멀티코어에서 피부색상 정보와 병렬처리 방법을 이용한 얼굴 검출

김홍희*, 이재흥*
*한밭대학교 컴퓨터공학과
e-mail:freemoora@naver.com

Face Detection using Skin Color Information and Parallel Processing Method on Multi-Core

Hong-Hee Kim*, Jae-Heung Lee*
*Dept of Computer Engineering, Hanbat University

요 약

최근 얼굴검출에 관한 연구는 FPGA를 통한 H/W설계부터 DSP, GPU, ARM Core에 효율적인 S/W 설계까지 다양하게 연구되고 있다. 본 연구에서는 Multi-Core에 효과적인 얼굴검출 방법을 제안한다. 피부색을 통한 얼굴 후보를 추출하고 그 외의 배경 이미지는 삭제하여 연산처리를 빠르게 하였다. Viola-Jones가 제안한 얼굴검출 알고리즘을 POSIX Thread를 사용하여 병렬 처리하였고 그 성능을 단일 코어와 멀티코어에서 측정하였다. 단일 코어에서는 성능의 향상이 없었으나 멀티코어에서는 약 1.8배 속도가 향상되었고 검출 성공률은 기존과 동일하였다.

1. 서론

최근 각종범죄가 사건사고 뉴스에 끊이지 않고 그에 따른 영상 저장 장치에 대한 소유가 늘어나고 있다. 영상 저장 장치는 CCTV에서 DVR, Network Camera등으로 변천해가고 있으며 카메라 또한 지능화되고 있다. 현재 생활 곳곳에 볼 수 있는 번호판 인식 카메라, 지문 인식 카메라 등과 같은 고성능 인식 알고리즘이 탑재된 시스템의 경우 PC를 활용하여 처리하는 경우가 많아 가격이 비싸고 설치범위가 제한적이다. 이와 더불어 최근 ARM 프로세서를 기반으로 임베디드 모듈의 성능이 향상되고 다양해짐에 따라 임베디드 기반의 영상처리 알고리즘에 대한 연구가 활발히 진행되고 있다. FPGA(Field-Programmable Gate Array)를 기반으로 얼굴검출 전용 알고리즘 하드웨어 설계가 가장 대표적인 방법이다[1].

최근 많은 모바일 기기에는 Single-Core 기반에서 Multi-Core로 발전해가고 있다. Single-Core에서 클록을 높이는 것보다 Multi-Core로 발전함이 다양한 확장성과 시스템 성능향상, 전력 소비의 감소 등의 효과를 얻을 수 있다[4]. Multi-Core로 H/W가 발전해감에 따라 S/W 또한 그에 맞게 프로그램이 설계되어야한다. Multi-Core가 내장된 칩에서 순차적인 프로그래밍은 효율적인 방법이 되지 못하며 컴파일러 또는 하드웨어의 컨트롤러를 통한 병렬 처리는 한계가 있다.

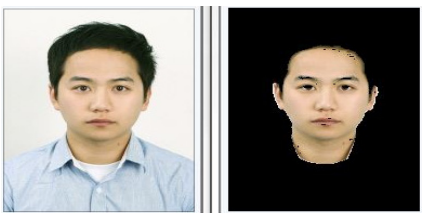
※ 본 연구는 교육과학기술부와 한국연구재단의 지역 혁신인력양성사업으로 수행된 연구결과임

본 연구에서는 다양한 얼굴검출 방법 중 Multi-Core에 효과적인 얼굴검출 방법을 연구하였다. 컬러영상 데이터를 피부색 기반으로 얼굴 후보를 추출하고 Viola-Jones이 제안한 얼굴검출 알고리즘[2]을 멀티프로그래밍으로 설계하였다. POSIX Thread를 사용하여 Viola&Jones 알고리즘의 독립적인 부분을 병렬 처리하고 Multi-Core가 내장된 SoC에서 그 성능을 측정하였다.

2. 기존 관련연구

2.1 기존의 얼굴검출 기술

기존의 얼굴검출 방법은 크게 지식 기반 방법(Knowledge-Based Top-Down Methods), 특징 기반 방법(Bottom-Up Feature-Based Methods), 형판 정합 방법(Template Matching), 외형 기반 방법(Appearance-Based Methods)으로 나눌 수 있다[4]. 지식 기반 방법은 사람의 이목구비(耳目口鼻)와 같이 얼굴의 형태를 구성하는 특성들을 기하학적인 방법으로 검출하는 방법이다. 특징 기반의 방법의 가장 대표적인 방법은 피부색을 이용하는 방법이다.



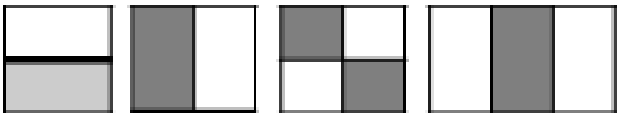
(그림 1) 피부색을 이용한 얼굴검출

RGB, YCbCr[3], HSI 등과 같은 영상 데이터 포맷을 기반으로 픽셀 데이터 값의 일정범위를 정하여 탐색을 하게 된다. 피부색을 이용하여 얼굴을 검출하는 방법은 고전적인 방법으로 검출 속도가 빠르지만 조명과 주변 환경, 피부색 등으로 인해 오류가 발생할 가능성이 크다.

형판 정합 방법은 미리 정해진 형판을 입력 영상과 비교하여 검출하는 방법이다. 고정된 영상에서는 성공률이 높지만 입력 형태가 달라지면 성공률이 낮아진다. 마지막으로 외형 기반 방법은 신경망, Support Vector Machine, HMM 등과 같이 많은 영상을 입력받아 학습한 후 학습된 데이터와 입력 영상을 비교하여 검출하는 알고리즘이다. 학습된 데이터의 양과 질에 따라 인식률이 달라지고 계산량이 많아 속도가 느리지만 다른 방법들이 비해 신뢰성이 높고 검출 성공률 또한 높다.

2.2 Viola-Jones 알고리즘

Viola와 Jones가 제안한 얼굴검출 알고리즘[3]은 현재 널리 사용되는 얼굴검출 알고리즘 중 하나이다. Viola-Jones 알고리즘은 Haar-like feature를 적분 이미지를 통하여 계산하는 알고리즘이다. 충분히 학습된 데이터가 있으면 성공률이 높으며 또한 픽셀을 직접적으로 연산하지 않고 특정 영역에 대한 정보만을 계산하므로 속도가 빠르다는 장점이 있다.

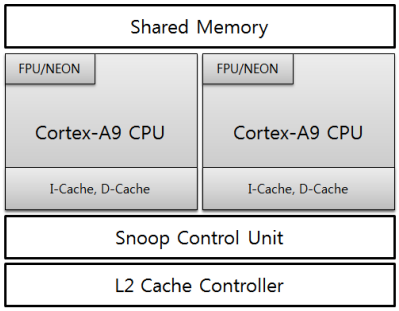


(그림 2) Haar-like feature

Haar-like feature를 이용하여 하나의 약분류기를 생성하고 분류기를 모아 하나의 Stage를 구성하여 강건한 AdaBoost 알고리즘[2]이 된다. 영상 데이터 모든 영역에 차례대로 Windows를 설정하고 Stage를 통과시켜 얼굴을 검출한다. 이를 통해 신뢰성 높은 검출기를 구성하게 되고 각각의 Windows는 독립적이다.

2.3 Cortex-A9 MPCore

Cortex-A9 MPCore 멀티코어 프로세서는 효율적인 전력 관리 기술과 높은 성능을 지원하며, 확장성이 뛰어나 현재 모바일 기기에서 많이 사용되는 최신 프로세서이다.



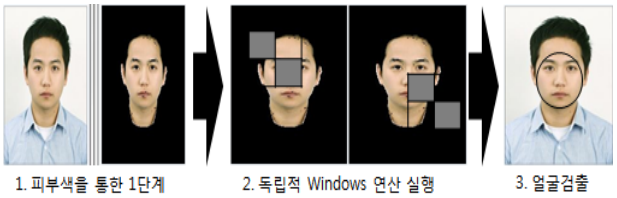
(그림 3) Cortex-A9 Dual-Core

Cortex-A9 MPCore는 1-4개의 프로세서를 지원하며 각각의 프로세서는 1GHz 클록 주파수 이상이 가능하다. 멀티코어와 SCU(Snoop Control Unit)를 통해 멀티코어 애플리케이션 프로그래밍이 효과적이다. 또한 NEON 미디어 처리 엔진을 통해 애플리케이션 성능을 개선하고 소프트웨어 개발의 편리성을 제공한다. 얼굴검출 알고리즘을 설계하다보면 부동소수점 연산은 필연적으로 사용할 수밖에 없다. Cortex-A9는 이전 ARM 프로세서의 FPU(Floating Point Unit)보다 성능이 두 배 개선되었다[5].

3. 제안하는 얼굴검출 방법

3.1 얼굴검출 멀티프로그래밍

멀티코어를 가진 Cortex-A9 MPCore는 기존의 단일코어 방식과 동일하게 설계하면 효율적이지 못하다. 컴파일러 또는 SCU를 통한 CPU의 업무 분담은 한계가 있으며 그 성능 또한 미비하다. 본 연구에서는 멀티코어에 효과적인 방법으로 다음의 2가지 단계를 제안한다.



(그림 4) 얼굴검출 멀티프로그래밍 흐름

3.1.1 피부색을 통한 얼굴 후보검출

피부색을 이용하여 얼굴을 검출하는 방법은 고전적인 방법으로 얼굴과 비슷한 색이 배경 영상에 있으면 심각한 오류를 범하게 된다. 하지만 얼굴 후보를 추출함에는 효과적이고 계산과정이 단순하여 ARM 프로세서에 적합하다. 컬러 영상 포맷 중 YCbCr의 CbCr 영역만으로 후보 얼굴 영역을 검출할 수 있다[3].

$$O_1(x,y)=\begin{cases}1 & \text{if } [100 \le Cb(x,y) \le 128] \\ & \text{and} [135 \le Cr(x,y) \le 164] \\ 0 & Otherwise\end{cases} \quad (1)$$

식(1)에 만족하는 Y영역의 범위만을 따로 저장하고 그 외의 범위에는 '0x00'으로 값을 채운다. 얼굴 후보 이외지역에 '0x00'를 채우게 되면 전체적인 연산이 빨라지고 강건하지 못한 분류기가 동작할 때 확실한 결과 값을 전달하여 최대 65%까지 시스템 속도가 빨라진다.

3.1.2 Windows 병렬 처리

피부색을 통한 1단계가 거치면 2단계는 POSIX Thread를 활용하여 Windows연산을 독립적으로 병렬 처리한다. 입력 영상에서 얼굴이 크기와 위치는 항상 다르기 때문에 Windows를 설정하여 모든 영상 범위를 차례대로 탐색해야한다.

Viola-Jones의 전체 알고리즘 순서는 2.2절에서 언급한 것과 마찬가지로 다양한 Haar-like feature가 모여서 Adaboost 알고리즘을 구성한다. Adaboost 알고리즘을 통해 높은 검출 성공률을 얻기 위해서는 매우 많은 수의 약분류기를 사용해야 한다. 많은 약분류기를 사용하게 되면 속도가 문제되기 때문에 Stage를 직렬로 배치하고 단계별로 모두 만족해야 얼굴을 검출하도록 되어있다[2]. 이러한 과정은 서로 의존성이 강하기 때문에 병렬 처리를 할 수가 없다. 또한 약분류기를 병렬처리하기에는 너무 많은 Thread를 생성해야 하므로 시간을 더욱 증가시킨다. 얼굴 탐색과정은 픽셀 처음부터 끝까지 모든 영역에 거쳐 이루어져야한다. 이러한 동작은 서로 독립적이고 의존성이 없다. 이러한 사실에 기반을 두어 탐색의 범위를 지정하는 Windows를 병렬 처리하도록 설계하였다.

(그림 5) 다양한 Windows 위치

(그림 5)에서의 빨간색 테두리가 Windows이다. Windows의 위치는 데이터의 크기에 따라 개수가 달라지며 상단 좌측부터 하단 우측까지 움직이게 된다. 본 연구에서는 코어 수에 맞게 POSIX Thread를 생성하여 Windows 탐색과정을 병렬처리 하였다.

```
for all WZ      // WZ: window size
 Resize image;
 Integral image;
 for all WP     // WP: window position
  Detect {
   for all SC  // SC: strong classifier
    for all WC // WC: weak classifier
     If failed, label the position as negative; jump to the next WP;
   If passed all the SCs, label the position as positive;
  }
```

(그림 6) Viola-Jones Pseudo Code

```
Make Thread[CORE]
for all/core WZ      // WZ: window size
   Thread[0] = Viola-Jones Face Detection; // Detecting face
   Thread[1] = Viola-Jones Face Detection; // Detecting face
  }
```

(그림 7) 제안하는 Viola-Jones Pseudo Code

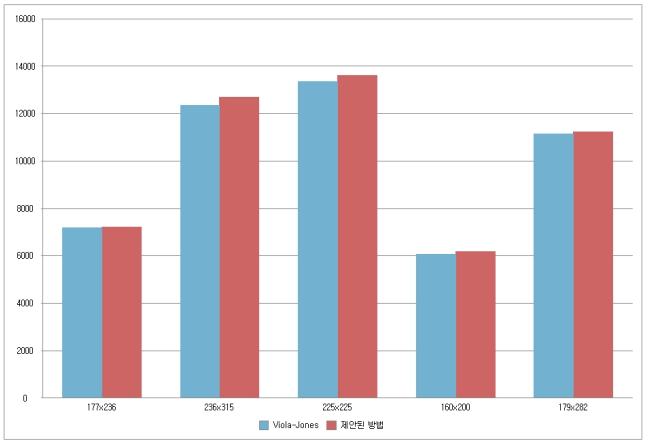
(그림 6)은 Viola-Jones 알고리즘의 Pseudo Code[7]로 모든 Windows 사이즈를 탐색하지만, 제안하는 방법은 Multi-Core의 개수만큼 병렬처리를 하게 때문에 반복 횟수가 줄어들어 얼굴검출 성능을 개선할 수 있다..

4. 실험

본 연구에서는 멀티코어에 효율적인 얼굴검출 알고리즘을 구현한 후 싱글코어와 멀티코어에서 성능을 측정하여 비교하였다. 얼굴검출에 필요한 학습데이터는 OpenCV Library에서 제공하는 "haarcascade_frontalface_alt2.xml"를 사용하였고 입력영상은 얼굴 정면이 보이는 이미지 파일을 사용하였다. 실험에 사용되는 SoC는 Cortex-A8이 내장된 S5PV210, Cortex-A9 Dual-Core가 내장된 Exynos4210[6]를 사용하였다. SoC에 리눅스 커널 4.6버전을 설치하고 Arm-linux-g++ 4.4.1 컴파일러를 사용하여 성능을 측정하였다.

<표 1> S5PV210에서 얼굴검출 성능 측정 (단위:ms)

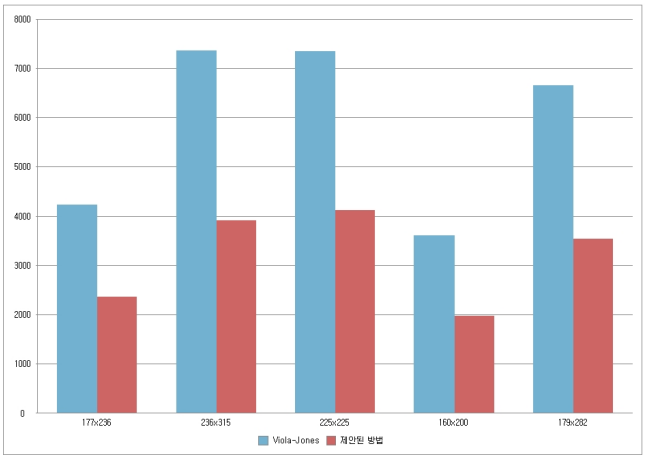
크기	Viola-Jones	제안된 방법	검출유무
177x236	7,186	7,226	O
236x315	12,354	12,701	O
225x225	13,346	13,611	O
160x200	6,076	6,165	O
179x282	11,154	11,215	X
평균	10,023	10,183	-



(그림 8) S5PV210에서 얼굴검출 성능결과 그래프

<표 2> Exynos4210에서 얼굴검출 성능 측정 (단위:ms)

크기	Viola-Jones	제안된 방법	검출유무
177x236	4,235	2,361	O
236x315	7,362	3,904	O
225x225	7,341	4,125	O
160x200	3,610	1,977	O
179x282	6,651	3,539	X
평균	5,840	3,181	-



(그림 9) Exynos4210에서 얼굴검출 성능결과 그래프

<표 1>과 같이 Single-Core에서는 제안한 멀티프로그래밍 얼굴검출보다 기존의 Viola&Jones 알고리즘이 미세하게 더 빠른 성능을 보였다. 그 이유는 제안하는 방법은 POSIX Thread를 사용하여 병렬처리 연산을 하기 때문에 Single-Core에서는 의미가 없고 또한 POSIX Thread를 생성하고 관리하는데 시간이 소요되어 효과가 없다. 하지만 Dual-Core를 가진 Exynos4210 SoC칩에서 성능을 측정하면 기존의 Viola-Jones 알고리즘에 비해 평균 1.8배 증가하는 결과를 보였다.

Core의 개수가 증가하고 병렬 처리량이 많아지면 현재의 제안한 얼굴검출 방법은 더욱 빠른 속도가 측정될 것으로 생각된다. 얼굴검출 성공률의 경우는 모두 동일한 학습데이터 파일을 사용하였고 개발환경 또한 동일하여 기존의 소스와 동일한 검출 성공률을 보였다. 영상 데이터의 얼굴 방향이 정면을 향하고 일정수준 이상의 크기와 화질을 가진 데이터는 99%의 얼굴검출 성공률을 보이지만 그 외의 얼굴의 각도가 달라진 사진 또는 배경화면에 따라 성공률은 많이 달라진다.

5. 결론

본 논문에서는 Multi-Core에 효과적인 얼굴검출 방법을 제안하였다. 멀티프로그래밍을 설계하면서 가장 중요한 부분 중 하나는 병렬처리 부분에 대한 데이터 의존성이다. Viola-Jones 알고리즘은 여러 부분이 반복적으로 동작하고 데이터가 독립적인 부분들이 존재함을 알 수 있다. 이런 영역을 한정하여 병렬처리를 함으로써 기존의 얼굴검출 알고리즘에 비해 속도가 개선됨을 확인하였다. 향후 얼굴검출 알고리즘 이외에도 다양한 알고리즘에 병렬처리를 통한 속도개선을 연구할 것이고 코어의 개수가 증가함에 따라 속도는 더욱 빨라질 것으로 예상된다.

참고문헌

[1] M. Tusch, “High-Performance Image Processing on FPGAs,” Xcell Journal, Vol.57, No.2, pp.42-44, April 2006.

[2] P. Viola, M.J. Jones, “Robust Real-Time Face Detection”, International Journal of Computer Vision, Vol.57 No.2, pp.137-154, May 2004.

[3] D. Chai, K.N. Ngan, “Locating Facial Region of a Head-and-Shoulders Color Image,” Proc. Third Int’l Conf. Automatic Face and Gesture Recognition, pp. 124-129, 1998.

[4] M. H. Yang, D. J. Kriegman, and N. Ahuja, “Detecting faces in images: a survey”, In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.24, No.1, pp.34-54, January 2002.

[5] Cortex-A9 MPCore Technical Reference Manual, http://www.arm.com

[6] Samsung Exynos 4210 RISC Microprocessor User’s Manual, http://www.samsung.com/sec

[7] Bo-Cheng Charles Lai, Chih-Hsuan Chiang, Guan-Ru Li, “Data Locality Optimization for A Parallel Object Detection on Embedded Multi-Core Systems“, Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on, pp.576-579, July 2011.