 처리할 수 있다. 벡터의 내적 연산은 그림 8과 같이 구현할 수 있다. 벡터의 모든 성분을 각 네 개씩 적재하여 곱-누적 연산을 하면, 두 벡터 성분의 곱 결과는 Acc의 32비트 네 개 성분에 나뉘어 누적이 된다. 따라서 Acc의 성분 네 개를 합하면 결과적으로 w와 x의 내적을 구할 수 있다.

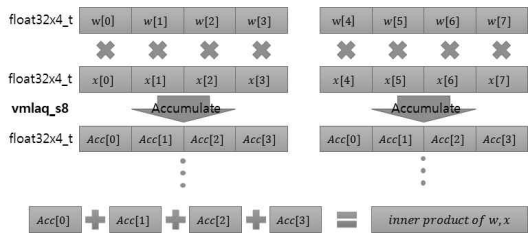

Fig. 8. SVM classification using NEON
그림 8. NEON을 이용한 SVM 분류

Ⅳ 실험 및 고찰

1. 실험 환경

분류기 학습은 PC 환경에서 SVM Light와 MIT Pedestrian Database, INRIA Person Dataset을 이용하였다. 선형 SVM을 사용하였고, 검출 프로그램 실행 시 초기화 과정에 걸리는 시간을 없애기 위해, 모델 파일에서 미리 결정초평면을 계산하도록 수정하였다. 학습의 최종 결과물은 결정초평면과 T_w가 저장된 데이터 파일이다.

타겟 환경은 그림 9의 ODROID-X (프로세서 Exynos4412 Cortex-A9 Quad Core 1.4GHz, 메모리 1GB)이며, 안드로이드 플랫폼 위에서 동작하는 애플리케이션을 작성하였다. 보행자 검출 알고리즘은 모두 C언어로 구현하였으며, JNI와 NDK를 이용하여 애플리케이션에 적용하였다.

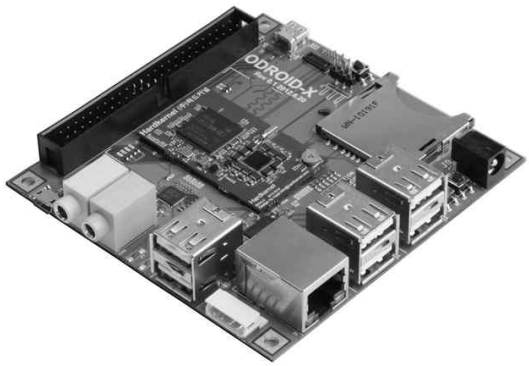

Fig. 9. Target Board (ODROID-X)
그림 9. 타겟 보드 (ODROID-X)

2. 실험 결과

보행자 검출기의 검출율 및 정확도는 표 2와 같다. 검출율 및 정확도를 측정하기 위해서 MIT CBCL Pedestrian Database, INRIA Person Dataset을 이용하였다. Recall rate는 입력 데이터에 있는 전체 보행자들 중에서 성공적으로 검출된 보행자 비율이고, Precision은 검출기가 검출한 결과들 중 정검출의 비율이다. 검출율 및 정확도는 보다 정교한 학습 과정을 통해, 현재보다 나은 결과를 얻을 수 있다.

Table 2. Recall rate and Precision
표 2. 검출율 및 정확도

Recall rate	96.08%
Precision	98.07%

그림 10은 INRIA Person Dataset에서 640×480 크기의 Positive Set만을 선택하여, 이미지를 320×240 크기로 스케일링 하며 얻은 검출 결과를 보여준다. 윈도우의 크기는 64×128이고 윈도우를 8픽셀씩 슬라이딩 하여, 영상 내 윈도우는 총 495개이다. Coarse Detection을 적용하면 영상의 복잡한 정도에 따라, Fine Detection을 수행하는 횟수가 달라진다. 그에 비례하게 윈도우가 슬라이딩 하는 속도가 빨라지고, 전체 검출 시간도 줄어들게 된다. INRIA Person Dataset의 경우 테스트 결과, 평균 147개의 윈도우를 배경 윈도우로 걸러내고 348개의 윈도우에 대해 Fine Detection을 하였다. 그 결과 1.27배의 향상된 속도를 보여주었으며, 비교적 한적한 배경을 갖는 이미지에서는 두 배 이상 빠른 결과를 보여주기도 하였다.

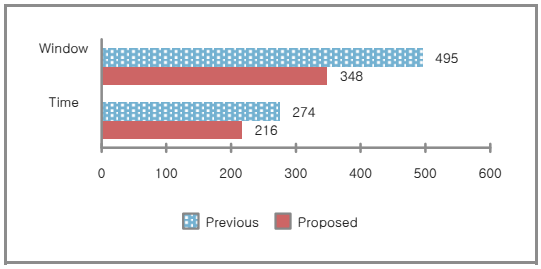

Fig. 10. Detection result with Coarse Detection
그림 10. Coarse Detection 적용 검출 결과

그림 11과 그림 12는 검출 각 단계별로 NEON 병렬화 기법을 적용한 보행자 검출 결과이다. YUV to RGB 변환과 스케일링 작업을 하는 전처리 부분, 영상으로부터 방향별 그래디언트 이미지를 만드는 부분, 각 윈도우에서 HOG 특징을 추출하는 부분, 추출한 특징으로 SVM 분류를 하는 부분에서 각각 속도 향상이 일어났다. 제시한 방법을 적용하였을 경우, 병렬화 하지 않았을 때보다 2.37배 빠른 속도를 보여주었다.

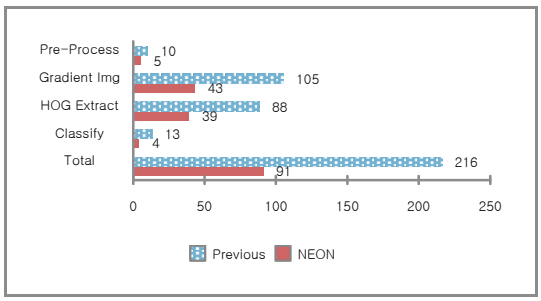

Fig. 11. Detection time with NEON parallel technique
그림 11. NEON 병렬화 기법 적용 검출 시간
(단위 : ms)

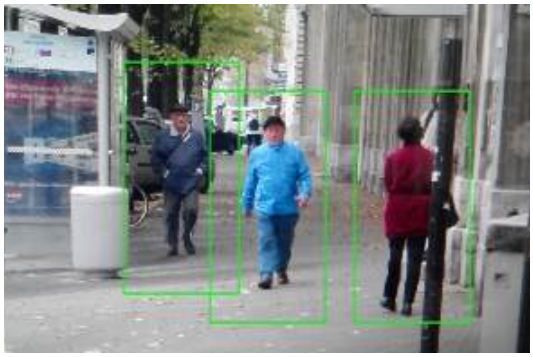

Fig. 12. Pedestrian detection using INRIA Person Dataset
그림 12. INRIA Person Dataset을 이용한 보행자 검출

V 결론

본 논문에서는 임베디드 시스템 위에서 HOG 특징 정보와 SVM 분류 기술을 사용하여 보행자 검출을 고속화 하는 방법을 제시하였다. 간단하지만 유용한 약분류 특징을 이용하여 영상 내 배경에 대해 불필요한 특징 추출 및 분류 작업을 피함으로써, 검출 속도를 1.27배 향상시켰다. 또한, 임베디드 시스템에서 가장 많이 쓰이고 있는 ARM 계열의 CPU에서 NEON 병렬화 기법을 적용하여 하나의 명령어로 다수의 픽셀을 한꺼번에 연산하도록 처리하여 하드웨어적인 이점을 활용할 수 있도록 함으로써, 검출 속도를 2.37배 향상시켰다. 본 논문에서 제시한 방법을 모두 사용하면, 기존보다 3.01배까지 검출 속도를 향상시킬 수 있다. 이를 이용하여 640×480 영상을 초당 10프레임 이상으로 처리할 수 있다.

추후 ARM SIMD 아키텍처에서 멀티코어를 효율적으로 활용할 수 있는 방법에 대한 연구가 이루어진다면, NEON과 함께 결합하여 더욱 고속화된 검출기를 구현할 수 있을 것으로 예상된다.

References

[1] Navneet Dalal, Bill Triggs, “Histogram of Oriented Gradients for Human Detection”, Proceedings of the IEEE, Computer Society Conference on Computer Vision and Pattern Recognition, vol.1, pp.886-893, 2005.

[2] Qiang Zhu, Shi Avidan, Mei-Chen Yeh, Kwang-Ting Cheng, “Fast Human Detection Using a Cascade of Histograms of Oriented Gradients”, Proceedings of the 2006 IEEE Computer Society

Conference on Computer Vision and Pattern Recognition, vol.2, pp.1491-1498, 2006.
[3] Christopher J.C Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", Journal, Data Mining and Knowledge Discovery, vol.2, pp.121-167, June, 1998.
[4] ARM, "RealView Compilation Tools v4.0 - NEON Compiler", http://infocenter.arm.com/. 2009.
[5] James Kennedy, Russell Eberhart, "Particle Swarm Optimization", Proceedings of the IEEE, Conference on Neural Networks, vol.4, pp.1942-1948, 1995.
[6] Renato A. Krohling, "Gaussian Swarm: A Novel Particle Swarm Optimization Algorithm", Proceedings of the IEEE, Conference on Cybernetics and Intelligent System, vol.1, pp.372-376, December, 2004.
[7] Sung-Tae An, Jeong-Jung Kim, Ju-Jang Lee, "SDAT: Simultaneous Detection and Tracking of Humans using Particle Swarm Optimization", Proceedings of the IEEE, International Conference on Mechatronics and Automation, pp.483-488, August, 2011.
[8] Guoqing Xu, Xiaocui Wu, "Real-time Pedestrian Detection Based on Edge Factor and Histogram of Oriented Gradient", Proceeding of the IEEE, International Conference on Information and Automation, pp.384-389, June, 2011.

BIOGRAPHY

Kwon Ki-Pyo (Student Member)

2012 : BS degree in Computer Engineering, Hanbat National University.
2012 ~ Present : MS Course in Computer Engineering, Hanbat National University.

Lee Jae-Heung (Member)

1983 : BS degree in Electronic Engineering, Hanyang University.
1985 : MS degree in Electronic Engineering, Hanyang University.
1994 : PhD degree in Electronic Engineering, Hanyang University.
1989 ~ Present : Professor in Dept. of Computer Engineering, Hanbat National University