우적제거 알고리즘의 DSP 최적화

최동윤, 서승지, *송병철
인하대학교 전자공학과
*bcsong@inha.ac.kr

DSP Optimization of Rain Removal Algorithm

Dong Yoon Choi, Seung Ji Seo, *Byung Cheol Song
Department of Electronic Engineering, Inha University

요 약

객체의 인식을 위한 컴퓨터 비전 알고리즘은 안개와 비와 같은 기상이 좋지 않은 상황에서는 인식 성능이 떨어지고 있다. 이로 인하여 최근 악천후 환경에서 촬영된 영상으로부터 날씨 현상을 제거하는 기법들이 연구되고 있다. 빗줄기는 시공간적 무작위성으로 인하여 검출 및 제거가 어려운 현상이다. 또한 기존의 빗줄기 검출 및 제거 기법들은 대부분 고정된 카메라로부터 촬영된 영상을 대상으로 처리함으로써 자동차와 같은 움직임이 있는 촬영환경에서는 부적합하다. 최근에는 카메라나 객체의 움직임에 대응할 수 있는 빗줄기 검출 및 제거 알고리즘이 개발되고 있으나, 방대한 연산량이 필요하기 때문에 실시간이 불가능하다. 본 논문에서는 최근 연구되고 있는 카메라 움직임이 있는 환경에서 빗줄기 검출 및 제거 알고리즘을 DSP 환경에서 구현하고 내부 메모리 최적화와 EMDA 이용, 소프트웨어 파이프라인 등을 통해 최적화를 수행하여 실시간성을 보인다.

1. 서론

최근 자동차산업에서 ADAS(advanced driver assistance system)기술 분야가 각광받고 있으며 그 시장 또한 지속적으로 확대되고 있다. 위 기술들은 차선이나 표지판, 사물에 대한 검출 등 컴퓨터 비전 알고리즘에 대한 활용도가 높다. 이러한 기술들은 외부 환경에 영향을 받지 않고 사용되어야 하는데 비나 눈이 내리는 악천후의 기상 환경에서 촬영된 영상은 피사체에 대한 인식률이 크게 떨어진다. 따라서 최근 비와 눈과 같은 기상 조건에서 열화된 영상에서 컴퓨터 비전 알고리즘의 성능 향상을 위하여 전처리 단계로서 기상 환경을 제거할 수 있는 기법이 개발되고 있다.

특히 기상 환경 중 비와 같은 경우 시간적-공간적 무작위한 특성으로 인하여 영상 전체에 대한 처리를 통해 제거되기 어려운 현상이다. 이와 같은 문제를 해결하기 위해 영상으로부터 비를 검출하고 제거하기 위해 다양한 비 검출이나 제거 기법들이 제안되었다 [1-4]. 제안된 비 검출 및 제거 기법들은 크게 움직임에 적응적인 방법과 그렇지 않은 방법으로 나눌 수 있다. 움직임에 적응적이지 않은 방법은 인접한 프레임간의 차를 이용하여 시간 영역을 기반으로 한 비 검출 및 제거 기법 [1], 비의 공간적 주파수 특성을 이용한 비 검출 및 제거 기법 [2], 마지막으로 단일 영상에서 공간적인 특성을 공간영역에서 풀어낸 단일 영상 기반의 비 검출 및 제거 방법 [3] 등 이 제안되었다.

그러나 이러한 알고리즘은 방대한 연산량을 필요로 하여 실시간을 확보하기가 불가능 하여 컴퓨터 비전 알고리즘의 전처리로 사용하기 어렵다. 따라서 본 논문은 비 검출 및 제거 비법 중 가장 고성능의 기법인 Kim 방법 [4]에 기반한 비 검출 및 제거 알고리즘을 영상처리에서 범용적으로 사용하는 DSP 인 TMS320DM6437 보드상에서 최적화하여 궁극적으로 임베디드 시스템에서 전처리 과정으로 적용 가능한 비 제거 기법을 개발한다. DSP 측면의 최적화 단계에서는 DSP 고유의 내장 함수 이용, 소프트웨어 파이프라인에 용이한 코드 구조 구축, EDMA(extended direct memory access)의 활용 등을 이용해 고속기법을 구현한다.

2. 알고리즘의 DSP 최적화

TI 사의 보드인 TMS320DM6437은 연산을 담당하는 DSP 칩과 L2 단계의 캐시, 입력 비디오의 크기 변경이나 자동 초점 기와 같은 비디오 전문 처리 모듈인 VPSS를 하드웨어적으로 제공한다. 또 외부에 McASP, McBSP, UART와 같은 I/O와 DDR2 외부 메모리를 가지고 있고 594MHz 동작 속도, 32KB 의 L1P, 80KB L1D, 128KB L2, 64ch EDMA 지원 등의 사양을 가진다.

DSP는 CPU 구조에 따른 여러 가지 고유의 내장 함수를 지원한다. 대부분의 경우 컴파일러가 코드를 컴파일할 때 자동으로 최적의 내장 함수를 사용할 수 있게 하지만, 일부 경우에는 사용자가 직접 작성해줘야 효율적이다. 본 논문에서는 가중치 합을 수행하는 부분에서 동시에 많은 데이터를 연산해 동일 클록동안 많은 연산 결과를 얻기 위해 8bit 데이터 8개를 64bit 데이터로 붙이고 붙은 64bit 데이터를 8bit 단위로 덧셈을 수행한 후 다시 8bit로 반환하는 과정을 내장 함수를 이용해 구현하였다.

TMS320DM6437의 경우 내부 메모리로 L2 구조의 캐시 메모리가 있다. 이를 사용자가 선택함에 따라 캐시로 사용할지 아니면 내부 메모리로 사용할지 결정할 수 있다.

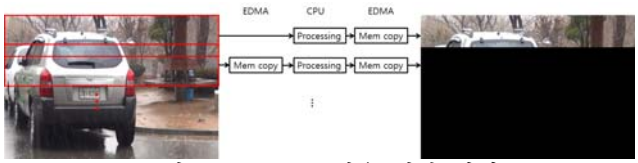

그림 1. EDMA 이용 영상 처리

영상 데이터의 정보는 매우 크기 때문에 내부 메모리로 사용이 불가능하여 EVM 보드를 통해 외부메모리를 통해 사용해야 한다. TI 보드에서는 EDMA 라는 메모리 지원 툴이 따로 존재한다. 이는 외부 메모리에 있는 데이터를 내부 메모리로 가지고 오는데 CPU 의 연산량을 소모하지 않고 독립적으로 처리해 메모리를 불러오는 연산을 CPU 연산 뒤로 숨김으로써 외부 메모리에 의한 메모리 오버헤드를 줄일 수 있다. 본 논문에서는 검출 파트 전반을 해당 알고리즘을 적용하여 연산량 최적화를 수행하였다. 검출 파트의 경우 입력 영상 하나만 있으면 검출을 할 수 있기 때문에 대상 주소가 복잡하게 구성되지 않는 반면, 제거 파트는 검출 지도, 현재 영상을 포함한 앞뒤의 인접한 여러 영상이 필요하기 때문에 EDMA 를 적용하기에 적절하지 않기 때문이다.

영상 전체를 내부 메모리에 저장할 수 없기 때문에 EDMA 를 이용해 영상의 일부만을 내부 메모리로 저장해 연산하게 된다. 이런 과정은 그림 1.과 같이 수행된다. 먼저 외부 메모리에 있는 영상 데이터를 내부 메모리 크기에 맞게 적당한 크기만큼 복사한다. 그런 후 복사된 데이터를 이용해 영상 처리를 수행한다. 처리된 데이터를 다시 외부 메모리로 복사하고, 사용된 내부 메모리에 저장된 데이터를 새로운 데이터로 덮어쓴다. 이때 EDMA를 이용한다면 CPU가 영상 처리를 수행할 때 외부 메모리와 내부 메모리 사이에서 데이터를 복사할 수 있다. CPU는 내부 메모리에 있는 데이터를 처리하는 역할만을 담당할 수 있기 때문에 처리 속도가 빨라진다. 이때 일반적으로 영상을 세로 방향으로 분할해 사용한다. 분할된 영상 크기는 내부 메모리 공간보다 작아야 하며, 클수록 더 적은 횟수의 반복 횟수가 필요하기 때문에 내부 메모리에 꼭 배치해야 할 변수나 함수가 차지하고 있는 공간을 제외한 공간만큼의 크기로 결정한다. 이때 처리되는 도중 데이터가 손실됨을 막기 위해 처리중인 데이터와 다음에 처리될 데이터를 각각 저장한다. 또, 입력과 출력에 각각의 버퍼가 필요하기 때문에 총 4개의 내부 메모리 버퍼가 필요하다. 본 논문에서 사용하는 내부 메모리 공간이 128KB의 L2 캐시를 사용하고 한다. 여기서 look-up table과 같은 내부 메모리에 배치되어야 하는 공간을 제외하고 하나의 분할 영상 버퍼를 32.4KB 만큼의 공간으로 배치하였다.

3. 모의실험 결과

우리는 DSP 상에서 구현한 알고리즘의 수행시간을 비교하였는데 실험 동영상으로는 해상도가 720x384 인 영상 하나와 720x480 인 영상 두 개를 사용하였고. PC 환경에서의 수행시간과 DSP 에서 최적화를 수행하지 않은 경우와 수행한 경우의 연산 시간을 비교하였다. PC 상에서의 시간 측정은 3.2GHz 로 동작하는 i5-3470 CPU 와 16GB RAM 의 사양을 가진다. 수행 시간을 비교한 결과는 표 1 과 같다.

DSP 상에서 연산 시간은 연속된 3장의 영상에 대해 기법 동작 부분에서 동작 클록 수를 기록해 CPU 동작 클록으로 나누어 측정하였다. CPU 동작 속도에 비교했을 때 DSP 동작 시간은 5배 정도 느려짐을 기대할 수 있다. 하지만 실제로는 약 8~10배의 연산 증가가 이루어진다.

표 1. PC와 DSP상에서의 수행시간 비교

Test video	PC	DSP optim. X	DSP optim. O
Video1	0.003s	0.033s	0.012s
Video2	0.005s	0.041s	0.018s
video3	0.0053s	0.042s	0.018s

원인으로는 PC에서는 캐시의 용량이 매우 크기 때문에 필요한 데이터를 캐시에 많이 저장할 수 있어서 메모리 오버헤드가 크지 않고 PC에서는 복잡한 CPU 구조에 기인한 강력한 소프트웨어 파이프라인이 가능한 반면, DSP는 단순한 CPU 구조를 가지고, PC 와 동일한 수준의 소프트웨어 파이프라인이 불가능한 것으로 볼 수 있다. 본 논문에서 수행한 DSP 단계의 최적화를 통해 약 3배의 연산 시간 측면의 이득을 보았다. 이 결과는 충분히 전처리 과정으로 사용할 수 있을 정도의 연산시간임을 확인 할 수 있다.

4. 결론

본 논문은 비가 오는 환경에서 촬영된 영상으로부터 빗줄기를 제거하는 알고리즘을 TMS320DM6437 에서 구현하였고 메모치 최적화 및 EDMA 등을 이용하여 최적화를 수행하였고 실험결과에서 임베디드 환경인 DSP 상에서 실시간성을 검증하였다. 향후 컴퓨터 비전 알고리즘에 전처리 과정으로 수행하여 본 제안 기법으로 인한 인식성능의 향상을 확인하는 연구를 진행할 것이다.

감사의 글

본 연구는 2012년도 정부 (교육부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행되었음 (과제 번호 2012000446).

참고문헌

[1] K. Garg and S. K. Nayar, " Detection and removal of rain from videos," *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, pp. 528-535, Jun. 2004.

[2] P. Barnum, T. Kanade, and S. G. Narasimhan. " Spatio-temporal frequency analysis for removing rain and snow from videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007.

[3] L. W. Kang, C. W. Lin, and Y. H. Fu, " Automatic single-image based rain streaks removal via image decomposition," *IEEE Trans, Image Process.*, vol. 21, no. 4, pp. 1742-1755, Apr. 2012.

[4] H. G. Kim, S. J. Seo, and B. C. Song, " Multi-frame de-raining algorithm using a motion-compensated non-local mean filter for rainy video sequences," *J. Visual Commun. Image Represent.*, vol. 26, pp. 317-328, Jan. 2015.