논문 2012-49SP-2-4

영상 특징 추출을 위한 내장형 FAST 하드웨어 가속기

(An Embedded FAST Hardware Accelerator for Image Feature Detection)

김 택 규*

(Taek-Kyu Kim)

요 약

특징 추출 알고리즘은 영상 내에서 중요한 특징을 추출하기 위해 실시간 영상 처리 응용 분야에서 활용된다. 특히, 특징 추출 알고리즘은 추적 및 식별의 목적으로 다양한 영상처리 알고리즘에 특징 정보를 제공하기 위해서 활용되며, 주로 영상처리 전처리 단계에서 구현되고 있다. 광범위한 응용 분야에 이용되는 특징 추출 알고리즘의 처리 속도를 높인다면 혼합되어 사용될 다른 알고리즘 처리 소요 시간의 여유를 확보 할 수 있을 뿐만 아니라, 특징 추출 알고리즘이 적용된 영상 처리 응용 분야의 실시간 요건을 만족시키기 용이하기 때문에 중요하다. 본 논문에서는 특징 추출 기법을 고속으로 처리하기 위해 FPGA 기반의 하드웨어 가속기를 제안한다. 하드웨어 가속기 구현에 사용된 E. Rosten의 Feature from Accelerated Segment Test 알고리즘과 디지털 로직으로 구현한 하드웨어 가속기의 구조와 동작 절차에 대해 기술하였다. 설계한 하드웨어 가속기는 ModelSim을 이용해 동작 및 성능을 검증하였고, Xilinx Vertex Ⅳ FPGA 기반으로 로직을 합성해 구현 비용을 계산하였다. 제안한 하드웨어 가속기를 구현하기 위해 2,217개의 Flip Flop, 5,034개의 LUT, 2,833개의 Slice, 그리고 18개의 Block RAM을 사용하였으며, 640x480 크기의 영상으로부터 954개의 특징을 추출하는데 3.06 ms의 시간이 소요되어 기존의 결과보다 구현 비용 면에서의 우월함이 확인되었다.

Abstract

Various feature extraction algorithms are widely applied to real-time image processing applications for extracting significant features from images. Feature extraction algorithms are mostly combined with image processing algorithms mostly for image tracking and recognition. Feature extraction function is used to supply feature information to the other image processing algorithms and it is mainly implemented in a preprocessing stage. Nowadays, image processing applications are faced with embedded system implementation for a real-time processing. In order to satisfy this requirement, it is necessary to reduce execution time so as to improve the performance. Reducing the time for executing a feature extraction function dose not only extend the execution time for the other image processing algorithms, but it also helps satisfy a real-time requirement. This paper explains FAST (Feature from Accelerated Segment Test algorithm) of E. Rosten and presents FPGA-based embedded hardware accelerator architecture. The proposed acceleration scheme can be implemented by using approximately 2,217 Flip Flops, 5,034 LUTs, 2,833 Slices, and 18 Block RAMs in the Xilinx Vertex Ⅳ FPGA. In the Modelsim - based simulation result, the proposed hardware accelerator takes 3.06 ms to extract 954 features from a image with 640x480 pixels and this result shows the cost effectiveness of the propose scheme.

Keywords : Feature extraction, hardware accelerator, FPGA, featue from accelerated segment test.

Ⅰ. 서 론

비디오 영상을 이용한 정보 획득, 식별, 추적 등의 영상 처리 응용이 내장형 시스템에 널리 적용되어 감에 따라, 제한된 재원을 가지는 내장형 시스템의 특성을 고려한 실시간 영상 처리 기법 개발에 관심이 집중되고 있다. 영상 처리 응용 분야에서는, 활용 하고자 하는 목적에 따라 여러 영상 처리 알고리즘들이 조합되어 사용된다. 예를 들어, 영상 처리를 이용해 외부 영상의 분석

* 평생회원, 한국원자력연구원
(Korea Atomic Energy Research Institute)
접수일자: 2011년8월25일, 수정완료일: 2011년12월27일

및 추적을 수행한 응용 분야[9]에서는, 영상 내에서의 특징을 추출하는 알고리즘, 추적해야 할 특징을 선별하는 알고리즘, 그리고 선별된 특징들의 이동거리를 추출하는 알고리즘으로 구성되었다. 이렇게 여러 영상처리 알고리즘들이 조합되어 실시간으로 내장형 시스템에서 수행되기 위해서는, 알고리즘들이 효율적으로 처리되어 수행 시간이 단축되어야 한다. 영상 내 특징 추출 알고리즘은 추적뿐만 아니라 획득, 식별 등의 다양한 분야에서 영상 내 분석이 필요한 특징들의 정보를, 다른 알고리즘들에게 제공하는 역할로 사용된다. 광범위하게 응용되는 특징 추출 기법의 수행 시간을 단축하면, 영상 처리를 적용한 다양한 응용 분야에서 조합되는 다른 알고리즘들의 수행 시간 여유를 확보할 수 있기 때문에, 특징 추출 알고리즘의 수행시간을 단축시킬 필요성이 있다.

널리 이용되는 특징 추출 기법들로서는 SIFT[1], Harris[2], SUSAN[3] 등이 있으나, 적은 하드웨어 자원으로 구성된 내장형 시스템에서 실시간 처리되도록 구현하기에는 계산 수식이 복잡해 실시간 요건을 만족시키기 힘들거나 시스템 구현비용이 커진다[4,5]. 본 논문에서는 구현비용 및 실시간을 고려하여 수식이 비교적 단순한 FAST-n (Feature from Accelerated Segment Test)[6] 알고리즘을 하드웨어로 구현하였다. FAST-n 알고리즘은 기준 픽셀과 인접 픽셀들 간의 어둡고 밝음을 임계값을 이용해 판단하고, n개 이상의 연속되는 인접 픽셀들이 어두운 상태 혹은 밝은 상태를 유지하면 해당 기준 픽셀을 특징으로 판단한다. 처리 과정이 간단한 가감셈 연산과 비교 연산만으로 구성되기 때문에 순수 하드웨어 구현시, 적은 비용을 들여 로직 설계가 가능하여 하드웨어 가속기에 해당 알고리즘을 선정하고 구현하였다.

640x480 화소 크기의 초당 25 프레임을 출력하는 비디오 신호를 내장형 시스템이 입력받아 프레임 별 영상의 모든 픽셀들에 대해 특징 여부를 판단하는 과정이 요구되면, 영상 크기에 의해 메모리로 부터 읽고 처리해야할 계산 양이 많아진다. 많은 계산 양을 실시간 처리 요건에 따라 일반 목적 프로세서가 장착된 내장형 시스템에서 수행하면 하드웨어 구현 비용 증가뿐만 아니라, 고속 클럭 동작에 따른 하드웨어 발열의 효율적인 처리가 요구된다. 그러나 FPGA를 이용하면 비교적 적은 하드웨어 비용으로 실시간 처리가 가능하게 구현할 수 있고, 일반 목적 프로세서와 비교해 낮은 클록을 사용하기 때문에 하드웨어 발열과 전력 소모를 저하시킬 수 있다.

본 연구에서는 Rosten의 FAST-n 알고리즘의 n에 9를 적용(FAST-9)하여 실시간 처리가 가능한 하드웨어 가속기를 로직으로 FPGA에 구현하였다. Ⅱ장에서는 FAST-9 알고리즘을 3단계로 구분하여 설명하고, 설계된 하드웨어 가속기를 설명 세부적으로 설명한다. Ⅲ장에서는 구현된 하드웨어를 합성 툴과 시뮬레이션 툴을 사용해 구현 비용 및 성능 분석을 수행한다. Ⅳ장에서는 제안한 하드웨어 가속기에 대해 결론을 지었다.

Ⅱ. 본 론

1. FAST-9 알고리즘

일반적인 특징 추출 알고리즘은 입력 영상 내의 모든 픽셀들에 대해 특징 추출 검사를 하기 때문에, 추출된 특징 정보만을 사용하는 다른 알고리즘에 비해 수행 시간이 길다. 따라서 내장형 시스템에 적용할 특징 추출 알고리즘의 선정은 영상 처리를 수행하는데 소요되는 수행시간을 결정짓는 중요한 요소가 된다. 최근 자주 사용되는 알고리즘 중, Harris 및 DoG (Difference of Gaussian)를 적용한 SIFT 특징 추출 기법은 상관관계에 기반을 둔 적분연산을 수행해 수식을 하드웨어로 구현하였을 시의 구현 비용이 증가하고 처리 시간이 오래 걸리는 반면, FAST-9 기법은 구간 비교방식을 이용한 가감산 및 비교 연산을 수행하기 때문에 알고리즘을 하드웨어로 구현하였을 시에 적은 하드웨어 비용이 요구될 뿐만 아니라 신속 처리가 가능하다.

그림 1은 FAST-9 알고리즘을 설명하는 그림이다. FAST-9 기법은 3단계로 나누어 정의된다. 특정 픽셀

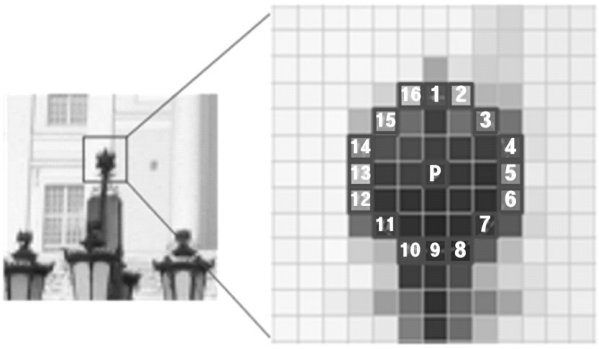

그림 1. FAST-9 알고리즘
Fig. 1. FAST-9 algorithm.

을 인접한 픽셀들과 임계값 t를 기준으로 비교해, 9개 이상의 인접 픽셀들이 연속으로 밝거나 어두울 경우를 찾아 특징 후보를 정의하는 Feature Detection (FD) 단계, 각 특징 후보들의 특징 정도에 따라 점수를 부여하는 Feature Score (FS) 단계, 그리고 인접한 특징 후보들 간의 점수를 비교해 높은 점수를 가진 특징을 찾는 Non-maximal Suppression (NMS) 단계로 구성된다.

$$S_{p\to x}=\begin{cases} S_{dark}, & I_{p\to x}\le I_p-t \\ S_s, & I_p-t<I_{p\to x}<I_p+t \\ S_{bright}, & I_p+t\le I_{p\to x}\end{cases} \quad (1)$$

FD 단계에서 사용되는 수식 (1)은 영상의 모든 픽셀에 적용되어 수행되어야 하기 때문에, 이의 처리 시간이 특징 추출 알고리즘의 전체 수행 시간 중 큰 비중을 차지한다. 선택된 기준 픽셀 I_p와 원형으로 기준 픽셀 주변에 위치하고 있는 16개의 인접 픽셀 $I_{p\to x}$ ($x\in\{1,2,...,15,16\}$)들이 기준 픽셀 보다 밝은지 어두운지를 판단하는 수치적 기준인, 임계값 t를 설정한다. 설정된 임계값과 기준 픽셀의 합과 차를 이용하여, 만약 기준 픽셀과 임계값 t의 차보다 인접 픽셀 값이 작으면 인접 픽셀을 S_{dark}로, 임계값과 기준 픽셀과의 합보다 인접 픽셀 값이 크면 인접 픽셀을 S_{bright}로 정의한다. 정의된 인접 픽셀들이 9개 이상 연속으로 S_{dark}나 또는 S_{bright}이면, 기준 픽셀을 특징 후보로 정의한다.

$$V=\max\left(\sum_{x\in S_{bright}}|I_{p\to x}-I_p|-t,\ \sum_{x\in S_{dark}}|I_p-I_{p\to x}|-t\right) \quad (2)$$

특징 후보로 정의된 기준 픽셀들은 FS 단계에서 수식 (2)를 사용해 특징 정도에 따라 점수를 부여한다. 이는 FD 단계에서 사용한 임계값 t가 사용자가 초기에 임의로 설정한 값이라, 추출된 특징들은 임의성을 지니게 되기 때문이다. 이러한 특징들의 임의성을 NMS 단계에서 보상하고자 미리 수행되는 수식으로, FD 단계에서 특징으로 추출된 기준 픽셀 I_p이 인접 픽셀 $I_{p\to x}$들에 대해 특징 상태를 유지할 수 있는 최대 임계값 t를 찾고, 찾아진 임계값 t를 특징 점수로 정의한다.

FAST-9 알고리즘은 FD 단계에서 사용자 임의로 선정된 임계값 t를 사용하였기 때문에 임계값 t에 의존적인 특징 후보들이 추출된다. 이렇게 추출된 특징들은 FAST-9 알고리즘이 특징들 간의 비교방식을 취하지 않기 때문에 인접 특징들 간의 계산결과가 상호 영향을 주지 않아, 복수의 특징들이 밀집되어 발견된다. 그러나 특징이라 부를 수 있는 픽셀은 인접영역을 대표해야 하므로, NMS 단계에서는 군집되어 발견된 특징들 간의 점수를 비교하여 가장 점수가 큰 특징 후보를 제외한 나머지 특징 후보들을 제거한다. 찾아진 대표 특징 후보는 특징으로 정의된다. 대표 특징을 찾는 과정에서 서로의 특징 점수를 비교해야 하기 때문에 많은 비교연산이 수반된다.

2. 하드웨어 구현

본 논문에서는 디지털 논리 회로를 이용해 FAST-9 하드웨어 가속기(그림 2)를 FPGA에 설계하였다. 구현된 FAST-9 하드웨어 가속기는 알고리즘을 수행하기 위해 크게 특징 후보를 추출하는 FD 단계, 특징 후보들의 점수를 생성하는 FS 단계, 그리고 인접한 특징 후보들 간의 점수를 비교해 대표 특징을 찾는 NMS 단계로 나누어 설계하였다. 설계된 각 모듈들은 처리 효율을 향상시키기 위해 파이프라인 구조를 채택하였다. 각 단계에서 필요로 하는 데이터들의 접근성을 높이기 위해 Block RAM (BRAM)을 사용하였다. 특히, 외부 메모리와 가속기 간의 접근성을 높이기 위해 사용된 BRAM은 Finite-state Machine (FSM)을 사용해 제어하였다. 하드웨어 가속기는 전체 영상에서 8줄 씩을 미리 저장할 수 있게 순환 구조의 Pixel Line을 두었다. Feature Information과 Feature Position 정보를 저장하고 부르기 위해 First In First Out (FIFO)를 이용하였다. FS 모듈의 결과물 중 특징 후보들의 점수는 정보의 양이 방대하기 때문에 외부 메모리에 640x480KB를 할당하고 저장하였다. 이렇게 외부 메모리에 저장된 특징 후보의 점수들은 4줄까지 저장 가능한 순환 구조의 Score Line을 두어, NMS 모듈이 저장된 특징 후보들의 점수를 외부 메모리로부터 가능한 지연 없이 읽어오도록 설

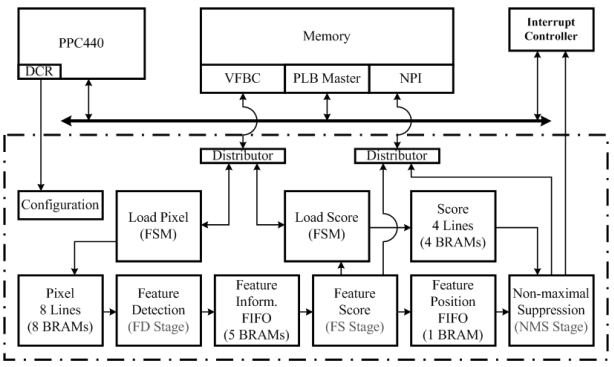

그림 2. 제안하는 FAST-9 하드웨어 가속기
Fig. 2. Proposed FAST-9 hardware accelerator.

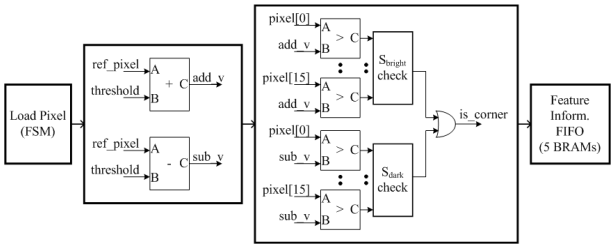

그림 3. Feature detection 모듈 구조
Fig. 3. Structure of feature detection module.

계하였다. 제안한 가속기의 설정 및 시작과 끝은 FPGA에 내장된 PPC440 프로세서를 이용해 소프트웨어로 처리하였다. 설계된 하드웨어는 아래 절차에 따라 수행된다. PPC440 프로세서는 비디오 영상이 외부 메모리에 저장되고 있는지를 확인하다가, 외부 메모리에 비디오 영상이 저장되면, 가속기 내부 Configuration 레지스터에 임계값, 각 단계의 가속기 모듈들이 사용할 외부 메모리 주소, 그리고 영상 크기에 대한 정보를 가속기가 동작하도록 설정한다. FD 모듈(그림 3)은 외부 메모리에 저장된 영상 내에서 특징을 찾기 위해 다음과 같이 수행한다.

1-1) 특징을 찾기 위해 기준 픽셀과 비교할 인접 픽셀 16개를 Pixel Line에서 읽어온다.

1-2) 다음으로 Configuration 레지스터에서 읽어와 임계값과 기준 픽셀과의 합(bmax)과 차(bmin)를 구해 16개의 인접 픽셀과 비교할 정보를 생성하고 레지스터에 저장한다.

1-3) 만약, 저장된 합과 인접 픽셀을 비교해 인접 픽셀이 크면 S_{bright}로 설정하고, 저장된 차와 인접 픽셀을 비교해 인접 픽셀이 작으면 S_{dark}로 설정한다.

1-4) 설정된 인접 픽셀들의 결과가 아홉 개 이상 연속으로 S_{bright}이거나 S_{dark}이면, 이를 특징 후보로 정의하고 FS 모듈로 전송하기위해 Feature Information FIFO에 저장한다.

FS 모듈(그림 4)은 Feature Information FIFO의 상태를 살펴보다가 데이터가 저장되면, 저장된 특징 후보 정보를 읽어 특징 후보의 점수를 산정한다. 특징 점수는 Feature Information FIFO에 저장되어있는 특징이 특징 상태를 유지할 수 있는 최대 임계값이다. FS 모듈의 특징 여부를 판단하는 연산은 FD 모듈과 동일하며, 특징 상태가 유지되는 최대 임계값을 찾는 로직이 추가되었다. FS 모듈은 처리는 다음과 같다.

2-1) Configuration 레지스터에 저장된 임계값을

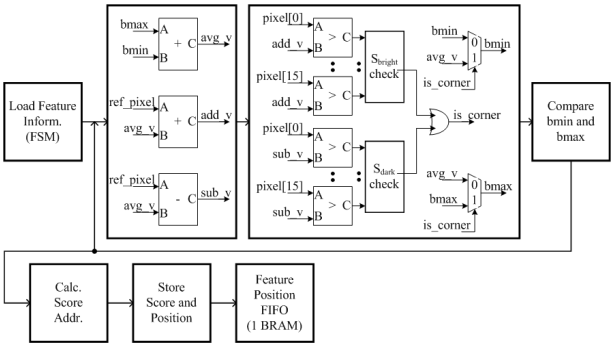

그림 4. Feature score 모듈 구조
Fig. 4. Structure of feature score module.

bmin에 저장하고 bmax에 255를 저장하고 평균값을 구해서 저장한다.

2-2) 저장한 평균값과 Feature Information에 저장되어있던 인접 픽셀들과 비교해 인접픽셀이 크면 S_{bright}, 작으면 S_{dark}로 설정한다. 설정된 S_{bright} 또는 S_{dark}가 9개 이상 연속이면 특징으로 판단한다.

2-3) 특징이면 bmin 값은 유지하고 bmax 값을 평균값으로 대체하고, 특징이 아니면 bmin 값을 평균값으로 대체하고 bmax 값은 유지한다.

2-4) 사용된 bmin, bmax가 같으면 bmin 값을 특징 후보의 점수로 정의해서 외부 메모리에 저장하고 외부 메모리 주소 정보를 Feature Position FIFO에 저장한다. 만약, 서로 다르면 저장된 bmin, bmax를 이용해 2-1)부터 다시 수행한다.

마지막으로 NMS 모듈(그림 5)은 외부 메모리에 저장되어있는 인접한 특징 후보들의 점수를 비교해, 가장 높은 점수를 가진 특징 후보를 특징으로 처리한다. 처리 절차는 다음과 같다.

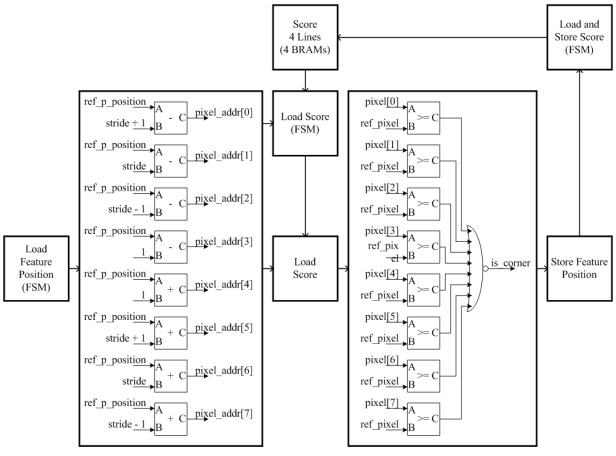

그림 5. Non-maximum suppression 모듈 구조
Fig. 5. Structure of non-maximum suppression.

3-1) Feature Position FIFO에 저장되어있는 기준 특징 후보 점수의 외부 메모리 주소를 읽어온다. 읽어온 주소 정보를 이용해 인접 특징 후보 점수들의 외부 메모리 주소를 계산한다.

3-2) 계산된 주소들로부터 특징 후보들의 점수를 읽어온 다음, 기준 특징 후보 점수와 비교한다.

3-3) 만약, 기준 특징 후보의 점수가 인접한 주변 점수들보다 크면 선택된 특징 후보를 특징으로 간주해 저장하지만, 작거나 같게 되면 특징에서 제외한다.

3-4) Feature Position FIFO에 저장되어있는 모든 특징 후보 정보에 대해서 상위 절차를 수행한다.

제안한 하드웨어 가속기는 입력 영상에 대해 FD 그리고 FS 단계를 거쳐 NMS 단계를 수행해 특징 추출을 마치게 되면, 인터럽트 제어기에 인터럽트 신호를 '1'로 설정함으로서 PPC440 프로세서에게 특징 추출이 완료되었음을 알려준다.

Ⅲ. 실 험

본 장에서는 제안한 특징 추출 가속기의 성능과 하드웨어 구현비용을 설명하고, 다른 논문에서 수행한 특징 추출 알고리즘들과 비교하였다.

1. 성능

본 논문에서 제안한 하드웨어 가속기의 성능을 검증하기 위해 ModelSim 툴을 사용하였다. 또한 성능 비교를 위해 다른 논문에서 제안한 Harris, SUSAN, 그리고 FAST 알고리즘을 적용한 플랫폼들과 표 1에서 비교하였다. 특징 추출 가속기는 640x480 크기의 영상을 입력받아, 영상 한 장당 총 307,200개의 픽셀들에 대해 특징 유무를 검사한다. 설계된 하드웨어 가속기는 외부 메모리에 영상이 적재되는 시점을 기준으로 640x480 영상 내에서 954개의 특징 추출을 마치고 인터럽트 컨트롤러에게 동작이 완료되었다는 신호를 보낼 때까지 3.06ms의 시간이 소요되었다.

640x480 크기의 영상이 초당 25개씩 제안된 하드웨어 가속기에 입력된다면, 실시간으로 영상을 처리하기 위해 하나의 영상 처리에 40ms의 시간이 소요된다. 이러한 환경에서 제안된 하드웨어 가속기는 대략 900개의 특징을 추출하는데 3.06ms 이내의 시간이 소요되기 때문에 다른 영상 처리 알고리즘을 수행하는데 36.94ms 시간이 확보 가능하다. 정량적 평가를 위해 클럭 사이클 당, 몇 개의 픽셀을 처리할 수 있는지를 계산하였다. 본 논문의 하드웨어 가속기는 (640 x 480)/(100 MHz x 3.06 ms) = 1.0039 pixel/clock cycle의 성능을 가진다. 본 논문의 하드웨어 가속기에 적용한 FAST-9 알고리즘을 2.6GHz의 Opteron 프로세서를 가지는 플랫폼에서 구현하면 약 500개의 특징을 추출하는데 1.33 ms가 소요되어[7], (640 x 480)/(2.6 GHz x 1.33 ms) = 0.064 pixel/clock cycle의 성능을 가진다. 약 450개 이상의 특징을 더 추출함에도 불구하고, 본 논문에서 제안한 하드웨어 가속기가 약 16배의 pixel/clock cycle 성능이 향상되었다.

표 1. 다른 플랫폼 상에서의 다른 특징 추출 모듈 성능 비교

Table 1. Performance Comparison of Different Feature Extraction Module on Different Platforms.

Algorithm (platform)	Frequency (MHz)	Image Resolution	Exec. time (ms)	pixel per clock cycle
Harris[5] (GPU)	1,350	640 x 480	2.6	0.088
SUSAN[10] (FPGA)	60	512 x 512	8.35	0.526
SUSAN[8] (FPGA)	66	640 x 480	16.67	0.279
SUSAN[4] (FPGA)	100	640 x 480	3.142	0.9652
FAST-9[7] (CPU)	2,600	768 x 288	1.33	0.064
FAST-9 (FPGA)	100	640 x 480	3.06	1.0039

본 논문에서 제안한 하드웨어 가속기를 FAST가 아닌 다른 알고리즘을 적용한 플랫폼들과 표 1에서 성능을 비교하였다. Harris 알고리즘은 GPU 상에서 구현하였을 시에 특징 추출은 2.6 ms[5]가 소요되었고, SUSAN 알고리즘을 적용한 플랫폼들에서는 각각 8.35 ms[10], 16.67 ms[8], 그리고 3.142 ms[4]가 소요되었다. 제안한 가속기보다 GPU에서 Harris 알고리즘을 적용한 플랫폼이 영상 내에서 특징을 추출하는데 적은 시간이 소요되었지만, 동작 클럭이 높기 때문에 제안한 가속기가 약 11배의 높은 pixel/clock cycle 성능을 가진다. 제안한 논문과 유사한 FPGA 환경에서 설계된 SUSAN[4]은 다른 플랫폼에 적용된 SUSAN[10], SUSAN[8]보다 소요 시간 및 pixel/clock cycle 측면 성능이 우수하다. 또

한 제안한 가속기는 결과가 SUSAN[4]하고 유사하지만, 처리 시간이 약 2% 단축되고 pixel/clock cycle이 4% 향상되었다.

2. 구현 비용

제안한 하드웨어 가속기의 구현 비용을 계산하기 위해 자일링스 Vertex Ⅳ FPGA 기반 ISE (Integrated Software Environment) 개발 툴을 이용해 합성하고, 구현 비용을 확인하였다. 제안한 하드웨어 가속기의 구현 비용을 비교 평가하기 위해 성능이 유사하게 나타난 SUSAN[4]을 이용하였다. 표 2는 제안한 가속기와 SUSAN의 구현 비용이다.

제안한 가속기의 구현 비용은 FAST-9 알고리즘에 해당하는 FD, FS, NMS 로직, 가속기 설정을 위한 Configuration 로직, 그리고 외부 메모리로부터 데이터를 읽고 쓰기위해 Native Port Interface (NPI) 및 Video Frame Buffer Controller (VFBC)를 제어하는 로직을 포함한다. 설계된 하드웨어는 100MHz의 동작 클록을 가지며 2,217개의 Flip Flops, 5,034개의 LUTs, 2,833개의 Slices, 그리고 18개의 BRAMs이 사용되었다. 비교 대상으로 사용된 SUSAN 가속기는 외부 메모리와 데이터를 주고받기 위해 사용한 PLB bus IP interface (IPIF)를 제외한 하드웨어 비용이 고려되었으며, 총 2,281개의 Flip Flops, 8,450개의 LUTs, 4,601개의 Slices, 16개의 BRAMs, 그리고 1개의 MULT18X18가 사용되었다. 비록, 하드웨어 구현 비용 평가로부터 제안된 하드웨어 가속기는 SUSAN보다 2개의 BRAMs을 추가 사용하였지만, 외부 인터페이스 모듈의 구현 비용까지 고려했음에도 불구하고 2.89%의 Flip Flops, 67.86%의 LUTs, 62.41%의 Slices, 그리고 1개의 MULT18X18 재원의 사용을 감소시켜 구현을 간소화하였다.

표 2. 하드웨어 구현 비용 비교
Table 2. Comparison of Hardware Implementation Cost.

Resource	SUSAN[4]	Prop. Hardware Accelerator
Slice Flip Flops	2,281	2,217
4 input LUTs	8,450	5,034
Occupied Slices	4,601	2,833
Block RAMs	16	18
MULT18X18s	1	0

Ⅳ. 결 론

본 논문에서는 FAST-9 알고리즘을 이용한 특징 추출을 가속화하기 위한 하드웨어 가속기에 대해 설명하였다. 제안한 가속기는 3단계로 나누어져 각각의 단계가 파이프라인 구조를 가지도록 설계되었으며, 영상 크기 및 임계값 변경이 소프트웨어적으로 용이하도록 설계에 반영되었다. 시험을 통해 640x480 화소를 가지는 영상에서 약 950개의 특징을 추출하는데 3.06 ms가 소요되어 1.0039 pixel/clock cycle의 성능을 가짐을 확인하였다. 25Hz의 비디오 영상을 받아 특징을 추출하였을 경우 프레임 별 40 ms의 시간을 가질 때, 기타 알고리즘 처리를 위해 매 프레임마다 36.94 ms의 여유시간이 확보된다. 또한 특징 추출 알고리즘만을 수행할 시에는 초당 약 326개 영상의 특징이 추출 가능하다. 본 논문에서 비교 평가로 사용한 다양한 플랫폼에 적용된 특징 추출 모듈들과 비교해 적은 하드웨어 비용으로 높은 성능을 가짐을 확인하였다. 그 중, SUSAN을 하드웨어 가속기로 구현한 가속기와 비교해 2.89%의 Flip Flops, 67.86%의 LUTs, 62.41%의 Slices, 그리고 1개의 MULT18X18 로직을 감소시켰다. 제안한 하드웨어 가속기는 특징 추출이 필요한 다양한 영상 응용 분야에 적용될 것으로 기대된다.

참 고 문 헌

[1] D. G. Lowe, “Distinctive Image Features from Scales-invariant Keypoints,” *International Journal of Computer Vision*, vol 60, no. 2, pp. 91-110, 2004.

[2] C. Harris, and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings of the Fourth Alvey Vision Conference*, pp. 147-151, 1988.

[3] S. Smith, and J. Brady, “SUSAN-A New Approach to Low-level Image Processing,” *International Journal of Computer Vision*, vol. 23, pp. 45-48, 1997.

[4] C. Claus et al, “Optimizing the SUSAN Corner Detection Algorithm for a High Speed FPGA Implementation,” *International Conference on Field Programmable Logic and Application*, pp. 138-145, 2009.

[5] L. Teixeria, W. Celes, and M. Gattass, “Accelerated Corner-detector Algorithms,” in

BMVC08, 2008. [Online]. Available: http://www.comp.leeds.ac.uk/bmvc2008/proceedings/paper/45.pdf

[6] E. Rosten et al, "Faster and better: A Machine Learning Approach to Corner Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105-119, 2010.

[7] E. Rosten, T. Drummond, "Machine Learning for High-speed Corner Detection," *European Conference on Computer Vision*, vol. 1, pp. 430-443, 2006.

[8] M. Arias-Estrada, E. Rodríguez-Palacios, "An FPGA Co-processor for Real-Time Visual Tracking," *in Proceedings of International Conference on Field-Programmable Logic and Application*, pp. 710-719, 2002.

[9] C. Claus, W. Stechele, A. Herkersdorf, "Autovision - a Runtime Reconfigurable MPSoC architecture for Future Driver Assistance Systems," *it - Information Technology*, vol. 49, no. 3, pp. 181-187, 2007.

[10] C. Torres-Huitzil, M. Arias-Estrada, "An FPGA Architecture for High Speed Edge and Corner Detection," *IEEE International Workshop on Computer Architectures for Machine Perception*, pp. 112-116, Washington, DC, USA, 2000.

[11] E. Rosten, T. Drummond, "Fusing Points and Lines for High Performance Tracking," *IEEE International Conference on Computer Vision*, vol. 2. Springer 1508-1515, Beijing, China, 2005.

[12] B. Yu, et al. "A Corner Detection Algorithm Based on the Difference of FCC," *International Conference on Computer Design and Application*, vol. 4, pp. 226-229, Qinhuangdao, China, 2010.

[13] W. Wang, R. D. Dony. "Evaluation of Image Corner Detectors for Hardware Implementation," *Canadian Conference on Electrical and Computer Engineering*, vol. 3, pp. 1285-1288, 2004.

[14] F. Mokhtarian, R. Suomela, "Robust Image Corner Detection Through Curvature Scale Space," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1376-1381. 1998.

[15] L.-H. Zou et al, "The Comparison of Two Typical Corner Detection Algorithms," *Second International Symposium on Intelligent Information Technology Application*, pp. 211-215, Shanghai, China, 2008.

[16] 김택규, 박기용, 김영기, "영상 특징 추출을 위한 내장형 가속기 구현," *대한전자공학회 하계종합학술대회 논문집*, 제 34권, 1호, 698-701쪽, 2011년 6월.

[17] 김택규, 박기용, 김영기, "하드웨어기반 실시간 특징추출을 위한 로봇 비전 가속기 설계 및 구현," *제6회 한국로봇종합학술대회 논문집*, 197-200쪽, 2011년 6월.

저 자 소 개

김 택 규(평생회원)

2005년 고려대학교 전자 및 정보 공학부 졸업.

2008년 고려대학교 전자정보 공학과 석사 졸업.

2010년～현재 한국원자력연구원 연구원

<주관심분야 : 마이크로 프로세서, 임베디드 시스템, 영상처리>