ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 16-03-01

j.inst.Korean.electr.electron.eng.Vol.20,No.3,209~219,September 2016
http://dx.doi.org/10.7471/ikeee.2016.20.3.209

실시간 영상처리를 위한 SVM 분류기의 FPGA 구현

FPGA Design of SVM Classifier for Real Time Image Processing

나 원 섭*, 한 성 우*, 정 용 진*★

Won-Seob Na*, Sung-Woo Han*, Yong-Jin Jeong*★

Abstract

SVM is a machine learning method used for image processing. It is well known for its high classification performance. We have to perform multiple MAC operations in order to use SVM for image classification. However, if the resolution of the target image or the number of classification cases increases, the execution time of SVM also increases, which makes it difficult to be performed in real-time applications. In this paper, we propose an hardware architecture which enables real-time applications using SVM classification. We used parallel architecture to simultaneously calculate MAC operations, and also designed the system for several feature extractors for compatibility. RBF kernel was used for hardware implemenation, and the exponent calculation formular included in the kernel was modified to enable fixed point modelling. Experimental results for the system, when implemented in Xilinx ZC-706 evaluation board, show that it can process 60.46 fps for 1360x800 resolution at 100MHz clock frequency.

요 약

영상처리에 쓰이는 기계학습 방법 중 하나인 SVM은 일반화 능력이 뛰어나 객체를 분류하는 성능이 뛰어나다. SVM을 이용하여 객체를 분류하기 위해서는 여러 번의 MAC 연산을 반복해서 수행해야 한다. 하지만 영상의 해상도가 늘어남에 따라 분류를 해야 하는 개체가 늘어나게 되면 연산 시간이 증가하게 되어 실시간 처리를 요하는 고속 시스템에 사용하기 어렵다. 본 논문에서는 실시간 처리를 요하는 고속 시스템에서도 사용이 가능한 SVM 분류기 하드웨어 구조를 제안한다. 실시간 처리를 하는데 제한 요소가 되는 반복 연산은 병렬처리를 통하여 동시에 계산할 수 있게 하였고 다양한 종류의 특징점 추출기와도 호환이 가능하도록 설계하였다. 하드웨어 구현에 사용한 커널은 RBF 커널이며 커널 사용으로 생기는 지수 연산은 식을 변형하여 고정소수점 연산이 가능하도록 하였다. 제안한 하드웨어의 성능을 확인하기 위해 Xilinx ZC706 보드에 구현하였고 1360 x 800 해상도 이미지에 대한 수행 시간은 동작 주파수 100 MHz에서 약 60.46 fps로 실시간 처리가 가능함을 확인했다.

Key words : Machine Learning, SVM, Image Processing, Real Time Processing, Parallel Processing

* Dept. of Electronics and Communications Engineering, Kwangwoon University
★ Corresponding author
yjjeong@kw.ac.kr / 02-940-5551

※ Acknowledgment
This work was supported in part by the Research Grant of Kwangwoon University in 2015 and the IT R & D program of Ministry of Trade, Industry and Energy (10049192, Development of a Smart Automotive ADAS SW-Soc for a Self-Driving Car)
Manuscript received Aug. 23, 2016; revised Sep. 26, 2016 ; accepted Sep. 26, 2016

Ⅰ. 서론

기계학습이란 다양한 데이터를 통한 학습을 하여 컴퓨터가 스스로 최적의 판단이나 예측을 가능하게 해주는 기술이다. 기계학습은 마케팅, 금융권 등에서 사용되는 데이터마이닝 분야나 숫자인식, 동작인식, 지문인식 등을 수행하는 영상처리 분야 등 다양한 분야에서 사용하고 있다.[1] 그 중 SVM(Support Vector Machine)은 단순히 오류율을 줄이기 위한 학습을 하는 신경망(Neural Network)과는 달리 일반화 성능의 극대화를 꾀하는 방법으로 영상처리를 통한 패턴인식 분야에서 뛰어난 성능을 나타낸다.[2]

SVM 알고리즘은 최근 많은 연구가 진행 중인 운전자 보조 시스템(Advanced Driver Assistance System : ADAS)의 핵심 기술인 TSR(Traffic Sign Recognition), VD(Vehicle Detection), PD(Pedestrian Detection) 등에 적용하였을 때도 뛰어난 성능을 보인다. ADAS의 특성상 차량이 주행 중인 상태에서 사용하여야하기 때문에 실시간 처리를 필요로 하지만, SVM 분류(classification)를 수행하기 위해서는 여러 개의 객체 후보에 대해 MAC (Multiply-ACcumulate) 연산을 반복적으로 수행하기 때문에 영상의 해상도가 커질수록 실시간으로 처리하기 힘든 문제점이 있다.

SVM 분류를 실시간으로 처리하기 위하여 여러 가지 가속화 방법들이 제안되었다.[3]-[8] 첫 번째 방법으로는 SVM 분류 시 사용하는 RBF(Radial Basis Function) 커널(kernel)의 내부 MAC 연산 구조를 변형한 Hardware-Friendly 커널을 사용하여 가속화를 하였다.[4],[5] 하지만 해당 논문에서 사용한 커널은 RBF 커널에 비해 분류 성능이 낮은 단점이 있다. 두 번째 방법은 GPU(Graphic Processing Unit)를 이용하여 SVM 분류를 가속화 하는 것이다.[6],[8] 하지만 GPU의 경우 FPGA(Field-Programmable Gate Arrays)보다 최적화가 어렵고 전력 사용량이 높다는 문제가 있어 임베디드 시스템에서 사용하기에는 적합하지 않다. 마지막 방법은 FPGA를 이용하여 SVM 분류기 내부의 MAC 연산을 병렬적으로 처리하는 것이다.[3],[7]

본 논문에서는 [3], [7]와는 달리 실시간 처리를 하는데 제한 요소가 되는 반복 연산중 분류기 내부 MAC 연산을 병렬로 설계하는 것이 아닌 SVM 분류기 자체를 병렬로 설계한 하드웨어 가속기를 제안한다. 설계에 사용한 커널은 RBF 커널이며 커널 사용으로 생기는 지수 연산을 고정소수점 연산이 가능하도록 식을 변형하였다. 학습에는 오픈소스인 LibSVM을 사용하였고[9] 설계한 가속기의 검증에는 TSR 알고리즘을 이용하였다.

본 논문의 구성으로는 2장에서는 개략적인 SVM 알고리즘에 대한 이론적 배경 및 사용 커널에 대한 소개를 하고, 3장에서는 SVM 분류의 세부 하드웨어 모듈에 대한 분류 흐름을 설명하며, 4장에서는 MAC 연산 병렬화 및 분류기 병렬화 구조에 따른 성능 분석 및 TSR 알고리즘을 이용한 SVM 분류 검증을 진행한다.

Ⅱ. SVM 알고리즘

1. SVM 학습

SVM은 일반화 성능을 올리기 위하여 마진(margin)이라고 하는 여백을 최대화 하는 방법을 사용한다.[10],[11] 학습을 하기 위해서는 동일한 특징점 추출기를 이용하여 positive 영상(찾고자 하는 영상)과 negative 영상(찾고자 하지 않는 영상)의 특징점을 추출한 후 추출한 특징점을 이용하여 일련의 방식으로 학습을 진행한다. SVM에서 학습을 통해 결정되는 분류기준을 초평면(hyperplane)이라 하는데 그림 1은 SVM 학습 과정에서 마진이 최대가 되는 초평면을 찾아가는 과정을 도식화 한 것이다.[12]

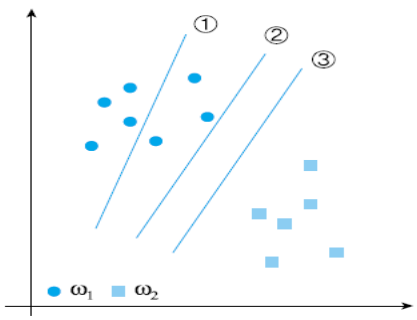

그림 1. SVM 학습과정
Fig. 1. SVM Learning Process

그림 1의 ①은 오류가 3개 있는 분류이다. 신경망은 초기 값 ①에서 시작하여 오류를 줄이며 오류가 없어지는 ②를 찾아간 후 학습을 종료한다.

하지만 SVM은 ②에서 멈추지 않고 ③을 찾아내어 학습을 종료하는데, 이 둘은 미지의 패턴을 분류하는 일반화 능력에서 상당한 차이를 보이게 된다. 즉, ③의 분류기가 테스트 패턴의 변형에도 더 강인한 분류기이고 이는 분류기의 성능이 더 뛰어나다는 것을 의미한다.

상술한 바와 같이 SVM을 이용한 학습은 마진을 최대로 하는 초평면을 찾는 과정을 말한다. 학습을 통하여 결정한 초평면에서 가장 가까운 샘플들을 서포트 벡터(Support Vector)라 하는데 다른 positive 서포트 벡터와 negative 서포트 벡터 사이의 거리를 마진(margin)이라 한다. 이러한 방식으로 학습된 SVM은 기본적으로 선형 이진 분류기의 특성을 가지게 된다.

하지만 일반적인 학습 데이터는 비선형적인 특성을 가지는 경우가 많은데 이를 해결하기 위해 SVM은 커널기법을 이용한다. 커널기법은 일정한 함수를 통해 평면상으로 보면 비선형적인 데이터를 더 높은 차원으로 매핑(mapping)하여 선형적인 형태가 나오도록 하는 방법을 말한다. 그림 2는 커널을 사용하였을 때 학습데이터가 고차원으로 매핑되는 것을 도식화 한 것이다.[12]

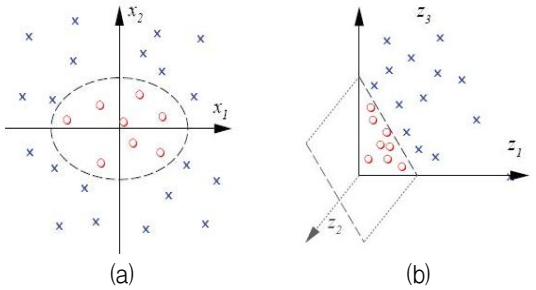

그림 2. 커널기법의 예
Fig. 2. Example of Kernel Method

그림 2-(a)는 커널기법을 사용하기 전 선형 분류가 되지 않는 상황이며 그림 2-(b)는 커널기법을 사용하여 선형분류가 가능한 차원으로 매핑이 된 것을 나타낸다. SVM 학습에는 다양한 커널들이 사용 가능한데 표 1은 각 SVM에서 주로 사용하는 커널 함수를 보여준다.

표 1. 커널 함수
Table 1. Kernel Functions

Kernel Function	Formula
Linear	$k(x,v) = x \cdot v$
Polynomial	$k(x,v) = (1+x \cdot v)^d$
RBF	$k(x,v) = \exp(-a\|x-v\|^2)$

세 커널 함수 중에서 가장 뛰어난 분류 성능을 나타내는 것은 RBF지만, Linear 및 Polynomial 커널에서는 벡터 내적을 하는 반면 RBF 커널의 경우 벡터 norm 연산을 하기 때문에 더 많은 계산량을 필요로 한다.[8] 본 논문에서는 다른 커널에 비해 분류 성능이 뛰어난 RBF 커널을 사용하였다.

SVM 학습을 위해서는 학습 데이터를 특징점 추출기를 이용하여 특징점을 추출하는데, 사용한 특징점 추출기 기술자(Descriptor) 값의 범위가 특징점 추출기 별로 다르기 때문에 입력한 데이터 값을 기준으로 하여 일정한 범위로 매핑한다. 그 방법은 아래 식 (1)과 같다.

$$x' = -1 + 2*(x - min)/(max - min) \quad (1)$$

여기서 x는 입력 값이고 x'은 매핑된 데이터 값이다. 그리고 min, max는 입력한 데이터 값들 중 최솟값과 최댓값을 의미한다. 만일 입력 값이 최솟값이면 -1로 최댓값이면 1로 매핑이 되어 모든 데이터 값은 -1부터 1사이로 매핑된다. 학습에 사용한 데이터의 최솟값, 최댓값을 이용하여 분류 시에도 같은 범위의 값으로 매핑한다.

2. SVM 분류

학습을 통해 만들어진 초평면을 이용하여 임의의 샘플이 입력되었을 때 그 샘플이 positive 영역에 속하는지 negative 영역에 속하는지 판단하는 것을 분류라 한다. 그림 3은 SVM을 이용한 분류를 도식화 한 것이다.

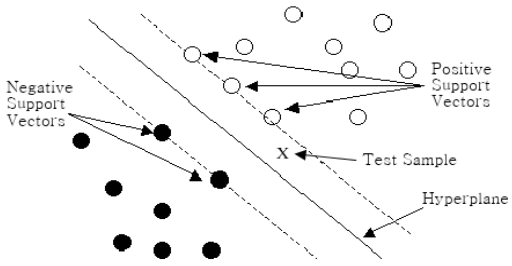

그림 3. 커널기법의 예
Fig. 3. Example of Kernel Method

SVM을 이용한 분류는 커널기법의 사용여부에 따라 방법이 바뀌게 된다. 커널기법을 사용하지 않은 선형 SVM의 경우는 학습을 통해 결정된 초평면의 법선 벡터를 미리 구해놓고 테스트 샘플

과의 벡터 내적(inner product, dot product)값과 바이어스 값의 합으로 분류를 할 수 있지만 커널 기법을 사용한 비선형 SVM의 경우 초평면의 법선 벡터를 미리 구해놓을 수 없고 커널함수 연산을 서포트 벡터의 수만큼 반복하여 수행하여야 한다.[12],[13] 즉, 비선형 SVM의 경우는 서포트 벡터의 수에 비례하여 분류시간이 증가하게 된다.

일반적으로 학습에 사용하는 데이터가 늘어나게 되면 서포트 벡터의 숫자가 늘어나게 된다. 서포트 벡터의 수가 늘어나면 분류 시간이 증가하게 되는데 이는 실시간 처리를 저해하는 요소가 될 수 있다. 또한 1장에서 서술한 것과 같이 영상의 해상도가 늘어남에 따라 분류해야 하는 개체가 늘어나 실시간 처리를 저해하는 경우도 나타나게 된다. 이를 통해 SVM 분류의 실시간 처리를 위해서는 다수의 서포트 벡터로 인해 생기는 반복 연산을 병렬처리를 통해 가속시키는 방법과 다수의 개체의 분류를 통해 생기는 반복 연산을 병렬처리를 통해 가속시키는 방법이 있다는 것을 확인할 수 있다. 두 가지 병렬처리에 대한 분석은 본 논문의 4장에서 서술한다.

III. 하드웨어

이번 장에서는 본 논문에서 설계한 하드웨어의 구조 및 수행 시간을 분석한다. 먼저 2장에서 설명한 SVM 분류를 실제적으로 수행하기 위한 알고리즘을 소개하고 이를 바탕으로 제안하는 하드웨어 구조를 보여준다. 그리고 수행 시간을 분석하여 설계한 하드웨어의 실시간 처리 가능성을 확인한다.

1. SVM 분류 알고리즘

그림 4에 있는 *SVM Classifier* 모듈은 RBF 커널을 사용한 SVM 분류 결과를 얻기 위해서 식 (2)를 풀어낸다.

$$D(x)=\sum_{i=1}^{SV}(coef*e^{-\gamma*\|\vec{x}-\vec{v_i}\|^2})-\rho \qquad (2)$$

SVM Classifier 모듈은 *Vector Calculation*, *Exponent Calculation*, *Taylor Series*, *Coefficient Calculation* 그리고 *D(x) Calculation* 총 5가지 파트로 나뉜다. 그림 4는 SVM 분류 흐름도이다.

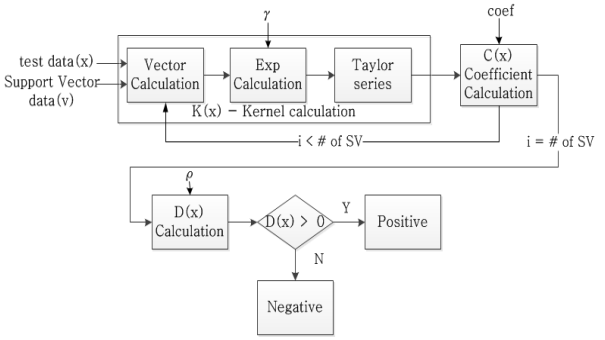

그림 4. SVM 분류 흐름도
Fig. 4. SVM Classification Flow

SVM Classifier 모듈의 각 파트별로 수행하는 연산의 내용은 다음과 같다.

a. Vector Calculation

Vector Calculation 파트는 식 (3)을 계산한다.

$$\|\vec{x}-\vec{v}\|=\|\vec{x}\|^2-2*\vec{x}\cdot\vec{v}+\|\vec{v}\|^2 \qquad (3)$$

여기서, $\vec{x}$는 테스트 벡터 값이고 $\vec{v}$는 서포트 벡터 값이다. *Vector Calculation* 파트는 벡터 차원에 따라 최대 350번 반복하여 계산한다.

b. Exponent Calculation

Exponent Calculation 파트는 식 (4)를 계산한다.

$$-\gamma*\|\vec{x}-\vec{v}\|^2 \qquad (4)$$

이때, γ는 학습 시 얻어진 데이터 값이다.

c. Taylor Series

식 (5)의 계산을 고정 소수점으로 계산하기 위해 식 (6)과 같이 테일러 급수로 변환하여 계산한다.

$$e^{-\gamma*\|\vec{x}-\vec{v}\|^2} \qquad (5)$$

$$e^x=1+x+\frac{x^2}{2} \qquad (6)$$

d. Coefficient Calculation

Coefficient Calculation 파트는 식 (7)을 계산한다.

$$coef*e^{-\gamma*\|\vec{x}-\vec{v}\|^2} \qquad (7)$$

coef는 학습시 얻어진 데이터 값이다. *Coefficient Calculation* 파트는 서포트 벡터의 수만큼 최대 500번 반복 계산한다.

e. D(x) Calculation

D(x) Calculation 파트는 전 과정까지 계산된 값을 이용하여 SVM 분류를 위한 식 (2)를 계산하고 최종 결과 값을 만든다. 이때, ρ는 학습 시 얻어진 데이터 값이다.

최종 결과 값(D(x))을 얻고나서, 그 값이 0보다 크거나 같으면 positive한 샘플이고 0보다 작으면 negative한 샘플로 판단한다. 그리고 positive한 샘플의 경우 flag를 1로하고 negative한 샘플의 경우 flag를 0으로 하여 Output Memory에 저장한다. 저장된 결과는 호스트 PC로 내보내어 질 때 32 bit버스에 8, 16, 24, 32번째 bit에 하나씩 입력하여 내보낸다.

2. 전체 하드웨어 구조

그림 5는 설계한 SVM 분류 가속기의 블록도이다. 영상의 해상도가 높아지면 입력 영상에서 분류를 하기위한 영역의 수가 늘어나 반복 연산이 늘어나고 전체 수행 시간이 오래 걸리게 되어 실시간 처리에 부적합한 결과가 나타난다. 따라서 본 논문에서는 분류를 할 영역인 마스크(mask) 100개를 동시에 계산함으로써 전체 수행 시간을 줄일 수 있었다. 동시에 계산하는 마스크의 개수를 늘리면 전체 수행 시간이 줄어드는 효과를 얻을 수 있으나 하드웨어 사용량이 늘어나게 되는 단점이 생긴다. 따라서 900개의 마스크를 처리하면서 60 fps (frame/sec) 이상의 성능을 낼 수 있는 최소 조건인 100개의 마스크를 동시에 계산할 수 있도록 설계하였다.

100개의 마스크를 병렬적으로 처리하기 위하여 100개의 SVM 분류기를 배치하고 각각의 분류기에 2개의 테스트 벡터 메모리(Test Vector Memory)를 더블 버퍼의 역할로 연결 하였다. 미리 학습된 서포트 벡터와 RBF 커널을 사용함으로써 학습 시 계산되어 나오는 γ, ρ, coef와 같은 파라미터를 저장하는 메모리(Support Vectors Memory, Parameter Memory)는 100개의 분류기가 같은 값을 공유한다.

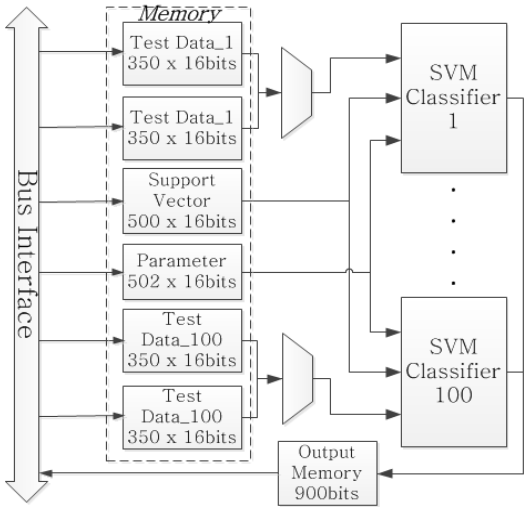

그림 5. SVM 분류 가속기 블록도

Fig.5. Block Diagram of SVM Classification Accelerator

3. Timing Analysis

그림 6은 설계한 하드웨어가 한 마스크를 수행하는데 소요되는 시간을 분석한 것이다. 그림 6-(a)는 하나의 서포트 벡터에 대한 계산과정을 보여준다. 특징점 추출기의 기술자를 통해 나오는 데이터는 하나의 영상에 대해 하나의 값이 나오는 것이 아니라 다차원의 벡터 형태로 값이 나타난다.

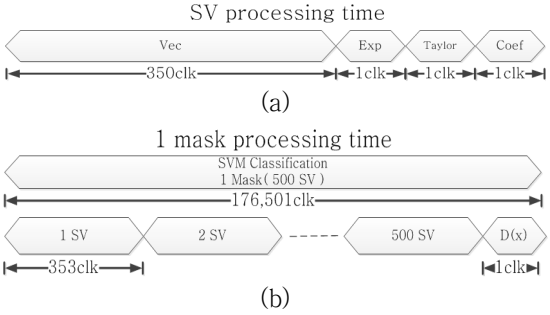

그림 6. 한 마스크에 대한 수행 시간 분석

Fig. 6. Excution Time Analysis for 1 mask

실제로 학습 시 사용한 특징점 추출기의 종류를 특정하지 않고 다양한 종류의 특징점 추출기와 호환이 가능하도록 서포트 벡터와 테스트 벡터의 최대 차원을 350으로 설계하였다. 또한 학습할 때 사용한 샘플의 수나 학습하는 샘플의 종류에 따라서 나타날 수 있는 서포트 벡터의 수도 정해져 있지 않기 때문에 최대 500개 까지 서포트 벡터를 사용할 수 있도록 설계하였다. 한 마스크에 대한 SVM 분류 결과를 얻기 위해서는 입력 데이터의 차원 수만큼 반복 계산하고 그 값들을 누적시

켜 서포트 벡터의 수만큼 반복 연산을 해야 한다. 이러한 내부 반복 연산으로 하나의 마스크에 대해 SVM 분류 결과를 얻기 위해서는 176,501clk이 소요된다. 한 마스크에 대한 수행 시간 분석은 그림 6-(b)를 통해 확인할 수 있다.

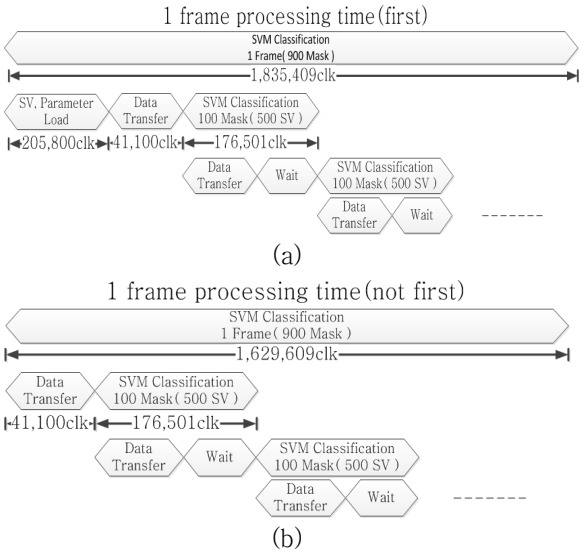

그림 7. 한 프레임에 대한 수행 시간 분석
Fig. 7. Excution Time Analysis for 1 frame

그림 7-(a)는 첫 번째 프레임을 수행하는데 대한 연산 시간을 분석한 것이다. 첫 번째 프레임을 수행하기 위해서는 학습을 통해 미리 가지고 있는 서포트 벡터 데이터와 파라미터 데이터를 불러와 저장한다. 그 후 분류에 사용할 테스트 벡터 데이터를 100개의 마스크 수만큼 불러와 첫 번째 버퍼에 저장한다. 저장이 끝나면 첫 번째 버퍼의 데이터를 이용하여 SVM 분류를 진행하고 동시에 두 번째 버퍼에 다음 100개의 마스크에 대한 테스트 벡터 데이터를 저장한다. 첫 번째 버퍼에 대한 SVM 분류가 끝나면 두 번째 버퍼에 저장된 데이터를 이용하여 SVM 분류를 진행하고 첫 번째 버퍼에 다음 100개의 마스크에 대한 테스트 벡터 데이터를 저장한다. 이와 같은 과정을 반복하여 총 900개의 마스크에 대해 SVM 분류를 진행하고 결과 값을 저장한다. 일련의 과정을 통한 첫 번째 프레임의 SVM 분류 결과를 얻기 위해서는 1,835,409clk이 소요된다.

그림 7-(b)는 첫 번째 프레임을 제외하고 나머지의 한 프레임을 수행하는데 대한 연산 시간을 분석한 것이다. 첫 번째 프레임에서 서포트 벡터 데이터와 파라미터 데이터를 저장해 놓았기 때문에 테스트 벡터 데이터만 순차적으로 불러와 연산을 처리하면 된다. 연산을 하는 과정은 첫 번째 프레임과 같고 한 프레임의 SVM 분류 결과를 얻기 위해서는 1,629,609clk이 소요된다.

Ⅳ. 실험 및 분석

1. 병렬구조 비교

그림 8은 SVM 분류기의 내부를 모듈화 하여 나타낸 것이다. Vector Unit(VU)은 벡터의 norm을 계산하는 모듈이고, Support Vector Unit(SVU)은 하나의 서포트 벡터와 테스트 벡터를 이용하여 커널함수를 연산하여 누적하는 모듈이다. 그리고 Classifier(CL)는 SVM 분류기 전체 모듈을 의미한다.

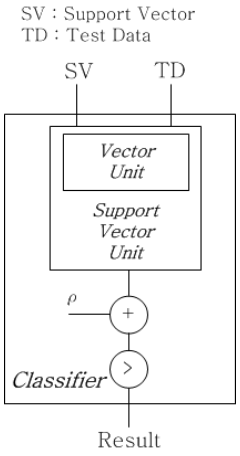

그림 8. SVM 분류기 내부
Fig. 8. Inside of the SVM Classifier

2장에서 서술한 것과 같이 SVM 분류 시 수행 시간을 증가시키는 요인은 서포트 벡터의 수와 분류해야 하는 개체의 수로 나누어진다. 즉, SVM 분류의 실시간 처리를 위해서는 SVU를 병렬처리하여 다수의 서포트 벡터를 동시에 연산하거나 CL을 병렬처리 하여 다수의 개체를 동시에 연산하여야 한다. SVU를 병렬처리를 한 [2], [6]에서는 동시에 여러 개의 서포트 벡터를 처리할 수 있도록 설계하였고 본 논문에서는 CL을 병렬처리하여 동시에 여러 개의 마스크를 처리하도록 하여 수행 시간을 단축시켰다.

그림 9는 SVU를 병렬적으로 구성한 SVM 분류기 이다. 본 논문에서 100개의 CL을 이용하였으므로 동일한 조건으로 100개의 SVU를 구성하

여 비교하였다. 동시에 100개의 서포트 벡터를 처리하기 위한 SVU 병렬구조의 메모리 구조는 CL 구조와는 반대로 100개의 서포트 벡터 메모리로부터 입력받고 하나의 후보군 벡터를 공유하여 계산한다. SVU까지 계산된 결과는 덧셈기를 통해 누적 계산하고 모든 서포트 벡터에 대해 계산이 완료 되면 ρ값과 뺀 결과로 positive인지 negative인지 판별하게 된다. SVU는 동시에 여러 개의 서포트 벡터를 계산하기 때문에 하나의 후보군을 빠르게 처리할 수 있다.

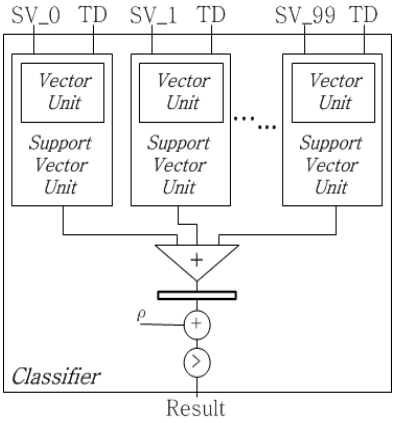

그림 9. SVU 병렬구조
Fig. 9. SVU parallel architecture

$$SVU_{dk} = \lceil N_{SV}/N_{SVU} \rceil * (350+3+1)+1 \quad (7)$$

$$Classifier_{dk} = \lceil N_{TD}/N_{Classifier} \rceil * ((350+3)*500+1)) \quad (8)$$

식 (7)은 SVU 병렬구조를 이용하여 하나의 후보군을 처리하는데 소요되는 클럭(clock)을 구하는 식이다. 100개의 SVU를 이용해 500개의 서포트 벡터를 가지는 하나의 후보군을 분류하는데 1,771clk, 900개의 모든 후보군을 처리하는데 1,593,900clk이 소요된다. 식 (8)을 이용하여 CL 병렬구조의 소요 클럭을 구하면 100개의 후보군을 분류하는데 176,501clk, 900개의 후보군을 처리하는데 1,588,509clk이다. 500개의 서포트 벡터를 가지는 900개의 후보군을 처리하는데 CL 병렬구조가 5,391clk만큼 소요 클럭이 적은 것을 확인할 수 있다.

그림 10은 SVU 병렬구조 방식과 CL 병렬구조 방식의 실행 시간을 서포트 벡터와 후보군의 수에 따라 비교한 것이다.

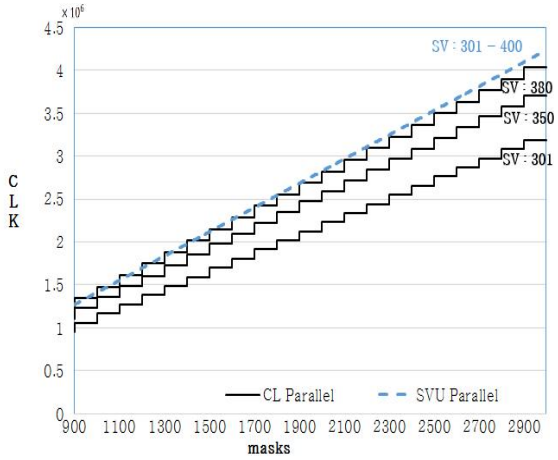

그림 10. 서포트 벡터 및 마스크의 개수에 따른 수행 시간 분석
Fig. 10. Execution speed analysis by the number of masks and Support Vectors

일반적으로 입력 영상의 해상도가 증가할 경우 후보군의 개수는 증가하기 때문에 서포트 벡터의 숫자와 더불어 후보군의 개수에 따른 처리 속도 분석도 함께 진행하였다. 병렬처리를 한 SVU와 CL의 개수를 100개로 가정하면 100단위의 서포트 벡터를 기준으로 비슷한 양상의 그래프가 나오게 되어 301 ~ 400개의 서포트 벡터에 대하여 분석을 진행하였다. 그림 10에서 점선은 SVU 병렬구조 방식의 수행 시간을 나타낸 것이며 실선은 CL 병렬구조 방식의 수행 시간을 나타낸 것이다. 그래프의 x축은 수행해야 하는 마스크의 수이며 y축은 소요 클럭이다.

병렬구조 방식은 SV의 수가 증가함에 따라 실행 시간이 선형적으로 증가하게 되는 반면에 CL 병렬구조 방식은 후보군 100개마다 계단식으로 증가함을 볼 수 있다. 이는 분류기가 100개의 후보군을 동시에 처리하기 때문에 나타나는 현상이다. 서포트 벡터의 수가 301에 가까울수록 CL 병렬구조 방식이 SVU 병렬구조 방식에 비해 소요 클럭이 더 적지만 서포트 벡터의 수가 400에 가까워 질수록 점점 CL 병렬구조 방식의 소요 클럭이 늘어나 SVU 병렬구조 방식에 비해 소요 클럭이 많은 구간이 있는 것을 확인 할 수 있다. 하지만 처리해야 하는 후보군이 늘어날수록 CL 병렬구조 방식이 SVU 병렬구조 방식보다 소요시간이 더 적은 것을 확인할 수 있다. 본 논문에서 구현한 CL 병렬구조 방식이 다른 논문들에서 사용한 SVU 병렬구조 방식보다 더 효과적인 가속방식임

을 확인할 수 있다.

2. 하드웨어 검증

개발한 가속기의 성능을 확인하기 위하여 TSR 알고리즘에 적용하였다. 그림 11은 Xilinx ZC706 보드에 구현한 실험 환경의 블록도이다.

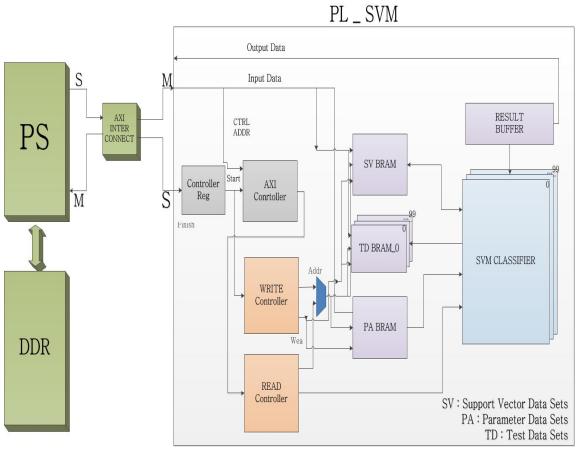

그림 11. 실험환경 블록도
Fig. 11. Block diagram for experiment

실험환경의 호스트 PC 및 소프트웨어 부분은 Zynq XC7Z045의 PS(Processing System)로 CPU는 Dual-Core ARM Cortex-A9, 1GB RAM, 운영체제는 Linaro 14.04, 컴파일러는 g++ 4.8.1이며 OpenCV 2.4.9 라이브러리를 사용하였다. 하드웨어를 구현은 Verilog HDL을 이용하였으며 Xilinx Vivado 2014.4.1.을 이용하여 합성하였다. FPGA는 Zynq XC7Z045의 PL(Programmable Logic)이고 동작 주파수는 100MHz이다. 사용한 영상의 해상도는 1360 x 800이며, 표지판 영역을 판단할 마스크의 크기는 40 x 40이고, 사용한 특징점 추출기는 HOG(Histogram of Oriented Gradient)이다. 학습 및 분류에 사용한 HOG의 크기는 32 x 32이다.

그림 12는 실험에 사용한 TSR 알고리즘의 순서도이다. TSR 알고리즘은 입력영상에서 표지판이 있을 법한 영역을 분할(segmentation)한 후 분할한 영역을 마스크를 이동하며 HOG 특징점 추출기를 이용하여 특징점을 뽑아낸다. 그리고 추출한 특징점을 학습 시 사용한 매핑 값을 기준으로 -1부터 1사이로 매핑하고 분류를 실시하고 SVM 분류로 나온 결과가 0 이상이면 positive로 판단하고 마스크 크기만큼 표지판 영역임을 표시한다. SVM 분류 후 마지막 영상이면 종료하고 아니면 처음부터 다시 수행한다.

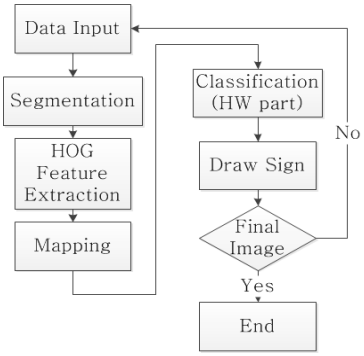

그림 12. TSR 알고리즘 순서도
Fig. 12. TSR algorithm flow

TSR 알고리즘 중에서 본 논문에서 제안하는 하드웨어 가속기를 사용하는 부분은 매핑 이후 분류를 하는 부분이다. 하드웨어 부분을 제외한 나머지는 ZC706 board의 PS에서 처리하여 그림 13과 같이 정상적으로 표지판을 표시하는 것을 확인하였다.

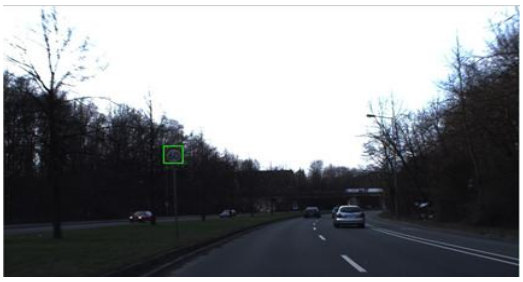

그림 13. TSR 결과영상
Fig. 13. TSR result image

표 2는 TSR 알고리즘 수행 시 검출률을 비교한 것이다. 독일 벤치마크 영상 900장을 이용하여 실험을 진행하였고 검출률을 구하는 방법은 식 (9)와 같다.

$$Detection\ rate = \frac{\#\ of\ detected\ signs}{\#\ of\ all\ signs} \quad (9)$$

표 2. TSR 수행결과 비교
Table 2. Comparison of TSR Result

	SW	HW
# of traffic signs	986	986
# of detected signs	943	943
Detection rate (%)	95.6	95.6

900장의 영상에 총 986개의 표지판이 있는데 그 중 943개를 검출하여 검출률은 95.6%이다. TSR 알고리즘을 이용한 실험 결과 소프트웨어만으로 수행하였을 때와 설계한 하드웨어 가속기를 이용하였을 때 같은 결과를 보였다. 이를 통해 본 논

문에서 제안하는 고정 소수점을 사용한 하드웨어의 성능이 부동 소수점을 사용하는 소프트웨어의 성능과 비슷하다는 것을 확인할 수 있다.

표 3. 소프트웨어, 하드웨어 수행 시간
Table 3. Execution time of SW and HW

	SW (ms)	SW + HW (ms)
Data Input	321.14	321.14
Segmentation	3378.35	3378.35
HOG	491.88	491.88
Mapping	45.76	45.76
Classification	**422000**	**21.34 (HW part)**
Output	18.11	18.11
Total	426255.24	4280.34

표 3은 실험환경에서 TSR 알고리즘을 소프트웨어만으로 수행했을 때와 본 논문에서 제안하는 하드웨어 가속기를 이용하여 수행했을 때의 수행 시간을 분석해 놓은 것이다. *Classification* 부분이 소프트웨어만을 이용했을 때는 422,000ms(422sec)가 소요되지만 하드웨어 가속기를 이용하면 21.34ms가 소요됨을 확인할 수 있다. 하지만 이 소요시간에는 하드웨어가 한 프레임 데이터를 처리하는 동안 소프트웨어는 다음 프레임 데이터를 준비하는 시간이 포함되어있다. 이는 실제 시스템의 소요시간에는 영향을 미치지 않는 시간이다. 이를 분석하기 위한 하드웨어 구간별 수행 시간은 표 3와 같다.

표 4. 하드웨어 구간별 수행 시간
Table 4. Execution time for each HW section

	Excution time for one frame(ms)
MMAP	4.8
Data Write	**0.4**
SVM Classification	**15.94**
Data Read	**0.2**
Total	21.34

표 4에서 MMAP은 소프트웨어에서 하드웨어로 보내기 위해 PS DDR 메모리에 맵핑하는 과정으로 하드웨어가 연산을 수행하는 동안 소프트웨어에서 미리 처리하여 소모되지 않는 시간이다. Data Write는 하드웨어에서 사용할 데이터를 FPGA의 BRAM(Block RAM)에 저장하는 부분이고 Data Read는 SVM 분류 결과를 소프트웨어가 읽어오는 시간이다. 따라서 실제적으로 하드웨어 수행 시간은 16.54ms(약 60.46 fps)이다. 이를 통해 제안하는 하드웨어가 실시간 처리가 가능함을 확인할 수 있다.

3. 타 논문과의 비교

본 논문에서 설계한 하드웨어와 동일한 구조를 가지는 논문을 찾을 수 없어 제안하는 하드웨어와 유사한 구조를 가지는 논문들과 성능을 비교하였다. 표 5는 본 논문과 타 논문과의 성능비교를 보여준다.

표 5. 타 논문과의 비교
Table 5. Comparison to other papers

	[3]	[8]	[14]	proposed
Device	Vertex5	GPU	Vertex6	Zynq
Operating frequency	50MHz	63MHz	40MHz	100MHz
Used feature point extractor	SIFT	HOG	LBP	any kind of descriptor
# of input dimensions	500	1980	1440	350
# of SVs	100	-	-	500
Parallel architecture (# of unit)	SVU (20)	SVU (-)	SVU (-)	CL (100)
Processing speed (# of masks)	4.44fps (900)	11fps (900)	60fps (19200)	60.46fps (900)
Used kernel	RBF	RBF	Linear	RBF

SVU를 병렬처리를 하여 SVM 분류를 가속시킨 [2]의 하드웨어는 사용하는 특징점 추출기를 SIFT으로 고정시키고 입력 벡터는 최대 500차원이며 서포트 벡터의 수는 최대 100이다. 속도향상을 위한 병렬구조는 20개의 SVU를 이용하였고 900개의 후보군을 처리하는데 225ms가 소요되어 실시간 처리가 제한된다. GPU를 이용하여 SVM 분류를 가속한 [7]은 900개의 마스크를 처리하는데 90ms가 걸려 실시간 처리에 한계가 있을 뿐 아니라 GPU의 특성상 임베디드 환경에서 전력 소모가 많은 한계점이 있다. 선형 SVM을 사용한 [13]은 LBP(Local Binary Pattern) 특징점을 이용한 머리-어깨 탐지 시스템으로 640 X 480 해상도에서 19200개의 후보군에 대해 60fps로 동작한다. 이는 선형 SVM의 특성상 분류속도가 빠르기 때문에 가능하지만 선형 SVM을 사용하기 때문에 커널 기법을 사용한 것에 비해 낮은 SVM 분류 성능을 가진다. 따라서 SVM의 성능을 보완하기

위해 추가적으로 전경검출 알고리즘까지 추가하는 단점이 있다. 이들에 비해 제안하는 하드웨어는 특징점 추출기를 고정하지 않고 다양한 특징점 추출기를 이용할 수 있고 입력 벡터는 최대 350차원, 서포트 벡터는 최대 500개이다.

본 논문에서는 처리 속도 향상을 위해 동시에 100개의 마스크를 계산하여 수행 시간을 줄이는 반면 [2]에서는 20개의 SVU를 이용하여 수행 시간을 줄이는 차이가 있다. 이로 인해 100개의 마스크를 처리하는 수행 시간은 [2]의 경우 25ms이지만 제안하는 하드웨어는 1.77ms이다. 또한 900개의 마스크에 대해 [2]의 구조는 실시간 처리가 불가능하지만 제안한 구조는 15.94ms로 실시간 처리가 가능하다. 이를 통해 본 논문에서 제안하는 하드웨어가 실시간 처리를 요하는 고속 시스템에 사용하기 더 적합함을 확인할 수 있다.

V. 결론

본 논문에서는 SVM을 이용한 영상처리 인식 시스템 중 실시간 처리를 요하는 고속 시스템에서 사용가능한 SVM 분류기의 설계 방법을 제안한다. 영상의 해상도가 높아짐에 따라 나타나는 반복 연산으로 인한 수행 시간의 저하를 여러 개의 분류기를 병렬구조로 하여 해결하였고 RBF 커널의 사용으로 인해 생기는 지수계산을 테일러 급수로 변환하여 고정 소수점 연산만으로 SVM 분류 연산을 수행하였다. 또한 내부 서포트 벡터 연산기 병렬구조와의 비교를 통해 제안하는 하드웨어의 구조가 고해상도 영상을 처리하는데 더 효과적임을 확인했다. 본 논문에서 제안하는 하드웨어의 성능을 검증하기 위해서 운전자 보조 시스템에서 사용하는 교통 표지판 인식 알고리즘에 적용하여 실험을 진행하였고 처리속도는 동작 주파수 100 MHz에서 60.46 fps로 실시간 처리가 가능함을 확인하였다. 하지만 서포트 벡터의 수가 늘어나 내부 반복 연산 횟수가 증가하거나 실험에 사용한 영상보다 고해상도인 Full HD 이상 영상의 사용 시 생기는 수행 시간의 증가를 해결하기는 어렵다. 이를 해결하기 위해 내부 반복 연산을 병렬처리를 통하여 전체 수행 시간을 감소시킬 필요성이 있다.

References

[1] K. Grauman and T. Darrell, “The pyramid match kernel Discriminative classification with sets of image features,” *Proceedings of IEEE International Conference on Computer Vision*, 2005

[2] Y. H. Kang, Y. B. Park, “Design of Automatic Document Classifier for IT documents based on SVM” *j.inst.Korean.electr.electron.eng,* Vol. 8, No. 2 186-194, 2004

[3] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, “FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification”, *Computational Imaging, IEEE Transactions*, Vol. 1, No. 1 56-70, 2015

[4] Marta Ruiz-Llata, Guillermo Guarnizo and Mar Yébenes-Calvino, “FPGA Implementation of a Support Vector Machine for Classification and Regression,” *IJCNN*, 2010

[5] Xipeng Pan, Huihua Yang, Lingqiao Li, Zhenbing Liu, Le Hou , "FPGA Implementation of SVM Decision Function Based on Hardware-friendly Kernel," *ICCIS*, 2013

[6] Bryan Catanzaro, ,“Fast Support Vector Machine Training and Classification on Graphics Processors,” *IMAGING*, Vol. 1, No. 1, March 2015

[7] Christos Kyrkou, Theocharis Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines," *IEEE TRANSACTIONS ON COMPUTERS*, Vol. 61, No. 6, 2012

[8] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, “A Practical Guide to Support Vector Classification,” Master thesis, Department of Computer Science National Taiwan University, 2003

[9] C. Chang and C. Lin, “LIBSVM : A Library for Support Vector Machine,” *ACM Transactions on Intelligent Systems and Technology*, Vol 2, No.3, 1-27, 2011

[10] B. E. Boser, I. Guyon, and V. Vapnik, “A training algorithm for optimal margin classiers,” *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144-152, 1992

[11] C. Cortes and V. Vapnik, “Support-Vector Network,” *Machine Learning*, Vol. 20, No. 3 273-297, 1995

[12] Il-Seok Oh, *Pattern Recognition*, Kyobomoongo, 2008

[13] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, 2008

[14] Tomasz Kryjak, Mateusz Komorkiewicz, Marek Gorgon, “FPGA IMPLEMENTATION OF REAL-TIME HEAD-SHOULDER DETECTION USING LOCAL BINARY PATTERNS, SVM AND FOREGROUND OBJECT DETECTION”, *Design and Architectures for Signal and Image Processing (DASIP) Conference*, Karlsruhe, Germany, 2012

BIOGRAPHY

Won-Seob Na (Student Member)

2014 : BS degree in Electronics and Communications Engineering, Kwangwoon University.
2014~ : Course of MS in Electronics and Communications Engineering, Kwangwoon University.

Sung-Woo Han (Student Member)

2016 : BS degree in Electronics and Communications Engineering, Kwangwoon University.
2016~ : Course of MS in Electronics and Communications Engineering, Kwangwoon University.

Yong-Jin Jeong (Member)

1983 : BS degree in Control and Instrumentation Engineering, Seoul National University.
1995 : MS, PhD degree in Electronics and Computer Engineering, University of Massachusetts, Amherst
1983~1989 : Research Engineer, ETRI
1995~1999 : Chief Researcher, Samsung Electronics
1999~current : Professor, Dept. of Electronics and Communications Engineering, Kwangwoon Univ.