다빈치 프로세서 기반 스마트 카메라에서의 객체 추적 알고리즘의 최적 구현

An Optimal Implementation of Object Tracking Algorithm for DaVinci Processor-based Smart Camera

이병은*, Thanh Binh Nguyen*, 정선태*
*숭실대학교 임베디드 실시간 컴퓨팅연구실

Lee Byung-Eun*, Thanh Binh Nguyen*, Chung Sun-Tae*
*Embedded Real-time Computing Lab., SoongSil University

요약

다빈치 프로세서는 임베디드 멀티미디어 응용 구현 프로세서로 많이 사용된다. ARM 9 코어 및 DSP 코어의 듀얼 코어로 되어 있어 ARM 코어 에서는 주변 장치 제어, 비디오 입출력 제어, 네트워킹 등을 지원하며, DSP 코어는 보다 효율적인 디지털 신호 처리 연산을 지원한다. 본 논문에서는 본 저자들의 연구실에서 만들고 있는 다빈치 프로세서 기반의 스마트 카메라에 있어서 객체 추적 알고리즘의 최적 구현 방안 노력을 기술한다. 본 논문의 스마트 카메라는 입력 영상에서 관심 객체를 검출하고 이를 추적하며, 분류하고 감시 구역에 침입한 경우 이를 IP 프로토콜로 원격 클라이언트에게 통보하는 기능을 보유한다. 객체 추적은 전방 마스크 추출, 전방 마스크 교정, 연결 요소 레이블링, 블롭 지역 계산 등 계산량이 많은 절차들로 구성되어 효율적으로 구현되지 않으면 실시간 처리가 힘들다.

Abstract

DaVinci processors are popular media processors for implementing embedded multimedia applications. They support dual core architecture: ARM9 core for video I/O handling as well as system management and peripheral handling, and DSP C64+ core for effective digital signal processing. In this paper, we propose our efforts for optimal implementation of object tracking algorithm in DaVinci-based smart camera which is being designed and implemented by our laboratory. The smart camera in this paper is supposed to support object detection, object tracking, object classification and detection of intrusion into surveillance regions and sending the detection event to remote clients using IP protocol. Object tracking algorithm is computationally expensive since it needs to process several procedures such as foreground mask extraction, foreground mask correction, connected component labeling, blob region calculation, object prediction, and etc. which require large amount of computation times. Thus, if it is not implemented optimally in Davinci-based processors, one cannot expect real-time performance of the smart camera.

Ⅰ. 서론

스마트 카메라는 종래의 네트워크 카메라와 달리 캡쳐한 영상을 해석하여 상황을 인지하고 이에 따른 실시간 조치가 가능한 지능 기능을 추가적으로 갖춘 카메라이다. 종래의 네트워크 카메라는 현재 시장에 많은 제품이 출시되어 있으며 [1,2], 이에 대한 관련 연구도 많이 이루어지고 있다 [3,4]. 그런데, 종래의 네트워크 카메라는 획득 영상의 정보를 해석하지 않고 단순히 압축·전송하기 때문에, 서버에서 네트워크 카메라에서 전송

된 영상을 모니터링 하여 사람 침입, 위험한 행동 분석 등을 수행하여 필요한 조치를 취하게 된다. 전송 에러가 발생하거나, 여러 네트워크 카메라에서 전송되는 비디오 양의 부하 때문에 서버에서 모니터링이 늦어지는 경우가 발생할 수 있다. 이 경우에 실시간 감시가 힘들게 된다. 따라서 영상 획득 시에 즉시 영상데이터를 해석하여 금지 구역 침입, 위험 행동 등의 상황을 인지할 수 있으면, 이에 따른 실시간 조치(경보, 이메일 전송 등) 가 가능해 질 것이다. 획득한 영상을 해석하고 상황을 인지하는 작업은 컴퓨터 비전 기능으로 카메라에 컴퓨터 비전 기능을 추가하려는 연구는 최근 들어 매우 활발히 진행되고 있다 [5,6].

DaVinci 프로세서는 TI 사에서 제공하는 듀얼 코어 미디어 프로세서로 내부에 ARM 9 코어 및 C64+코어를 제공하고 있다[7]. 또한, 네트워킹, 각종 주변 장치 제어뿐만 아니라 비디오 및 오디오 데이터 입력 및 출력을 지원하고 있어서 현재 임베디드 멀티미디어 시스템 개발 플랫폼으로 매우 인기가 높다. 본 논문의 저자들의 연구실에서는 현재 DaVinci 프로세서 DM6446 다빈치 프로세서 기반의 스마트 카메라를 개발하고 있다 [8].

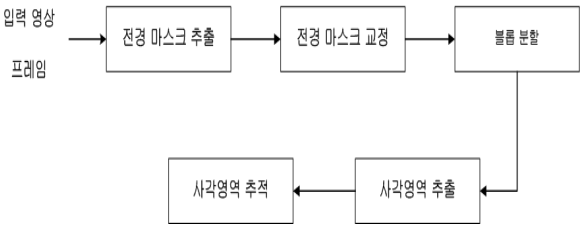
객체 추적 알고리즘은 전경 마스크 추출, 전경 마스크 교정, 연결 구성 요소 레이블링, 블롭 사각 영역 계산, 객체 위치 추정 등의 과정들을 필요로 하며 각각의 과정은 많은 연산량이 필요로 한다. 따라서, 객체 추적 알고리즘은 효율적인 연산 처리를 위하여 DM6446 다빈치 프로세서 기반 스마트 카메라에서는 DM6446의 DSP 코어에서 구현되게 된다. 그러나, 전체적으로 연산 처리 요구량이 많아 실시간 처리 성능을 위해서는 전체 S/W 구조의 효율적인 설계 및 DSP 코어에서의 객체 추적 알고리즘의 최적 구현을 필요로 한다.

본 논문은 DM6446 다빈치 프로세서 기반 스마트 카메라에 실시간 객체 추적을 위한 전체 S/W 구조 설계 및 객체 추적 알고리즘의 DSP 코어에서의 최적 구현 방안에 관한 연구 결과를 기술한다.

본 논문의 구성은 다음과 같다. 2절에서는 영상 감시에서의 객체 추적 알고리즘 실행 과정에 대해 소개하며, 3절에서는 전체 S/W 구조 설계에 대해 기술하고 4절에서는 DSP 코어에서의 객체 추적 알고리즘의 최적 구현 방안에 대해 기술한다. 마지막으로 5절에서 결론 및 향후 연구방향이 주어진다.

Ⅱ. 영상 감시에서의 객체 추적 알고리즘

지능형 영상 감시에서는 관심 있는 객체를 검출하고 이를 추적하여 감시 구역에 침입하는 경우에 이를 원격 클라이언트에게 통보하는 기능을 지원하며, 본 논문의 스마트 카메라는 임베디드 지능 영상 감시 시스템이다. 객체 추적을 위해서는 여러 과정의 수행이 필요하다[9] (그림 1 참조).



▶▶ 그림 1. 객체 추적 알고리즘의 실행 과정

먼저 입력된 영상 데이터로 부터 전방 마스크 검출을 필요로 한다. 전방 마스크(foreground mask)는 전방 픽셀들의 모임을 나타내는 이미지를 말한다. 입력 영상 데이터의 각 픽셀이 기존 배경(background) 과는 픽셀(전방 픽셀)인지 기존 배경과 유사한 픽셀(배경 픽셀)인지의 구분은 배경 모델과의 매칭을 통해 이루어진다. 즉, 입력 영상 데이터 픽셀이 배경 모델과 매칭되면 배경 픽셀로, 그렇지 않으면 전경 픽셀로 분류한다. 배경 모델이 정교할수록 전경 마스크 추출이 정확해지지만, 정교한 배경 모델은 온라인 적응적 갱신까지를 요구하기 때문에 계산량이 매우 많이 필요로 되어 실시간 임베디드 시스템처리가 힘들다. 본 논문에서는 적응적 배경 모델 갱신이 이루어지면서, 비교적 계산량 처리가 다빈치의 DSP 코어에서 실시간으로 처리가 가능한 가우시안 혼합 모델 기반 배경 모델[10]을 사용하였다. 배경 모델과의 매칭으로 추출된 전경 마스크는 배경 모델의 비완벽성으로 전경 픽셀이 추출되지 못하거나, 배경 픽셀이 잘못 전경 픽셀로 추출되므로 전경 마스크 교정이 필요하다. 즉, 미추출된 전경 픽셀들 때문에 생기는 전경 블록들의 구멍을 채워 넣거나, 전경 픽셀로 잘못 추출된 배경 픽셀들은 제거하여야 한다. 잘못 추출된 전경 픽셀들은 전경 블롭의 삐져나온 픽셀들 또는 전경 마스크에서 고립된 픽셀 영역으로 나타나게 된다. 보통 전경 마스크 교정을 위해 열림(opening) 및 닫힘

(closing) 등의 모폴로지 연산[11]을 이용한다. 전경 마스크 교정 후에는 전경 마스크에서 분리된 블롭들을 분할(segment)하여야 한다. 여기서 블롭은 연결된 전경 픽셀들의 모임을 말하며, 분할된 블롭들이 영상 감시에서 관심 있는 객체들을 구성하게 된다. 블롭의 분할에는 연결 요소 레이블링 알고리즘이 필수적으로 이용된다[11]. 블롭이 분할되면, 블롭을 최소로 포함하는 사각 영역을 계산하고, 이 영역을 계속 추적하게 된다. 다음 입력 영상 프레임에서 역시 블롭을 검출하고 블롭을 포함하는 사각 영역들을 계산하여 이전 프레임에서 구해진 사각 영역들과 동일 여부를 판별하여 객체 추적을 달성한다.

연속된 2개 프레임에서 각각 검출된 사각 영역들의 동일 여부 판별은 사각 영역의 위치 추정 정보, 히스토그램 정보 등을 이용하게 된다.

Ⅲ. 객체 추적 관련 S/W 구조 설계

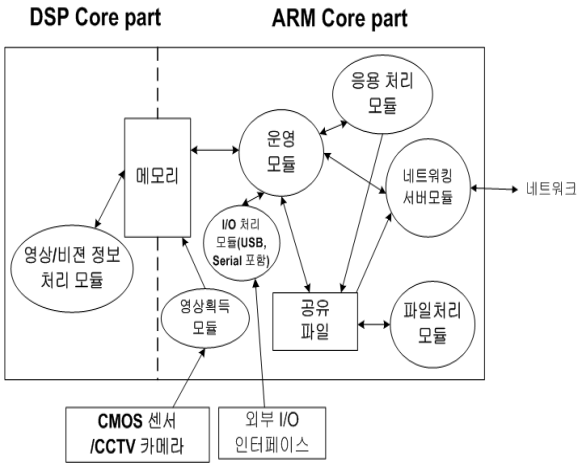
3.1. 스마트 카메라 S/W 구조

스마트 카메라의 S/W는 크게 시스템 S/W, 응용 S/W 로 구성된다. 시스템 S/W 로는 다빈치 DM6446의 ARM 코어 및 DSP 코어의 시스템 S/W 로, 부트로더, ARM 용 OS, 장치 드라이버, DSP 운영체제 및 ARM과 DSP의 동기화에 필요한 핵심 모듈인 DSPLINK[12] 등이 있다. 또한 응용 S/W 로는 ARM 부분에 탑재될 시스템 운영 모듈, 네트워킹 서버 모듈, 응용 처리 모듈, 영상 획득 모듈, I/O 처리 모듈, 파일 처리 모듈, 영상 및 비전 정보 처리 모듈 등이 있으며, 이 가운데, 연산량이 많아 효율적인 연산 처리를 필요로 하는 영상 및 비전 정보 처리 모듈은 DSP에, 나머지는 ARM에 구현된다 (그림 2 참조).

객체 추적 알고리즘 S/W 는 그림 2에서 DSP 코어 부분의 영상/비전 정보 처리 모듈에서 구현된다.

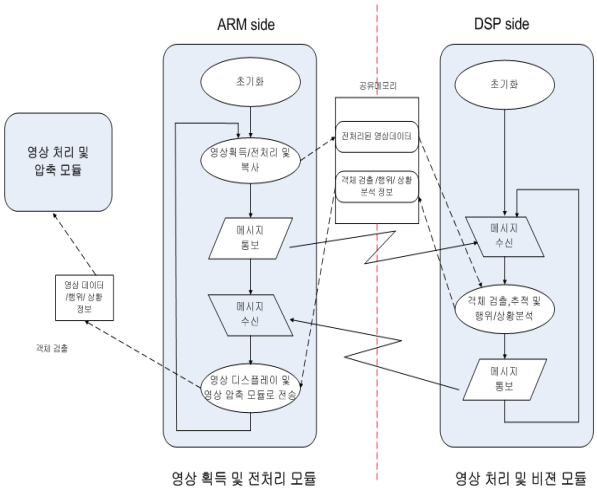
3.2. 객체 추적 관련 S/W 동기화 구조

다빈치 프로세서 기반 스마트 카메라에서 객체 추적



▶▶ 그림 2. 스마트 네트워크 카메라 S/W 구성

관련 S/W 동기화 동작 구조는 아래 그림3과 같다.



▶▶ 그림 3. 객체 추적 관련 S/W 동기화 구조

먼저 ARM 코어 부분에서 영상 입력을 수행하여 영상 센서로부터 영상 데이터를 획득한다. 본 논문의 스마트 카메라는 입력 영상 획득 이미지 센서로 Micron사의 MT9T001 을 사용하였다. MT9T001은 입력 영상 데이터를 최대 3M 픽셀 및 10비트 베이어 패턴으로 출력해 주는 CMOS 이미지 센서이다[13]. 입력 영상 데이터는 640x480의 베이어 패턴이다. DM6446의 VPFE의 Preview 엔진을 이용하여 YCrCb 4:2:2 packed 모드 포맷으로 전환한 후에 다시 Resize 엔진을 이용하여 160x120 사이즈의 YUV 4:2:2 packed 모드로 고친 후에 여기서 160x120 크기의 그래이 레벨 영상 데이터

(luminance, Y)를 뽑아 이를 공유 메모리에 복사하고 이를 DSP 코어 쪽으로 메시지를 통보한다. DSP 코어 S/W 모듈은 ARM 코어 S/W로부터 메시지를 수신하면, 공유메모리에서 160x120 사이즈 그레이레벨 영상 데이터를 읽어 와 객체 추적 알고리즘을 수행한 후에 얻은 객체 정보를 공유 메모리에 저장 후 ARM 코어 S/W 쪽으로 메시지를 통보한다. ARM 코어 S/W 모듈은 메시지를 수신하면 공유 메모리로부터 객체 정보를 읽어와 이 정보를 통해 필요에 따라 객체 추적 영상 로칼 디스플레이, 영상 압축, 감시 구역 침입 이벤트의 원격 클라이언트로의 통지 등을 수행한다.

Ⅳ. 객체 추적 알고리즘 최적화 방안

4.1. 코드 최적화

1) 고정 소수점 연산 활용

2.3절에서 살펴 본바와 같이, 다빈치 DSP 코어는 C64+ 로 고정 소수점 연산을 지원하며, VLIW 구조를 갖는다. 그림 1로 설명되는 객체 추적 알고리즘은 보다 정확한 계산을 위해 많은 실수 연산을 필요로 한다. 그러나, C64+ DSP 코어는 부동소수점 연산을 지원하지 않으므로, 실수 연산은 보통 S/W 에뮬레이션으로 구현 되는 데, 많은 사이클을 필요로 한다. 따라서, 본 논문에서는 float 와 double 은 고정 소수점 표현으로 변환하고 실수 연산은 고정소수점 연산 지원을 이용하여 계산량을 줄였다. 고정소수점 표현으로 Q20을 사용하였다.

2) SIMD 명령 이용

C64+ 는 8개의 바이트를 동시에 적재하거나 저장할 수 있다. 본 논문의 객체 추적은 그레이레벨 8비트 픽셀 데이터를 사용하므로, 동시에 8개 픽셀 데이터를 적재하거나 저장이 가능하다. 또한, 2개의 16비트 동시 곱셈, 4개의 8비트 동시 곱셈, 4개의 8비트 동시 덧셈/뺄셈, 4개의 8비트 뺄셈후 절대값 동시 연산, 4개 바이트별 동시 비교 등의 SIMD 명령이 가능하다. 즉, C64+는 동시에 4개 픽셀에 대한 산술연산(덧셈,뺄셈,곱셈, 나눗셈) 및 논리연산(AND, OR, XOR, 비교 등)이 가능한 명령어를 지원한다. 내부고유함수(intrinsics)는 C64+ 내장 명령어로 직접 매핑되는 C 언어 특별함수를 말한다. 위의 C64+ 의 많은 효율적인 SIMD 명령어들이 C 언어에서 내부고유 함수로 호출 가능하도록 지원하므로 본 논문의 객체 추적 구현은 가능하면 이를 사용하였다.

3) VLIW

C64+ 는 동시에 8개의 명령 수행이 가능하다. 루프의 경우, VLIW를 가장 효율적으로 이용할 수 있는 스케줄링 방법은 S/W 파이프라이닝[14]을 이용하는 것이다. 본 논문의 객체 추적 알고리즘은 많은 루프를 이용한다. 먼저 컴파일러의 S/W 파이프라이닝 지원 옵션을 이용하였으며, 프로파일링 하여 자주 쓰이는 루프에 대해서는 어셈블리로 S/W 파이프라이닝 스케줄링을 하였다.

4) DSP/Imaging 라이브러리 활용

TI 에서는 DSP에서 처리되는 디지털 신호 처리 및 영상 처리 알고리즘 중 자주 사용되는 주요 알고리즘(FFT 연산, SAD 연산, Convolution, 모폴로지 연산, DCT, IDCT 등)들을 C64+ 코어에 최적화 구현하여 라이브러리로 제공하고 있다. 본 논문의 객체 추적 구현은 가능하면 이를 이용하여 구현하였다.

4.2. 메모리 최적화

1) 내부 메모리 최적화

다빈치 C64+ 코어는 내부에 32KB L1P 캐쉬, 80KB L1D 캐쉬, 64KB L2 통합 메모리를 지원하고 있다[15]. L2 메모리 64KB는 일부 또는 전부를 L2 캐쉬 또는 내부 메모리로 구성이 가능하다. 내부 메모리의 경우, 접근에 6~8 CPU 사이클 (C64+ 의 경우 1 사이클은 1.68msec(594MHz)임)밖에 소요되지 않아 외부 DDR2 메모리 접근보다 훨씬 빠르다. 따라서 자주 호출되는 함수나 데이터는 L2 내부 메모리에 배치하였다.

2) 캐쉬 최적화 [16]

객체 추적 알고리즘은 비디오 스트림 처리 작업이므로 처리 데이터는 매번 바뀌나 같은 내용의 반복 작업 처리가 많다. 따라서, 한번 캐쉬된 명령어들은 조만간 재사용될 가능성이 무척 많다. 따라서, 조만간 재사용 가능성이 있는 동안에는 해당 L1P 캐쉬 블록의 방출이

적게 발생하도록 노력하여야 한다. 그런데, C64+는 L1P 캐쉬는 direct-mapped 방식이기 때문에 다른 n-way (n=2,3,4, …) set-associative 방식 캐쉬보다 충돌 가능성이 더 높다. 따라서, 객체 추적과 같은 비디오 스트림 처리 응용의 경우, L1P 캐쉬의 메모리 액세스 시간 개선(감소)에는 충돌 캐쉬 미스(conflict cache miss)를 줄이는 게 매우 중요하다. 현재의 컴파일러 와 링커는 자동적으로 캐쉬 충돌에 대한 고려를 하지 못하기 때문에, 링커에 의한 부적절한 프로그램 코드 메모리 주소 배치가 프로그램 수행 중에 충돌 캐쉬 미스를 초래할 수 있다. 따라서, 충돌 캐쉬 미스 경우는, 컴파일러 옵션을 조정하여 충돌 나는 함수들을 메모리에 연속적으로 배치하도록 하면, 미스 율을 크게 개선시킬 수 있다.

객체 추적 알고리즘 구현의 경우, 프로파일링 하여 빈번하게 사용되고 계산량이 많아 수행 시간 성능 개선에 주로 고려되어야 함수들을 찾아내고, 이들에 대해 수행중 충돌 가능성에 대해 조사하고, 충돌 가능성이 있는 함수들을 메모리에 연속적으로 배치되도록 컴파일러의 옵션을 통해 조정하여야 한다. 즉, 링커에 의해 생성되는 프로그램 메모리 배치도(map 파일)에서 메모리 주소와 L1P 캐쉬의 매핑 관계를 분석하여, 이들 함수중 수행중 L1P 캐쉬에서 서로 충돌날 수 있는(즉, 동일한 캐쉬 블록에 할당되는) 가능성을 조사하고 충돌이 나는 경우, 충돌을 줄일 수 있도록 메모리 연속적으로 배치되도록 컴파일을 통해 조정하였다. 이 경우, 실험을 통해 캐쉬 미스율이 개선되었음을 확인할 수 있었다.

4.3. 알고리즘 최적화

객체 추적 알고리즘을 기본적으로 다빈치 DSP 코어에 맞게 최적화하지 않으면, 코드 최적화 및 메모리 최적화만으로는 실시간 동작을 이루기가 힘들다. 본 논문에서는 객체 추적 알고리즘에서 가장 시간이 많이 소요되는 부분의 하나인 전경마스크 교정과 블록 분할을 동시에 수행할 수 있는 개선된 알고리즘을 개발하고 이를 구현하였다[17]. 이 알고리즘 개선으로 기존 알고리즘보다 전경 마스크 교정 및 블롭 분할에 한 프레임 당에 5배 이상의 처리 시간 개선을 이루었다. 또한, 빠르고 효과적인 그림자 제거 알고리즘의 개선도 이루었다[18].

V. 결론

본 논문에서는 다빈치 프로세서 기반 스마트 카메라에서의 객체 추적 알고리즘의 최적 구현 연구 방안을 살펴보았다. 향후 성능 분석을 통해 보다 최적화된 구현을 수행하고 이를 보고할 예정이다.

■ 참 고 문 헌 ■

[1] AXIS Homepage http://www.axis.com

[2] WEBGATE Homepage http://www.webgateinc.com

[3] Y.Kakimura, et al., "A Real-Time MPEG1 Ethernet Camera System", IEEE, 1999

[4] Mi-Joung Choi, Hong-Taek Ju, Hyun-Jun Cha,Sook-Hyang Kim, Won-Ki Hong, "An Efficient Embedded Web Server for Web-based Network Element Management", IEEE, 2000

[5] S. Hengstler and H. Aghajan, "A Smart Camera Mote Architecture for Distributed Intelligent Surveillance," ACM SenSys Workshop on Distributed Smart Cameras (DSC), Oct. 2006.

[6] R. Kleihorst, and et al., "Applications of wireless Smart Cameras (Networks)," ICASSP 2007, pp. IV-1373-IV-1376.

[7] TMS320DM6446 datasheet, http://www.ti.com

[8] 유희재, 정선태, 정수환, "다빈치 기반 스마트 카메라 S/W 설계 및 구현", 2008년 한국컨텐츠학회 춘계학술대회

[9] A.Yilma, O. Javed and M. Shah, "Object tracking: A survey," ACM Computing Surveys, Vol. 38, Issue 4, 2006.

[10] C. Stauffer C, W.E.L Grimson, "Adaptive background mixture models for real-time tracking," 1999 IEEE CVPR.

[11] L. Shapiro and C. Stockman, *Computer Vision*, Prentice Hall, 2001.

[12] TI datasheet, LNK 058 USR, “DSP/BIOS LINK User's Guide”.
[13] Micron, MT9T001 datasheet, http://download.micron.com/pdf/datasheets/imaging/MT9T001_3100_DS.pdf
[14] *TMS320C6000 Programmer's Guide.* No. SPRU189E, Texas Instruments, Jan., 2000.
[15] TI datasheet, spru610c, "TMS320C64x DSP Two-Level Internal Memory Reference Guide"
[16] TMS320C6000 DSP Cache User's Guide, Texas Instruments
[17] Thanh Binh Nguyen, Sun-Tae Chung, "A fast and precise Blob Detection", 2009년 한국컨텐츠학회 춘계학술대회.
[18] Thanh Binh Nguyen, Sun-Tae Chung, "Cast Shadow Removal in Motion Detecion", 2009년 한국컨텐츠학회 춘계학술대회.