Journal of The Institute of Electronics and Information Engineers Vol.52, NO.9, September 2015 http://dx.doi.org/10.5573/ieie.2015.52.9.096
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2015-52-9-12

비 검출 및 제거 알고리즘의 DSP 최적화

(DSP Optimization for Rain Detection and Removal Algorithm)

최 동 윤*, 서 승 지, 송 병 철**

(Dong Yoon Choi, Seung Ji Seo, and Byung Cheol Song©)

요 약

본 논문은 비 검출 및 제거 알고리즘의 DSP 최적화 기법을 제안한다. 우리는 카메라 움직임이 있는 환경에서 비 검출 및 제거 기법을 제안하고, 알고리즘 레벨 및 DSP 레벨에서 최적화를 수행한다. 제안하는 기법은 알고리즘 측면에서 기존에 활용하던 라벨링을 블록 단위의 이진 패턴 분석을 통해 오 검출 영역을 제거하는 방식으로 대체하였고 고속 움직임 추정 알고리즘을 이용하여 연산 시간을 개선하였다. DSP 측면에서는 내부 메모리 최적화와 EMDA이용, 소프트웨어 파이프라인 등을 통한 최적화를 통해 임베디드 환경에서 실시간성을 보이며 실험 결과에서는 제안 기법의 성능과 시간 측면의 우수성을 보여준다.

Abstract

This paper proposes a DSP optimization solution of rain detection and removal algorithm. We propose rain detection and removal algorithms considering camera motion, and also presents optimization results in algorithm level and DSP level. At algorithm level, this paper utilizes a block level binary pattern analysis, and reduces the operation time by using the fast motion estimation algorithm. Also, the algorithm is optimized at DSP level through inter memory optimization, EDMA, and software pipelining for real-time operation. Experiment results show that the proposed algorithm is superior to the other algorithms in terms of visual quality as well as processing speed.

Keywords : rain streaks, rain detection, rain removal, DSP optimization, real-time implementation

Ⅰ. 서 론

최근 비와 눈과 같은 악천후의 기상 조건에서 촬영되어 열화된 영상으로부터 컴퓨터 비전 알고리즘의 성능을 향상시키기 위한 전처리로서 비와 눈 등을 제거할 수 있는 기법들이 개발 되고 있다. 특히 비와 같은 경우 시간적-공간적 무작위성으로 인하여 예측이 어렵고 영상 전체에 대한 처리를 통하여 제거하기가 어려운 현상이다. 이와 같은 문제를 해결하기 위하여 영상으로부터 빗줄기를 검출하고 제거하는 다양한 기법들이 제안되었다. 제안된 비 검출 및 제거 알고리즘은 크게 움직임에 적응적인 방법과 그렇지 않은 방법으로 나뉠 수 있다. 움직임에 적응적이지 않은 방법은 다시 시간적-광학적 특성을 이용한 방법, 시간적-색상적 특성을 이용한 방법, 그리고 주파수 영역 기반 방법 등으로 나뉠 수 있다[1~6].

Garg와 Nayar는 인접 프레임 사이에서 차이 값과 비의 광학적 특성을 이용하여 비를 검출하는 방법을 제안하였다[1]. 그러나 단순히 프레임 사이의 차이에만 의존하기 때문에 비의 양이 많거나 밝은 조명이 있는 경우 성능이 저하되고 수십 프레임의 정보를 이용하기 때문에 방대한 연산량을 필요하게 된다. Zhang 등은 시간적 특성과 색상적 특성을 결합한 기법을 제안하였다[2]. 이 방법은 시간적 정보와 색상적 정보를 모두 사용하여 고정된 카메라 환경이나 움직임이 없는 영상의 경우 우수

* 학생회원, ** 평생회원 인하대학교 전자공학과
(Department of Electronic Engineering, Inha University)
© Corresponding Author(E-mail: bcsong@inha.ac.kr)
Received ; July 14, 2014 Revised ; August 8, 2015
Accepted ; September 3, 2015

한 성능을 보이지만 영상에 움직임이 있는 경우 성능이 저하될 뿐만 아니라 영상에 아티팩트가 크게 발생된다. Barnum 등은 주파수 영역 기반의 빗줄기 검출 및 제거 방법을 제안하였다[3~4]. 기존 기법들과 다르게 시간적 특성을 고려하지 않아서 영상에 움직임이 있는 경우에도 뛰어난 성능을 보이지만 모델링이 적용되기 때문에 해당 모델에서 벗어나는 경우 성능이 검출이 되지 않는 단점이 있다. Xue 등은 시간적 특성과 웨이블릿 변환을 이용하여 카메라 움직임을 고려한 빗줄기 검출 방법을 제안하였다[5]. 이 방법은 웨이블릿 변환을 이용한 고주파 영역 검출 방법을 통해 초기 검출 영역 내 아웃라이어를 제거하였다. 그러나 시간적 특성을 이용한 초기 검출 방법에서 비가 많은 경우에 성능아 좋지 못하였다. Kim 등은 kernel regression 기법에 기반을 둔 단일 영상 비 검출 및 제거 기법을 제안하였다[6]. 하지만 근본적으로 그래디언트 기반의 방법이기 때문에 움직임 블러 특성을 갖는 빗줄기들을 검출하기 어렵다.

상기 알고리즘들의 공통적으로 큰 문제점은 방대한 연산량으로 실시간으로 처리하기가 불가능하다는 점이다. Kim 등은 실시간 처리가 가능한 카메라 움직임에 강인한 비 검출 및 제거 기법을 제안하였다[7]. 따라서 본 논문은 상기 Kim 방법을 알고리즘 레벨에서 최적화하여 PC환경에서 실시간성을 확보하고, TI사의 TMS320DM6437 보드 상에서 최적화하여 최종적으로 임베디드 시스템에서 실시간 동작이 가능한 비 검출 및 제거 방안을 제시한다. 알고리즘 측면의 최적화에서는 검출 과정에서 비의 밝기와 구조적 특성을 이용하여 영상 축소 및 이진 패턴 분석 등을 이용한 최적화를 수행하였고 제거 과정에서는 움직임 추정 과정을 예측 움직임 벡터 사용, 블록 축소 등을 이용하여 연산량을 줄였다. DSP 측면의 최적화 단계에서는 DSP 고유의 내장 함수 이용, 소프트웨어 파이프라인에 용이한 코드 구조 구축, EDMA (extended direct memory access)등을 활용하여 고속기법을 구현한다. 알고리즘 레벨 최적화를 통하여 검출 및 제거 성능을 떨어뜨리지 않고 PC환경에서 실시간 이상의 연산 속도를 검증하였고, DSP 상에서 최적화하여 720P 영상의 실시간 처리를 확인하였다.

본 논문의 구성은 다음과 같다. Ⅱ절에서는 제안 기법의 알고리즘에 대하여 설명하고 Ⅲ절에서 알고리즘 최적화에 대하여 간략히 설명하고 Ⅳ에서 DSP최적화에 대하여 설명한다. Ⅴ에서는 실험 결과에 대한 분석을 하고 마지막 Ⅵ에서 결론을 맺는다.

Ⅱ. 비 검출 및 제거 알고리즘

본 절에서는 우리가 최적화하고자 하는 알고리즘[7]을 소개한다. 이 기법은 우적의 검출 과정과 제거 과정 등 2단계로 이루어져 있다. 검출 과정에서는 카메라의 움직임에 영향을 받지 않도록 공간적 특성만을 이용한 검출을 수행하고 제거 단계에서는 영상의 시간적 특성을 고려하여 움직임 보상을 이용한 정합을 통하여 우적 제거를 수행한다.

2.1 검출 과정

빗줄기의 검출 과정에서는 비의 밝기 특성과 구조적 특성과 같은 공간적 특성만을 이용한 검출 방법을 제안한다. 또한 가우시안의 분포의 특성을 이용하여 빗줄기 영역과 세기를 측정하여 제거 과정의 성능을 향상시키도록 한다. 빗줄기는 대기에서 주변 빛을 흡수하기 때문에 주변보다 높은 밝기 값을 갖는다. 또한 비의 방향 성분이 수평보다 수직 성분이 크기 때문에 비의 중심이 수평 방향으로 지역적으로 최대 밝기를 갖는다고 가정할 수 있다. 한편 영상의 잡음에 의한 성능의 저하를 방지하기 위해서 전형적인 가우시안 저대역필터 (LPF)를 수행하고 지역적으로 중심 픽셀이 주변 픽셀들보다 큰 값을 가지면서 평균 밝기와 적절한 차이를 갖는 성분들을 초기 비 후보로 하여 비 줄기 지도를 생성한다. 8-연결성을 이용하여 라벨링을 수행한다.

밝기 특성을 이용한 검출 결과에서는 비 이외에도 영상의 디테일 등과 같은 성분들이 포함될 수 있다. 따라서 초기 후보군에서 비가 아닌 아웃라이어를 제거하기 위하여 비의 구조적 특성을 추가적으로 이용한다. 비는 카메라의 노출 시간동안 블러된 직선 형태의 방향성을 가지고 일정 크기 이상의 길이와 서로 유사한 각도를 갖는다. 이러한 특성을 이용하여 비 지도 상에서 일정한 픽셀 개수 미만의 라벨들을 비 후보군에서 제외하고 남아있는 라벨들의 평균 각도를 계산하여 평균 각도와 차이가 큰 라벨들 또한 후보군에서 제외하여 최종적인 비 성분을 검출한다. 일반적으로 영상 내부에서 빗줄기는 비 영역뿐만 아니라 주변도 열화시키는 특성이 있어 빗줄기의 경계가 뚜렷하지 않다. 빗줄기의 세기 분포는

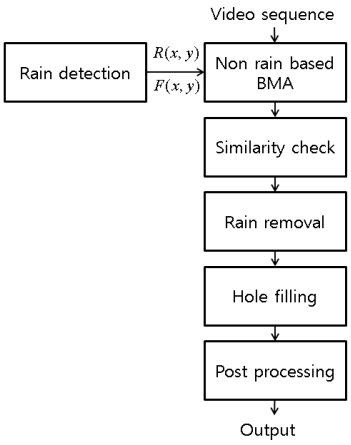


그림 1. 제안하는 빗줄기 제거 기법의 블록도
Fig. 1. The block diagram of the proposed rain removal algorithm.

가우시안의 형태와 비슷하기 때문에 가우시안 분포 모델을 통해 빗줄기 영역의 범위와 세기정보를 갖는 비세기 지도를 생성한다. 최종적으로 검출 과정에서는 빗줄기 지도와 비세기 지도를 추정하여 비 제거 과정에서 영상 복원에 활용한다.

2.2 제거 과정

우리는 시간적 상관성을 고려하여 인접 프레임 간 움직임 보상을 적용함으로써 카메라 움직임에 강인한 우적 제거 기법을 제안한다 (그림 1 참조). 먼저 비 검출 결과를 참조해 인접 프레임 간 비가 아닌 영역 중심의 유사한 블록 후보들을 선정하고 가장 유사한 블록들을 선별한다. NLM필터링을 통해 비 영역을 복원한다. 마지막 후처리를 통하여 동영상 내 깜빡임 효과를 제거하고 가시성을 향상시킨다.

영상에서 비 성분은 시간적, 공간적으로 무질서하기 때문에 비를 포함한 영상 간 움직임 보상을 위해 직접적으로 블록 정합을 적용할 경우 신뢰할만한 결과를 얻을 수 없다[8~9]. 따라서 일정 크기 이상의 세기를 갖는 강한 비 성분만을 취하여 새로운 빗줄기 지도를 검출 과정에서 구한 빗줄기 영역 지도와 세기 지도를 참조하여 재 정의한다. 그런 다음 블록 내부에서 비가 아닌 픽셀들만을 이용해 블록 매칭을 수행하여 유사도가 높은 20개 블록을 선별하고 상대적인 오차 (Sum of Absolute Differences; SAD)를 측정하여 유사도가 낮은 블록들을 후보에서 제외한다. 유사도 측정을 통해 걸러진 신뢰할만한 후보 블록들의 가중치 합을 구하고, 이를 최종적으로 비가 제거된 블록으로 활용한다. 본 논문은 NLM 개념을 응용하여 SAD값들과 빗줄기 지도로부터 블록 내 비 성분이 아닌 영역의 비율을 구한 후 가중치를 계산한다.

검출 과정에서 포함된 아웃라이어나 프레임마다 중복된 위치의 비 성분은 여전히 제거되지 않고 영상 내 홀로 남을 수 있다. 우선 홀 주변의 일정 영역을 포함하여 바운딩 박스를 설정하고 바운딩 박스 내 홀을 제외한 영역의 복잡도를 이용하여 평탄한 영역일 경우 해당 영역에서 홀이 아닌 주변 값들의 평균을 계산하여 홀을 채우고, 복잡한 영역의 홀은 위의 제거 과정을 반복 수행하여 복원한다. 이 과정까지 수행하고도 남아있는 홀들은 오검출된 영역으로 간주하고 해당 위치의 원본 값을 그대로 사용하여 영상을 복원한다.

마지막으로 비가 오는 환경에서는 수분 성분들이 가시성 저하를 일으킨다. 이는 안개와 같은 효과를 가진다. 또한 빗줄기를 제거 후 주변의 잔상으로 인한 깜박임 효과가 있을 수 있다. 따라서 우리는 깜박임 효과를 억제하기 위해 널리 알려진 가이디드 필터 (guided filtering)[10]를 수행하고, 안개 효과를 제거하기 위해 가시성 향상 알고리즘[11]을 적용한다.

Ⅲ. 알고리즘 레벨 최적화

Ⅱ장에서 요약한 제안 기법[7]은 카메라와 물체의 움직임에 영향이 적은 알고리즘이지만, 라벨링 기법의 사용, 움직임 추정 기법 사용 등에 의해 실시간 처리에는 부담이 되는 연산량을 갖고 있다. 제안 기법 연산량의 대부분은 제거 파트에서 발생하지만, 검출 파트 역시 그 자체로 매우 많은 연산량을 가진다고 할 수 있다. 제안 기법의 검출은 초기 검출 지도를 8-연결 라벨링을 이용해 후보로 만들고, 각 후보의 두께, 길이, 각도 등의 정보를 이용해 오 검출 영역을 제거한다. 즉 8-연결 라벨링을 하고, 각 라벨에 대해 두께, 길이, 각도를 계산해야 하는 것이다. 이와 같은 형태의 알고리즘은 최적화에 적합하지 않다. 따라서 우리는 검출 기법의 최적화를 위해 다음과 같은 순서로 알고리즘 레벨 최적화를 수행한다. 먼저 검출 대상이 되는 영상을 세로 방향으로 축소한다. 그런 후 상당한 연산이 필요한 라벨링 과정을 없애고 이를 블록 단위의 이진 패턴화로 대체한다. 또한, 대부분의 연산량을 차지하는 움직임 추정부에 고속

움직임 추정 기법을 적용하였다. 그 외에도 제거 단계에서 가중치 합, 후처리 부분에서 기법의 단순화를 통해 최적화를 수행하였다.

Ⅳ. DSP 레벨 최적화

DSP 상에서의 최적화는 DSP 보드의 기종에 따라서 구현 환경의 특성이 달라진다. 본 논문에서는 영상처리에서 널리 사용되고 있는 DSP 중 하나인 TMS320DM6437 보드에서 알고리즘을 구현하고 최적화를 수행한다. 우리가 DSP 레벨 최적화를 위해 메모리 구조와 지원 하드웨어, CPU구조 등을 이용한다. 그에 따라 각각 메모리 최적화와 EMMA이용, 소프트웨어 파이프 라이닝을 위한 코드 재배치 등을 이용하여 최적화를 수행한다.

4.1 DSP 보드 특징

TI의 TMS320DM6437은 연산을 담당하는 DSP 칩과 L2 단계의 캐쉬, 입력 비디오의 크기 변경이나 자동 초점기 등을 처리하는 비디오 전문 처리 모듈인 VPSS를 하드웨어적으로 제공한다. 또한 외부에 McASP, McBSP, UART와 같은 I/O와 DDR2 외부 메모리를 가지고 있다. 상세 사양은 표 1과 같이 594MHz 동작 속도, 32KB의 L1P, 80KB L1D, 128KB L2, 64ch EDMA 지원 등의 사양을 가진다.

표 1. TMS320DM6437 상세 사양
Table 1. Specification of TMS320DM6437.

항목	사양
CPU	C64x+ 계열
동작속도	594Mhz
On-chip L1	32KB LIP, 80KB, L1D
On-chip L2	128KB L2
Rom	64KB
EMIF	8bit EMIFA, 1 32/16bit DDR2
External memory	Async SRAM, DDR2 SDRAM NAND Flash
DMA	64ch EDMA
Video port	1 dedicated output, 1 dedicate input
Hardware accelerators	Resizer, OSD, Previewer, H3A
EMAC	10/100
HPI	16bit
I/O	MMC/SD, DAC, ATA/CF, ASP, I2C, SPI, UART, VLYNQ, USB
PWM	3
Timer	2 64bit GP, 1 64bit WD

4.2 C 단계 최적화

DSP 구조를 이용하여 최적화를 수행하기에 앞서 알고리즘을 C단계에서 최적화한다. C 단계에서의 최적화는 다음과 같이 진행된다. 먼저 연산에 사용된 변수의 자료형을 실수형에서 정수형으로 바꾸어 준다. 그 다음 look up table을 이용하여 연산과정을 모두 대체할 수 있도록 한다.

실수형 변수 대체

본 논문에서 구현하는 환경인 TMS320DM6437 부동소수점을 이용한 연산모듈이 존재한다. 하지만 정수형 자료형의 연산속도는 실수형 자료형의 연산속도보다 더욱 빠른 속도를 갖는다. 따라서 본 논문에서는 실수형 변수를 모두 없애고 정수형 변수로 변경한다. 해당되는 모든 실수형 변수에 2^N과 같은 큰 값을 곱한 다음 정수형 변수로 형 변환을 수행하여 정수형 변수로 만든다. 연산과정이 끝난 후에는 N만큼 쉬프트를 수행하여 연산결과를 근사화한다. 쉬프트 연산을 사용함으로써 나눗셈 연산보다 더 적은 클록을 소요하여 연산속도를 빠르게 한다.

look up table 응용

통상적으로 대부분의 영상처리 과정에서 반복적인 계산을 통하여 연산을 수행하는 특징이 있다. 따라서 먼저 계산된 연산 결과를 다음 계산에서 활용하거나, 사전에 특정 범위 내에서 계산될 경우의 수를 계산하여 저장한 후 참조 연산을 통해 계산과정을 대체할 수 있다.

그 예로 빗줄기 제거 과정에서 유사도 블록 후보군들 간에 SAD값을 이용해 가중치를 계산하는 부분을 look up table 참조 연산으로 대체하고자 할 때, 블록 매칭 과정에서 계산되는 SAD값은 블록 크기가 MxM인 경우 $0 \sim 255 \times M^2$의 범위를 갖는다. 따라서 이 값에 해당되는 가중치를 사전에 계산할 수 있다. 이 때의 가중

치 look up table은 다음 식(1)와 같은 방법으로 참조할 수 있다.

$$w_k = LUT(SAD_k \gg x) \quad (1)$$

$$LUT(SAD) = e^{-\frac{SAD}{\sigma}} \quad (2)$$

이때 x는 양자화에 사용할 값으로 DSP의 메모리 크기가 제한되어 있음을 고려하여 선택적으로 사용할 수 있는 값이다. x가 커진다면 가중치 값이 정교하게 구해지지 않을 것이지만 메모리 공간을 적게 소모할 것이고, 반대로 x가 작다면 가중치 값이 정교해지지만 메모리 공간을 많이 차지할 것이다. 본 논문에서는 x를 4로 설정하였다.

4.3 Intrinsic 이용 최적화

TMS320DM6437은 CPU 구조에 따른 여러 가지 고유의 intrinsic을 지원한다. 대표적인 지원 intrinsic의 경우 아래 표 2와 같다.

일반적으로 코드를 컴파일할 때 컴파일러가 자동적으로 최적의 intrinsic을 사용하게 된다. 그러나 일보의 경우에는 사용자가 직접 작성할 할 때 더 효율 적인 경우가 있다. 본 논문에서는 가중치 합을 수행하는 부분에서 동일 클록 동안 많은 데이터를 한꺼번에 연산하기 위하여 8bit 데이터 8개를 64bit데이터로 변형하고 변형된 64bit 데이터를 8bit 단위로 덧셈을 수행한 후에 다시 8개의 8bit로 반환하는 과정을 intrinsic을 이용해 구현하였다.

4.4 메모리 구조 및 메모리 최적화

내부 메모리 구조

TMS320DM6437의 경우 240KB의 내부 메모리로 L2 구조의 캐시 메모리가 존재한다. 이를 사용자가 선택함에 따라 캐시 혹은 내부 메모리의 사용 여부를 결정할 수 있다. 또한 EDMA(extended direct memory access)라는 메모리 핸들러를 제공하여 CPU 클럭을 이용하지 않고 외부 메모리에 있는 데이터를 내부 메모리로 옮길 수 있도록 한다. EDMA를 이용하는 경우 외부 메모리에 저장된 영상 데이터를 블록 단위의 형태로 내부 메모리로 불러올 수 있고, 이를 CPU를 이용해 적절히 처리한 후 다시 외부메모리로 보내는 작업을 수행하여 외부 메모리에 의한 메모리 오버헤드를 줄일 수 있다.

메모리 오버헤드 분석

메모리에 저장되어 있는 데이터를 사용하기 위해서는 CPU가 메모리에 있는 데이터를 접근해야 한다. 이 경우 L1 메모리에 있는 데이터는 1클록 사이클로 접근이 가능하고 L2 메모리의 데이터는 2클록 사이클이 필요하다. 연속된 사이클에서 같은 뱅크의 접근의 경우 하나의 stall이 필요하며 외부 메모리인 DDR2 내부의 데이터의 경우 접근하는데 6 클록이 소모된다.

내부 메모리 변수 배치

일반적으로 영상처리 알고리즘은 여러 픽셀을 처리하기 때문에 반복적인 계산이 많다. 즉, 반복적인 계산에 따른 반복적으로 사용되는 변수나 함수와 같은 것들이 존재하게 된다. 이와 같은 부분을 외부 메모리에 배치하는 경우 반복적으로 불러 올 때마다 메모리 오버헤드가 발생하게 되어 연산 시간이 증가한다. 따라서 자주 사용되는 변수나 함수를 내부 메모리에 배치하는 작업이 필요하다. 본 논문에서 자주 사용되는 변수나 함수 선정은 수동적으로 수행하였다. 변수의 경우 반복문

표 2. TMS320DM6437 지원 intrinsic
Table 2. Intrinsics for TMS320DM6437.

Intrinsic	function
int _abs(int src)	절대 값 연산
int _add4(int srt1, int srt2)	입력을 8 bit 단위로 쪼개서 덧셈
Unsigned _clr(unsigned src, unsigned a, unsigned b)	입력을 일정 범위(a, b에 의해 결정)만큼 클리어
Long long &_ammem(void *ptr)	Load or store 8 byte
Unsigned _dotpu4(unsigned src1, unsigned src2)	상위 unsigned 8bit 와 하위 unsigned 8bit 끼리 dot product
Unsigned _mpyu(unsigned src1, unsigned src2)	LSB 16bit 끼리 곱셈
Unsigned _loll(long long src)	64bit 중 하위 32 bit 반환
Unsigned _hill(long long src)	64bit 중 하위 32 bit 반환

에 사용되는 변수나 look up table 과 같은 것들이 주로 쓰였고, 함수는 지역적 평균을 구하는 것과 같은 반복적이며 간단한 것들을 내부 메모리에 배치하였다.

EDMA 사용

영상 처리 알고리즘 최적화 시 메모리에 있는 데이터를 활용할 때 데이터를 접근하는 시간을 줄이는 것이 매우 중요하다. 상 신호는 2D 신호이고, 모든 픽셀에 대해 처리하기 때문에 방대한 횟수의 메모리 접근이 발생하게 된다. 따라서 이 데이터를 접근이 용이한 내부 메모리로 저장하고, 이를 접근해 처리하는 것이 가장 효율적인 방법이다. 그렇기 때문에 영상 처리 전용 모듈에서는 캐시를 거의 사용하지 않고, 모두 내부 메모리 공간으로 취급하는 것이 유리하다. 하지만 대부분의 임베디드 시스템의 경우 내부 메모리가 매우 부족하고 DM6437 또한 마찬가지이다. 실험에 사용한 EVM 보드의 경우 외부 메모리가 128MB를 기본적으로 제공되지만, 내부 메모리의 경우 128KB이기 때문에 영상 하나를 저장하기에는 매우 부족하다. 하지만 외부 메모리의 데이터를 불러와 순차적으로 처리하는 경우에는 매우 느린 수행속도를 갖게 된다. TI 보드에서는 EDMA라는 메모리 지원 툴이 존재하는데 이는 외부 메모리에 있는 데이터를 내부 메모리로 가지고 오는 과정을 CPU의 연산량을 소모하지 않고 독립적으로 처리하여 메모리를 불러오는 연산을 CPU 연산 뒤로 숨길 수 있다. 본 논문에서는 검출 과정의 대부분을 해당 과정을 통하여 연산량 최적화를 수행하였다. 검출 과정의 경우 단일 영상을 이용하여 검출을 수행하기 때문에 대상 주소가 복잡하게 구성되지 않는다. 그러나 제거 과정의 경우는 검출 지도, 현재 영상을 포함한 앞뒤의 인접한 여러 영상이 필요하기 때문에 EDMA를 적용하기에는 적절하지 않다.

영상 전체를 내부 메모리에 저장할 수 없기 때문에 EDMA를 이용해 영상의 일부만을 내부 메모리로 저장해 연산하게 된다. 이런 과정은 그림 2와 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 메모리의 데이터를 내부 메모리로 복사할 수 있다. 또 동시에 내부 메모리의 데이터를 외부 메모리로 복사할 수 있다. 따라서 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 작아야하며, 클수록 더 적은 횟수의 반복 횟수가 필요하기 때문에 내부 메모리에 꼭 배치해야 할 변수나 함수가 차지하고 있는 공간을 제외한 공간만큼의 크기로 결정하는 것이 좋다. 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 있다. 여기서 look up table과 같은 내부 메모리에 배치되어야하는 공간을 제외하고 하나의 분할 영상 버퍼를 32.4KB 만큼의 공간으로 배치하였다. 실험에서 가로가 720개의 픽셀로 이루어진 영상을 이용하고 있고 이 영상의 경우 세로방향으로 10 분할하여 사용하였다.

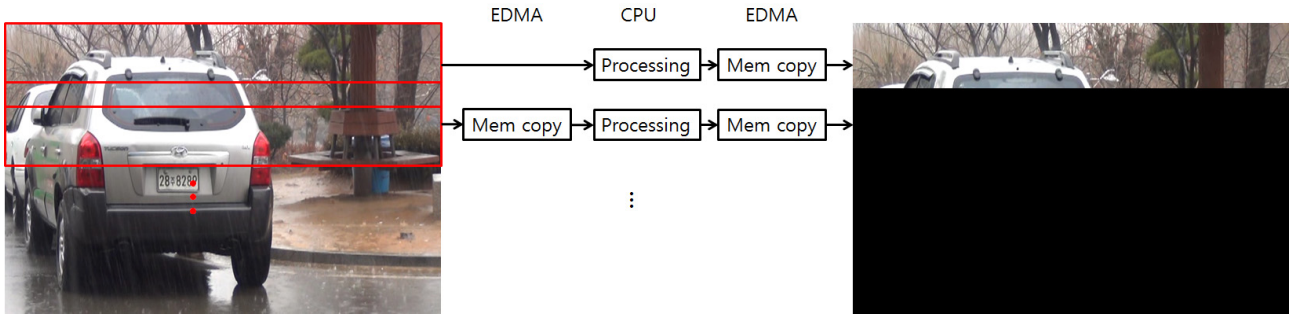


그림 2. EDMA 이용 영상 처리
Fig. 2. Image processing using EDMA.

V. 실험 결과

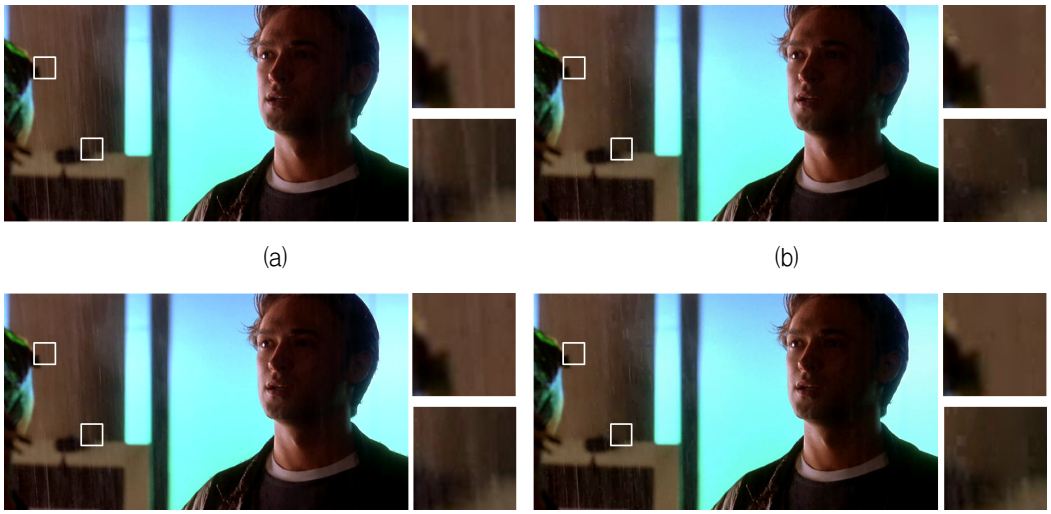
제안하는 기법의 성능과 시간을 검증하기 위해 여러 논문에서 실험 영상으로 사용된 바 있는 “이프온리” 영화 내 비오는 장면과 SONY HDR-CX130으로 직접 촬영한 동영상을 사용하였다. 영화에서 획득한 동영상의 경우 카메라 움직임이 없으면서 인물이나 물체의 움직임이 있지만 크지 않은 동영상으로 720x384@30Hz의 포맷을 가진다. 촬영 동영상은 카메라 움직임이 있는 상황에서 SONY HDR CX 130으로 촬영된 720x480 @30Hz 포맷의 동영상이다.

본 논문은 제안 기법의 성능 평가를 위해 Xue 방법[5], Kim 방법[7]과 비교하였다. 그림 4는 카메라 움직임이 없는 이프온리 영화 내 비오는 장면에서의 비 검출 및 제거 결과를 보여준다. 그림 4(a)에서 보듯이 이 영상은 비의 양이 매우 많은 영상이기 때문에 비가 매우 두껍게 촬영되어 있다. 또한 영상 내 존재하는 사람과 꽃다발에서 약간의 움직임이 있다. 그림 4(b)는 Kim 방법의 제거 결과를 보여준다. Kim 방법의 경우 두께를 고려한 검출 기법이 적용되었기 때문에 얇거나 두꺼운 비 모두 효과적으로 제거해 비 제거 효과가 큼을 볼 수 있다. 하지만 단일 영상만을 이용한 검출 기법의 한계로 기둥과의 경계 부분에서 오 검출이 발생한다. 또 블록 매칭 기반의 기법을 사용하였기 때문에 후보 블록을 잘 못 가져올 수 있고 결국 얇은 에지나 사람의 머리 부분의 영상 열화 현상이 나타난다. 그림 4(c)는 Xue

(a) (b) (c)

그림 3. 실험에 사용된 동영상들
(a) 영화 이프온리 내 비오는 장면, (b), (c) 카메라 움직임이 있는 상황에서 촬영된 영상

Fig. 3. Test videos.
(a) a rainy scene in movie “ifonly”, (b) and (c) images with camera motion



(a) (b)

(c) (d)

그림 4. 영화 이프온리의 비 제거 결과
(a) 입력 영상, (b) Kim 방법 결과, (c) Xue 방법 결과, (d) 제안 기법 결과

Fig. 4. Rain removal results for movie “ifonly”.
(a) Input image, (b) Kim’s method, (c) Xue’s method, (d) proposed method.

기법의 비 제거 결과를 보여준다. Xue 기법은 카메라 움직임이 없는 환경에서 검출 성능이 뛰어나며 비에 해당하는 부분은 대부분 검출하였다. 하지만 폭우 환경에서 나타나는 두꺼운 비 현상에 대해 대응하지 못하고, 그 결과 얇은 비는 제거되었지만, 확대된 영역에서 볼 수 있듯이 영상의 대부분을 차지하는 두꺼운 비들은 그 중심이 제거되었음에도 불구하고 여전히 대부분의 비 성분이 남아있기 때문에 비가 제거되어 있음을 확인하기 어렵다. 그림 4(d)는 본 논문에서 최적화된 기법의 비 제거 결과이다. 제안하는 기법은 Xue 기법과 비교했을 때 영상 상단부에 위치한 굵은 비를 효과적으로 제거함을 확인할 수 있다. 하지만 Kim 방법과 비교했을 때는 영상 하단부의 굵은 비 제거에 실패하는 현상을 보이는데, 예측 벡터 등의 움직임 추정 최적화 과정이 잘 동작하지 않는 영상 경계 부분에서 주로 나타난다.

그림 5는 카메라 움직임이 있는 실제 촬영 영상에서 제거 결과를 보여준다. 그림 5(a)는 촬영된 비가 있는 원본 영상과 비가 있는 영역을 확대한 영상이다. 이 영상은 카메라 움직임이 있으면서, 비가 오는 영역에 번호판과 같은 블러 현상이 눈에 띄기 쉬운 영역을 포함하고 있다. 또한 나무 기둥의 무늬와 같이 세로 방향의 얇은 에지가 존재하기 때문에 오 검출이 일어나기 쉽다. 그림 5(b)는 Kim 방법 제거 결과와 5(a)의 확대한 영역의 제거 결과이다. Kim 방법은 카메라 움직임에 강인한 검출 및 제거 기법인 만큼 영상의 비 영역을 매우 잘 제거하면서 영상 열화가 거의 발생하지 않았다. 하지만 영상의 나무 기둥과 같은 부분의 얇은 에지에서 영상이 일부 블러됨을 확인할 수 있다. 비 영역을 확대한 부분에서는 비를 완전히 제거하였다. 그림 5(c)는 Xue 방법 제거 결과이다. Xue 방법은 카메라 움직임이 있을 때 성능 저하가 심하게 되는 경향이 있다. 결과 영상에서 볼 수 있듯이 비 영역의 검출이 실패함에 따라 비 제거가 전혀 이루어지지 않았다. 확대한 영역에서는 비 제거를 일부 하였으나 여전히 비의 잔상이 남아있다. 그림 5(d)는 제안하는 기법의 제거 결과이다. 제안하는 기법은 비 제거 자체의 성능은 유지하고 있지만, Kim 방법에 비해 지저분한 잔상현상이 남아있는데, 이는 후처리 과정에서 Kim 방법에 비해 연산량을 크게 줄였기 때문에 나타나는 현상이다. 하지만 영상의 열화되는 영역이 없으면서 Xue 기법에 비해 좋은 비 제거 결과를 나타냄을 확인할 수 있다. 확대한 영역에서도 비 제거가 성공적이었지만, 검출 최적화 과정에서 비의 끝부분의 검출을 놓쳐서 약간의 비가 남아있는 문제가 있다.

그림 5. 촬영 영상에 대한 비 제거 결과, (a) 입력 영상, (b) Kim 방법 결과, (c) Xue 방법 결과, (d) 제안 기법 결과

Fig. 5. Rain removal result for a test video, (a) Input image, (b) Kim's method, (c) Xue's method, (d) proposed method.

표 3. PC상 연산 시간
Table 3. Operation time on PC.

Test Video	Xue method	Kim method	Proposed
Video1	95s	180s	0.003s
Video2	120s	230s	0.005s
Video3	119s	234s	0.0053s

표 4. DSP 최적화 후 연산 시간
Table 4. Operation time after DSP optimization.

Test video	DSP opt. X	DSP opt. O
Video 1	0.033s	0.012s
Video 2	0.041s	0.018s
Video 3	0.042s	0.018s

표 3는 제안하는 기법과 비교 기법들의 연산 시간을 PC 최적화 단계에서 측정한 것이다. 연속된 30프레임에 대해 비를 검출하고 제거하는데 소요된 시간의 평균을 나타내었다. 사용된 PC는 3.2GHz로 동작하는 i5-3470 CPU와 16GB RAM의 사양을 가진다. 표의 Xue 방법의 경우 약 100초의 비 검출 및 제거시간이 소요된다. 이는 검출 과정에서 사용한 웨이블릿 변환과 양방향 필터링에 많은 시간이 소요되고, 검출에서 인페인팅 기법을 사용하기 때문이다. Kim 방법의 경우 약 200초의 연산 시간이 소요되었다. 이 기법은 가장 많은 연산량을 가지고 있음을 알 수 있다. 이는 대부분 움직임 추정 과정에서 소요된 시간인데, 비가 아닌 영역만을 이용한 SAD 계산이 C 단계에서 컴파일러가 최적화하기 어려운 구조를 가지고 있기 때문이다. 표에서 보는 것처럼 제안하는 기법의 연산 시간은 비교 기법에 비해 매우 짧다. 제시하는 결과에서 영상에 따라 연산량 변화가 발생하는데, 이는 실험 동영상 1의 경우는 해상도가 720x384이고, 동영상 2, 3은 720x480 영상이기 때문에 해상도 차이만큼 연산량 차이가 생기게 된다.

또한 제안 기법이 영역 적응적인 기법과 예측 기반의 움직임 추정 등을 사용하기 때문에 영상 콘텐츠에 따라 연산량의 변화가 생기기 때문이다. 표 4는 제안하는 기법의 DSP 상에서 연산량 측정 결과이다. DSP 상에서 연산 시간은 연속된 3장의 영상에 대해 기법 동작 부분에서 동작 클록 수를 기록해 CPU 동작 클록으로 나누어 측정하였다. 연산량 비교는 알고리즘 최적화가 된 기법과 DSP 단계의 최적화까지 수행된 결과를 비교하였다. CPU 동작 속도에 비교했을 때 DSP 동작 속도는 산술적으로 약 5배 정도 느리기 때문에 단순하게 비교한다면 DSP 상에서 동작 시간은 5배 정도 느려짐을 기대할 수 있다. 하지만 실제로는 약 8~10배의 연산 증가가 이루어진다. 이는 다양한 이유가 있는데, 먼저 PC에서는 캐시의 용량이 매우 크기 때문에 필요한 데이터를 캐시에 많이 저장할 수 있어서 메모리 오버헤드가 크지 않다. 또한 PC에서는 복잡한 CPU 구조에 기인한 강력한 소프트웨어 파이프라이닝이 가능한 반면, DSP는 단순한 CPU 구조를 가지고, PC 와 동일한 수준의 소프트웨어 파이프라이닝이 불가능하기 때문이다. 본 논문에서 수행한 DSP 단계의 최적화를 통해 약 3배의 연산 시간 측면의 이득을 보았다. 이 결과는 충분히 전처리 과정으로 사용할 수 있을 정도의 연산 시간이다.

VI. 결 론

본 논문은 움직임이 있는 환경에서 촬영된 영상으로부터 빗줄기를 검출 및 제거하는 알고리즘을 알고리즘 및 DSP 단계에서의 최적화를 제안하였다. 알고리즘 레벨에서는 영상의 축소와 라벨링과정의 간소화를 통하여 검출단계의 연산량을 감소하였고 제거단계에서는 움직임 추정 과정을 분석하여 성능에 영향을 미치지 않는 오버랩 영상만을 이용해 제거하였다. 또한 카메라 움직임에 대응할 수 있는 예측 벡터를 사용하여 연산량 감소에 성공하였다. DSP 단계에서는 TMS320DM6437의 특성을 파악하여 메모리 최적화 및 EDMA등을 이용해 메모리 오버헤드를 줄이고 구현된 코드를 재배치하여 컴파일러의 소프트웨어 파이프라이닝이 잘 이루어지도록 하였다. 실험 결과를 통해 제안하는 기법이 비교 기법들과 성능 우위를 유지하면서 DSP상에서 실시간 처리가 가능함을 보였다.

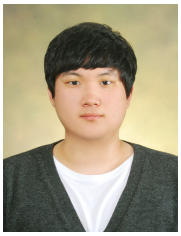
REFERENCES

[1] K. Garg and S. K. Nayar, “Detection and removal of rain from videos,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2004, vol. 1, pp. 528 - 535.
[2] X. Zhang, H. Li, Y. Qi, W. K. Leow, and T. K. Ng, “Rain removal in video by combining temporal and chromatic properties,” in Proc. IEEE Int. Conf. Multimedia Expo., Toronto, ON, Canada, Jul. 2006, pp. 461 - 464.
[3] P. Barnum, T. Kanade, and S. G. Narasimhan, “Spatio-temporal frequency analysis for

removing rain and snow from videos," in Workshop on Photometric Analysis for Computer Vision, in Conjunction with International Conference on Computer Vision, 2007.
[4] P. Barnum, S. Narasimhan, and T. Kanade, "Analysis of rain and snow in frequency space," Int. J. Comput. Vis., vol. 86, no. 2/3, pp. 256 - 274, Jan. 2010.
[5] X. Xue, X. Jin, C. Zhang, and S. Goto, "Motion robust rain detection and removal from videos," in Proc, IEEE Int. Workshop on Multimedia Signal Processing, 2012, pp.170-174.
[6] J. H. Kim, C. Lee, J. Y. Sim and C. S. Kim, "Single-image deraining using an adaptive nonlocal means filter," in Proc. IEEE Conf. Image Process., Sept. 2013, pp.914-917.
[7] 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., Jan. 2015, vol. 26, pp. 317-328.
[8] J. H. Lee, K. W. Lim, B. C. Song, and J. B. Ra, "A fast multi-resolution block matching algorithm and its LSI architecture for low bit-rate video coding," IEEE Trans. Circuits and Systems for Video Technology, vol. 11, no. 12, pp.1289 - 1300, Dec. 2001.
[9] A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," IEEE Trans. Circuits and Systems for Video Technology, vol. 12, no. 10, pp. 934- 941, Oct. 2002.
[10] K. He, J. Sun, and X. Tang, "Guided image filtering," in Proc. European Conf. Computer Vision, pp. 1-14, 2010.
[11] J. P. Tarel and N. Hautiere, "Fast visibility restoration from a single color or gray level image," in Proc. IEEE Int. Conf. Comput. Vis., Kyoto, Japan, 2009, pp. 2201 - 2208.

저 자 소 개

최 동 윤(학생회원)
2014년 인하대학교 전자공학과 학사 졸업.
2014년～현재 인하대학교 전자공학과 석사 과정.
<주관심분야 : 영상처리, 선명도 개선, DSP 최적화>



서 승 지(정회원)
2013년 인하대학교 전자공학과 학사 졸업.
2015년 인하대학교 전자공학과 석사 졸업
2015년 현재 고영테크놀로지
<주관심분야 : 영상처리, 선명도 개선>

송 병 철(평생회원)
1994년 한국과학기술원 전기 및 전자공학과 졸업 (학사).
1996년 한국과학기술원 전기 및 전자공학과 졸업 (석사).
2001년 한국과학기술원 전기 및 전자공학과 졸업 (공학박사).
2001년～2008년 삼성전자 디지털미디어연구소 책임연구원
2008년～현재 인하대학교 전자공학부 교수
<주관심분야: 영상 신호처리, 영상시스템/SoC>