특집논문-02-07-4-04

다양한 블록 크기 기반 선택적 움직임 추정 알고리즘

최 웅 일*, 전 병 우*

A Selective Motion Estimation Algorithm with Variable Block Sizes

Woong Il Choi* and Byeungwoo Jeon*

요 약

기존 비디오 압축 기술에 비해 높은 부호화 효율 및 추가적으로 오류내성, 네트워크 친화성 등의 특성을 지닌 H.264 표준안 부호화 기술 가운데, 복수개 참조 영상을 이용하여 다양한 블록 크기로 움직임 보상하는 기법은 높은 부호화 효율을 갖게 하는 주 요인임과 동시에 높은 복잡도를 갖게 하는 원인으로 작용한다. 이러한 움직임 보상의 복잡도를 줄이기 위해 본 논문에서는 다양한 블록 크기에 대한 선택적 움직임 추정 알고리즘을 제안하고자 한다. 제안된 기법에서는 먼저 다이아몬드 탐색 기법을 이용해 초기 위치 지점에 대해 SAD(Sum of Absolute Difference)를 구하고 이를 기반으로 움직임 추정을 추가적으로 수행할 것인가를 각 블록 크기 단위로 결정하게 된다. 실험 결과, 움직임 탐색 영역을 ±32로 하였을 경우 제안 기법은 기존의 전역 탐색(Full Search)에 비해 약 5배 정도 속도 향상을 보였다.

Abstract

The adaptive coding schemes in H.264 standardization provide a significant coding efficiency and some additional features like error resilience and network friendliness. The variable block size motion compensation using multiple reference frames is one of the key H.264 coding elements to provide main performance gain, but also the main culprit that increases the overall computational complexity. For this reason, this paper proposes a selective motion estimation algorithm based on variable block size for fast motion estimation in H.264. After we find the SAD(Sum of Absolute Difference) at initial points using diamond search, we decide whether to perform additional motion search in each block. Simulation results show that the proposed method is five times faster than the conventional full search in case of search range ±32.

Ⅰ. 서 론

블록 기반 움직임 보상(Motion Compensation) 기법은 비디오 압축 부호화에 있어서 가장 핵심적인 요소이기 때문에 모든 영상 부호화 표준은 블록 변환 부호화 기법과 더불어 움직임 보상 기법을 하이브리드 형태로 연결한 구조를 가진다. 움직임 보상 기법은 블록 단위의 움직임 추정을 통해 시간축 상의 영상 신호간 상관도를 제거함으로써 높은 부호화 효율을 얻는다. 차세대 영상 부호화 표준안인 H.264도 기존 H.261/263 및 MPEG-1/2/4의 표준과 같이 정수변환과 움직임 보상의 하이브리드 형태를 지니고 있으며 특히 움직임 보상의 경우 기존에 비해 효율적으로 개선된 여러 가지 기법들을 통해 종래보다 더욱 높은 부호화 효율을 가지게 되었다[1]. 종래에 H.26L이라는 이름으로 표준화가 진행되었던 H.264 표준안은 2002년 12월 FCD (Final Committee Draft) 단계를 지나 2003년 5월 H.264 국제 표준으로 확정될 예정이다. 본 논문에서는 아직 국제 표준으로 확정되지는 않았지만 편의상 H.26L 대신 H.264로 부르기로 한다. H.264 표준안이 종래의 표준보다 높은

* 성균관대학교 정보통신공학부
School of Information and Communication Engineering, Sungkyunkwan University

표 1. MPEG-4 VM 및 H.264 JM의 R-D(Rate-Distortion) 성능 비교[4]
Table 1. R-D performance comparison between MPEG-4 VM and H.264 JM[4]

sequences	#frames	VM		JM simple		JM complex*	
		Bitrate [Kbps]	PSNRY [dB]	Bitrate [Kbps]	PSNRY [dB]	Bitrate [Kbps]	PSNRY [dB]
Mother & Daughter	100(9*)	82	34.69	46	32.37	55	34.21
Foreman	100(9*)	167	30.91	141	30.90	135	32.24
Calendar & Mobile	100(9*)	195	23.18	243	26.54	244	27.54

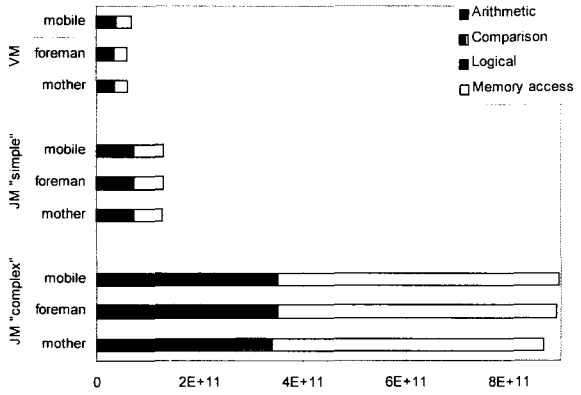
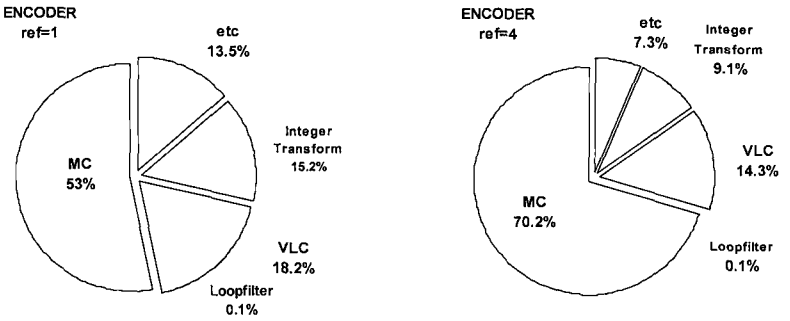


그림 1. MPEG-4 VM 및 H.264 JM의 부호화기 복잡도 분석[4]
Fig. 1. Encoder complexity analysis between MPEG-4 VM and H.264 JM[4]

부호화 효율을 갖는 가장 큰 이유로 16x16단위에서 4x4단위까지의 다양한 블록 크기를 사용하는 움직임 보상과 복수개의 참조 영상을 이용한 기법, 그리고 1/4 화소 단위 정확도의 움직임 벡터를 들 수 있다[1]. 그러나 이러한 H.264 표준안의 움직임 보상 기법들은 기존보다 더 많은 메모리 용량과 연산량을 요구하기 때문에 높은 복잡도로 인하여 실제적인 구현을 위해서는 많은 어려움이 있는 것으로 나타났다. 표 1과 그림 1은 MPEG-4 VM (Verification Model)[2]과 H.264 JM(Joint Model) 2.1[3]의 부호화기에 대한 PSNR 및 비트율, 그리고 연산량에 대하여 비교해 놓은 것이다[4].

이것은 복잡한 움직임을 갖는 "MOBILE AND CALENDAR (MOBILE)" 영상과, 카메라 움직임과 같은 규칙적인 움직임이 많은 "FOREMAN" 영상, 그리고 거의 움직임이 없는 "MOTHER AND DAUGHTER (MOTHER)" 영상의 세가지 영상에 대하여 각각 VM과 JM 단순모드(simple mode) 및 복잡모드(complex mode)들에 대하여 실험한 것이다. H.264 부호화기에 대한 JM 단순모드는 MPEG-4 VM과 비슷한 수준의 복잡도를 갖도록 제한된 몇 가지 부호화 기법들만 이용한 것이고 JM 복잡모드는 H.264가 갖는 모든 부호화 기법들을 적용한 것이다. 이렇게 H.264 부호화기를 2가지 모드로 설정하여 실험한 결과, JM 단순모드는 MPEG-4 VM에 비하여 연산량 측면이나 성능 면에서 비슷한 것으로 나타났으나 JM 복잡모드는 이러한 JM 단순모드에 비해 비트율 면에서 50% 가까이 감소하는 높은 부호화 효율을 보인 반면, 전체 복잡도는 14배 가까이 증가하는 것으로 나타났다. 그림 1에 나타난 것처럼, 부호화 복잡도를 산술(Arithmetic), 비교(Comparison), 논리(Logical), 메모리 접근(Memory access)에 대해 각각 나누어 분석한 결과, JM 복잡모드는 산술 및 메모리 접근을 위해 많은 복잡도를 필요로 하고 있는 것을 볼 수 있다. 이러한 경향은 입력 영상의 움직임과 같은 특성에 관계없이 일정하게 나타났다. 이 실험을 통해 H.264 부호화 기술은 MPEG-4와 비슷한 하드웨어 성능으로는 높은 부호화 효율을 얻을 수가 없으며 H.264 부호화 기술의 개선된 성능을 최대한 발휘하기 위해 기존 MPEG-4보다 수배 이상 빠른 하드웨어를 요구하는 것을 알 수 있다.

H.264가 이러한 높은 복잡도를 갖는 원인을 파악하기 위해 각 부호화 기법이 전체 복잡도에서 차지하는 비중을 분석하였다. 그림 2는 움직임 보상(MC)을 포함하여 정수 변환, 가변장 부호화(VLC) 및 블록킹 현상 제거 필터와 기타 부호화 기법들에 대하여 부호화기 전체 수행 시간에 대한 상대적 수행 시간을 나타낸 것이다. 이 수행시간 분석에 있어 부호화 시 참조영상의 개수에 따른 변화를 보기 위해 1개와 4개의 참조영상을 각각 사용하였을 경우로 나누어 분석하였다. 그 이유는 참조영상의 개수가 전체 복잡도를 좌우하는 가장 큰 요인으로 작용하기 때문이다. 그림 2에서 보는 바와 같이, 하나의 참조 영상을 사용하여 움직임 보상하는 경우, 움직임 보상이 부호화기 전체 수행시간에 차지하는 비중은 절반 정도에 가깝지만 참조 영상의 개수가 증가할수록 움직임 보상이 차지하는 비중이 높아지게 되어 4개의 참조영상을 사용하게 되면 그 비중이 70%까지 증가하게 된다. 이에 따라 상대적으로 VLC와 정수변환 등의 부호화 요소



(a) 참조영상이 1개인 경우 (b) 참조영상이 4개인 경우

그림 2. 각 부호화 기법별 상대적 수행 시간

Fig. 2. Relative execution time spent by each coding tool

들은 각각 18%, 15%에서 14%, 9%로 비중이 줄어들고 있다. 블록킹 현상 제거 필터의 경우에는 0.1%로 극히 미미한 비중을 차지하고 있어서 기존 부호화 기법에 비해 차지하는 비중이 워낙 작지만 이것은 전체 수행시간에서 크게 차이나기 때문에 발생하는 상대적 수치라고 볼 수 있다.

이처럼 H.264에서는 움직임 보상이 기존보다도 복잡도 측면에서 상대적으로 많은 비중을 차지하기 때문에 실제 구현을 위해서는 무엇보다 움직임 보상을 간소화하기 위한 해결책이 가장 급선무이다. 그림 1에서 파악된 복잡도의 주요인인 메모리에서 데이터를 읽어오는 연산과 산술연산은 대부분 움직임 추정을 위해 여러 참조영상의 데이터를 읽어오는 과정과 움직임 벡터를 찾기 위해 SAD(Sum of Absolute Difference) 연산을 하는 과정에 기인한 것이다. 그러므로 본 논문은 이러한 연산 과정을 줄이기 위해 H.264의 다양한 블록 크기와 복수개의 참조영상을 이용한 움직임 보상 과정을 간소화 하는, 기존 고속 움직임 추정 알고리즘을 변형시킨 형태의 고속 알고리즘을 제안한다. 이를 위해 먼저 현재 H.264의 움직임 보상 기법에 대해 간략하게 소개하고 이러한 움직임 보상을 간소화하기 위해 기존의 고속 움직임 추정 기법인 다이아몬드 탐색(DS) 기법에 대해 알아본다[5]-[7]. 그리고 이러한 DS 탐색 알고리즘을 기반으로 H.264의 움직임 추정을 간소화하기 위한 새로운 알고리즘을 제안한다.

Ⅱ. H.264에서의 움직임 보상 기법

H.264 표준안의 움직임 보상도 역시 기존 비디오 부호화 기법과 마찬가지로 블록 매칭 기법 기반으로 되어 있다. 기존 H.263++ 및 MPEG-4에서는 16x16 또는 8x8 블록 단위의 블록 매칭 기법을 사용하였는데 이러한 두 가지 모드는 움직임의 특성이나 비트율 등에 따라 성능 차이가 달라지는 것으로 알려져 있다[8]. 예를 들어, FOREMAN 영상이나 MOBILE 영상과 같이 움직임이 빠르고 불규칙적인 영상의 경우에는 8x8 블록 단위로 부호화 하는 것이 유리할 때가 많고 CONTAINER와 같이 움직임이 적고 규칙적인 영상의 경우에는 16x16 블록 단위로 부호화하는 것이 유리하게 작용한다. 또한 비트율이 낮아질수록 8x8보다 16x16 블록 단위로 부호화하는 것이 성능 면에서 유리하게 작용한다. 이러한 차이는 블록단위를 작게 나눌수록 실제 움직임에 가까운, 더욱 정확한 움직임 벡터를 찾을 수 있지만 이러한 움직임 벡터를 부호화하는데 필요한 비트량이 증가하여 그만큼의 오버헤드를 요구하기 때문에 발생하는 것이다. 이것은 객체기반의 움직임 보상이라는 관점에서도 해석될 수 있는데 블록 크기를 작게 할수록 각 객체의 움직임을 정확히 표현할 수 있기 때문에 다양한 객체를 포함하는 영상일수록 작은 블록 크기로 움직임 보상하는 것이 유리하다.

H.264에서는 이러한 기존 블록단위의 움직임 보상을 좀 더 개선하여 그림 3에 나타난 것처럼 16x16, 16x8, 8x16, 8x8하위 모드등의 4가지 블록 형태로 움직임 보상을 하며 8x8하위 모드의 경우에는 다시 8x8 블록 단위에 대해 각각 8x8, 8x4, 4x8, 4x4 블록의 4가지 형태로 움직임 보상할 수 있도록 하였다. 그러므로 종래보다 훨씬 더 다양한 형태의 블록 단위 움직임 보상이 가능해졌으며 특히 8x8 이하의 움직임 보상을 통하여 다양하고 복잡한 객체의 움직임을 충분히 표현할 수 있게 되었다. 또한 4x4 단위의 움직임 보상은 부호화 효율 면에서는 큰 효과가 없지만 주관적 화질을 개

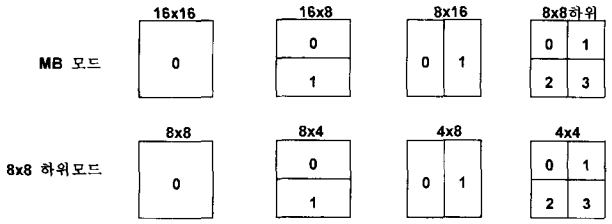


그림 3. 다양한 블록 단위 움직임 보상 모드
Fig. 3. Motion compensation mode with variable block size

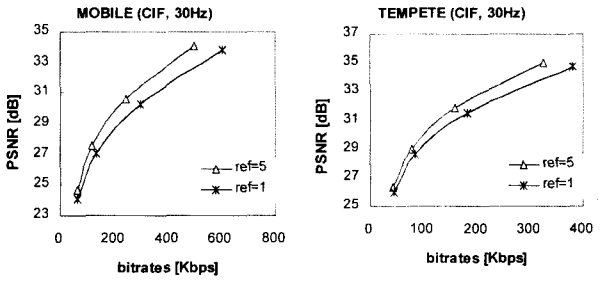


그림 4. 복수개의 참조 영상의 R-D 성능
Fig. 4. R-D performance of multiple reference frames

선하는 것으로 나타났다. 각 블록의 전송 순서는 그림 3에 나타난 것과 같다.

이러한 다양한 블록 형태에 대한 움직임 보상을 위해 참조하는 영상은 프레임 버퍼에 저장되어 있는 어떤 프레임도 가능하다. 종래에는 P 픽쳐의 경우 시간적으로 과거의 영상으로부터 반드시 움직임 보상하도록 하였으며 B 픽쳐의 경우에는 시간적으로 과거와 미래 2개의 픽쳐를 통해서 움직임 보상할 수 있었다. 그러나 H.264에서는 이러한 제약이 존재하지 않기 때문에 각 블록별로 선택적으로 다수의 참조 영상을 이용하여 움직임 보상할 수 있다. 이를 위해 16x16, 16x8, 8x16, 8x8하위모드 별로 어떠한 영상을 참조하였는지 그 인덱스를 전송하며, 8x8하위모드 이하의 블록 크기의 경우에는 동일한 참조영상을 사용한다. 일반적으로 사용할 수 있는 참조 영상의 개수는 H.264 표준안에 지정된 레벨에 따라 제한되어 있으며 일반적으로 레벨이 높을수록 참조할 수 있는 영상의 수도 증가한다. B 슬라이스는 종래와 달리 부호화 효율을 높이기 위해 참조영상으로 사용될 수 있으며 또한 2개의 참조영상은 시간적인 제약을 받지 않아 시간적으로 어떤 위치에도 있을 수 있다. 즉 H.264의 B 슬라이스는 2개의 과거 프레임이나 2개의 미래 프레임으로부터, 또는 과거와 미래의 프레임 각각으로부터 아무런 제약없이 예측될 수 있는 특징을 갖는다. 이러한 복수개의 참조 영상을 이용한 경우 얻게 되는 부호화 효율을 알아보기 위해 P 슬라이스만을 이용하여 MOBILE 영상과 TEMPETE 영상에 대하여 각각 1개와 5개의 참조영상을 사용하여 부호화한 결과가 그림 4에 나타나 있다.

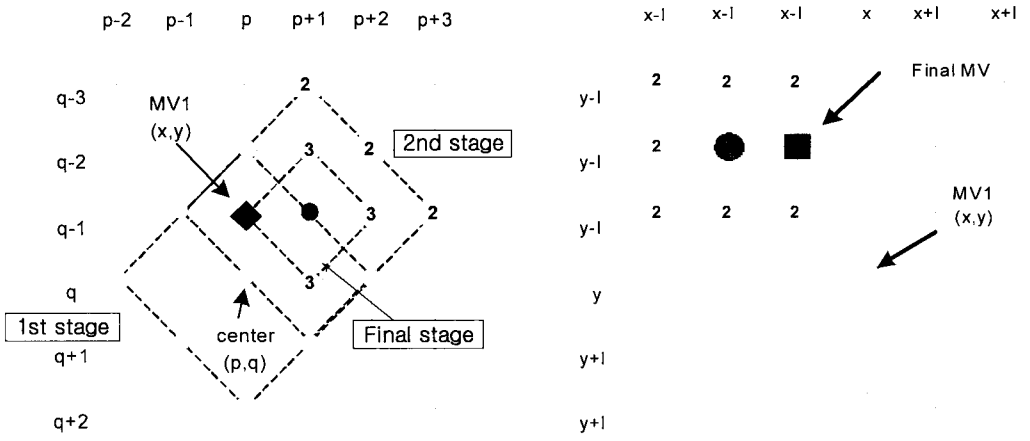
그림 4의 R-D 그래프를 보면 참조 영상을 5개까지 사용하게 되면 최대 1dB이상 PSNR이 증가하는 것을 볼 수 있으며 고비트율로 갈수록 부호화 효율도 증가하는 것을 파악할 수 있다. 복수개의 참조 영상을 이용한 경우, 특히 높은 부호화 효율을 가져오는 영상은 반복적인 움직임이 있는 특징이 있었다. 다양한 객체들의 복잡한 움직임을 담고 있는 MOBILE 영상의 경우에는 주로 상하 및 좌우의 주기 운동과 회전운동 등이 존재하기 때문에 복수개의 과거 프레임을 이용함으로써 부호화 효율을 높일 수 있다. 또한 움직임이 비교적 적은 TEMPETE 영상의 경우에도 반복적으로 떨어지는 낙엽과 같은 상하방향의 반복운동으로 인해 동일한 효과를 얻고 있다.

마지막으로 H.264 표준안이 갖는 움직임 보상의 특징으로 1/4 화소 단위의 움직임 예측을 들 수 있는데 이것은 좀 더 정확한 움직임 벡터를 찾기 위해 영상을 4배로 보간하여 움직임을 예측하는 기법이다. 4배 보간을 위해 bilinear 필터와 {1, -5, 20, 20, -5, 1}/32의 6-tap FIR 필터를 이용한다. 먼저 1/2 화소 위치를 6-tap FIR 필터로 보간한 다음, 나머지 1/4 화소 위치는 bilinear 필터를 통해 얻는다. 이러한 필터링은 많은 곱셈 및 나누셈의 산술 연산이 필요하기 때문에 복잡도를 크게 증가시키는 요인이 될 수 있지만 정확한 움직임 벡터로 인한 부호화 효율이 그만큼 크기 때문에 종래 MPEG-4 ASP(Advanced Simple Profile)도 이와 동일한 기술을 포함하고 있다.

이처럼 H.264 표준안의 움직임 보상은 전체 복잡도에 주요 비중을 차지하면서 종래에 비해 수 배 이상의 많은 연산량을 요구하지만 H.264 표준안이 나타내는 부호화 효율에 있어서 또한 가장 핵심적인 역할을 담당하고 있기 때문에 이러한 움직임 보상의 사용은 불가피하다. 따라서 종래보다 훨씬 다양하고 복잡해진 움직임 보상을 간략화하기 위한 선택적 움직임 추정 기법을 제안하고자 한다.

Ⅲ. 다양한 크기의 움직임 보상 블록에 따른 선택적 움직임 추정 기법

Diamond Search(DS) 기법은 종래의 고속 움직임 추정 알고리즘으로 연구된 바가 있으며 이것을 기반으로 또한



(a) 정수화소 단위 움직임 추정 기법 (b) 1/4 화소 단위 움직임 추정기법

그림 5. 다이아몬드 탐색에 따른 움직임 추정 기법의 예

Fig. 5. Example of motion estimation using diamond search

많은 개선된 알고리즘들이 개발되어 왔다[5]-[7]. DS기법은 탐색의 중심 화소와 그 중심으로부터 시티블록 거리가 2의 화소들로 이루어진 LDSP(Large Diamond Search Pattern)와, 중심 탐색화소와 그로부터 시티블록 거리가 1의 화소들로 이루어진 SDSP(Small Diamond Search Parttern)의 두 가지 형태의 탐색 패턴을 가지고 있다. LDSP는 중심을 포함하여 9개의 탐색 위치점을, SDSP는 중심을 포함하여 5개의 탐색 위치점을 갖고 있으며 둘 다 모두 다이아몬드 모양을 갖는다. 이러한 DS 기법에 대한 예를 그림 5(a)를 통해 살펴보도록 한다.

DS기법은 먼저 그림 5(a)에서 탐색의 중심점 (p,q)를 시작으로 LDSP 형태로 정수위치의 9가지 탐색 위치점에서 SAD를 비교하여 최소 SAD를 갖는 점을 찾게 되는데 이것을 첫번째 단계(stage)라고 하자. 그리고 그 최소 SAD를 갖는 점을 중심으로 다시 첫번째와 동일한 방법으로 두번째 단계를 진행하게 되는데 이러한 탐색은 최소 SAD가 9개 화소의 중심 지점에 발생할 때까지 반복한다. 만일 중심 지점에서 최소 SAD를 찾게 되면 그 지점을 중심으로 동, 서, 남, 북 4개의 지점인 SDSP에 대해 최소 SAD를 갖는 지점이 움직임 벡터가 된다. 그림 5(a)와 같이 두번째 단계에서 중심점((p+1,q-1)위치)이 최소 SAD가 된 경우, 마지막 단계로 SDSP 내에서 최종 움직임 벡터 MV1(x,y)을 찾게 된다((p,q-1)위치). 본 논문에서는 종래의 DS 기법에서 탐색의 첫 시작점을 탐색 영역의 중심인 (0,0)지점으로 하였던 것과 달리, 움직임 벡터 부호화시 사용되는 중간값(median)에 의해 구한 예측 벡터를 중심점으로 하였다. 이것은 그 예측 벡터와 차분 부호화하여 현 움직임 벡터를 전송할 때 사용하는 VLC 테이블을 보면 알 수 있듯이 현재 벡터가 예측 벡터와 동일할 확률이 높아서 시작점으로 예측 벡터를 이용하는 것이 효과적이기 때문이다[6][7]. 이러한 기존 DS의 경우, 전역 탐색(Full Search)과 비교하여 탐색 영역을 ±16으로 한 경우, SAD 비교 횟수가 대략 1/60 정도로 줄어들게 되며 PSNR은 약 0.2dB 정도 감소한다[5].

그림 5(a)와 같이 화소 단위 움직임 추정이 끝나면 그림 5(b)처럼 1/4 화소 단위의 움직임 추정을 수행한다. 그림 5(a)에서 찾은 정수화소 단위의 움직임 벡터 MV1이 가리키는 위치와 그 주변 8개의 위치에 대해 1/2 화소 단위 움직임 벡터를 찾고, 다시 그 위치를 중심으로 주변 8개의 위치에 대해 1/4 화소 단위 움직임 벡터를 찾아 최종 움직임 벡터를 추정하게 된다. 이러한 1/4 화소 단위 움직임 추정 기법은 현재 H.264 표준안의 참조 부호화기인 JM(Joint Model)에서 사용되고 있는 기법이다.

그러나 이러한 DS 알고리즘을 H.264 움직임 보상에 그대로 적용할 경우, 기존 MPEG-4 또는 H.263++에서와 동일한 성능을 얻을 수 없다. MPEG-4 및 H.263++에서는 움직임 보상 단위가 16x16 또는 8x8 블록으로 2가지 모드 뿐이지만 H.264에서는 7가지 형태의 움직임 보상 블록 단위가 존재하며 또한 복수개의 참조 영상을 이용하기 때문이다. 예를 들어 RxR 탐색 영역에 대해 16x16 매크로블록의

움직임 추정을 고려하면, 필요한 SAD 연산 수가 $256R^2$ 번이 되는데 이것을 S_0라고 하자. H.263 베이스라인의 경우에 하나의 매크로블록에 대한 움직임 추정시 S_0 만큼의 SAD 연산을 요구하게 된다. 반면, H.264의 경우에는 움직임 보상 블록 단위가 16x16에서 8x8 하위모드까지 4가지이며, 8x8 하위모드는 다시 8x8, 8x4, 4x8, 4x4 블록단위로 나눠진다. 따라서, $M\times N$ 크기의 블록에 대한 움직임 추정시 필요한 SAD 연산수를 $S_{M\times N}$이라고 할 때, 한 매크로 블록의 움직임 벡터를 모두 결정하기 위해 필요한 H.264에서의 SAD 총 연산수는 다음과 같다.

$$\begin{aligned} & S_{16\times16}+2S_{16\times8}+2S_{8\times16}+4\{S_{8\times8}+2S_{8\times4}+2S_{4\times8}+4S_{4\times4}\} \\ &= S_0+2\frac{S_0}{2}+2\frac{S_0}{2}+4\left\{\frac{S_0}{4}+2\frac{S_0}{8}+2\frac{S_0}{8}+4\frac{S_0}{16}\right\} \\ &= 7S_0 \end{aligned} \quad (1)$$

또한, 하나의 참조영상을 사용하는 경우 기존 16x16 매크로 블록의 움직임 추정에 비해 SAD 연산수가 7배 정도 증가하지만, 만일 복수개의 참조영상을 이용할 경우 그만큼 배수로 증가하게 된다. 이러한 SAD 연산의 증가는 1/4 화소 단위 움직임 추정의 경우에도 마찬가지이다.

그림 2의 부호화기 복잡도 실험에서 움직임 보상의 각 부호화 기법별로 차지하는 비중을 분석해본 결과, 4개의 참조영상을 사용한 경우 정수 단위 움직임 추정이 차지하는 비중이 43%이고 1/4 화소 단위의 움직임 추정이 40%를 차지하였으며 이외 나머지는 1/4 화소 단위 움직임 보상을 위한 보간이 차지하였다. 그리고 1개의 참조영상을 이용한 경우에는 각각 30%, 36%로 나타난 것을 볼 때, 1/4 화소 움직임 추정도 전역 탐색을 이용한 정수 화소 단위 움직임 추정만큼이나 많은 연산을 요구하는 것으로 나타났다. 또한 4x4 단위까지의 작은 블록 크기의 움직임 보상은 잦은 메모리 접근을 야기시켜 종래보다 메모리 대역폭을 더욱 증가시키게 되어 하드웨어 구현에 치명적인 단점으로 작용하고 있다. 그러므로 종래의 고속 움직임 추정 알고리즘을 그대로 적용하는 것보다 이러한 다양한 경우에 대해 효과적으로 대처할 수 있는 보다 더 근본적인 해결책이 필요하다.

본 논문에서는 이러한 움직임 추정을 간소화하기 위한 해결책으로 4x4 블록들에 대해 계산된 SAD값들을 재활용할 수 있도록 기존 다이아몬드 탐색법을 개선하며, 또한 블록 크기별로 최적의 부호화 모드를 미리 예측하여 차별적으로 움직임 추정을 하는 알고리즘을 제안하고자 한다.

본 제안 기법의 순서도가 그림 6에 나타나 있다. 먼저 4x4 블록 단위로 다이아몬드 탐색의 첫번째 단계(stage)에 대해 움직임 벡터를 추정한 다음(과정 1), 이를 이용하여 첫 단계의 각 블록 크기별 움직임 탐색을 수행한다(과정 2). 그리고 이 가운데 최적 움직임 벡터를 선정한 후(과정 3,4) 이를 기반으로 다이아몬드 탐색의 두번째 단계를 선택적으로 수행한다. 이를 위해 먼저 첫 단계의 최적 움직임 벡터인 경우에 한해 나머지 단계를 진행하고 그 외의 움직임 벡터들에 대해서는 곧바로 마지막 단계인 SDSP 탐색으로 진행한다(과정 5). 그리고 첫 단계에서의 최적 움직임 벡터는 1/4 화소단위 움직임 탐색을 수행하지만 그 외의 움직임 벡터들은 각 움직임 벡터의 R-D 비용(J_C)값이 최적 움직임 벡터의 것($J_{best.stage1}$)과 비교하여 그 차이가 임계치보다 작은 경우에만 1/4 화소단위 움직임 벡터를 수행하도록 한다(과정 6). 이렇게 각 블록 크기별, 참조영상별로 모든 움직임 벡터를 찾게 되면 R-D 최적화 기법을 이용하여 최적의 부호화 모드를 결정하게 된다. 이러한 제안 기법에 대해 각 단계별로 살펴보면 다음과 같다.

과정 1) 먼저 4x4 각 블록들에 대해 다이아몬드 탐색의 첫번째 단계를 수행한다. 여기서 9개 지점의 중심점은 16x16

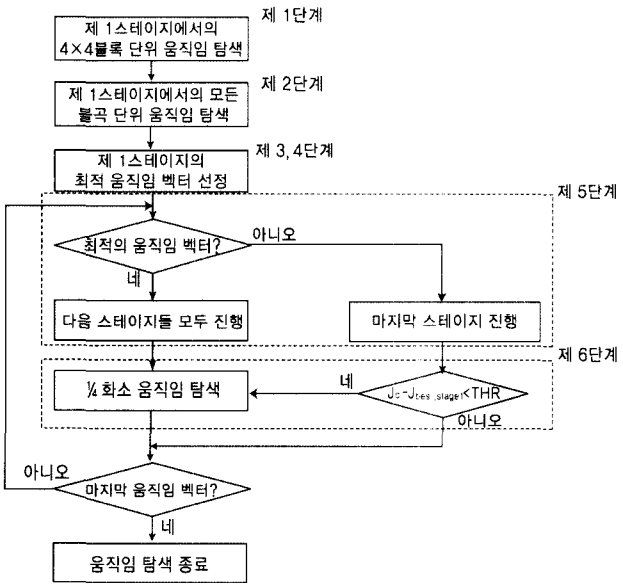


그림 6. 제안 기법의 순서도
Fig. 6. The flowchart of the proposed method

블록 움직임 벡터의 예측 벡터로 하며, 각 지점에 대해 R-D 관점에서 최적의 지점을 찾는다. 현재 H.264 JM(Joint Model)에서 사용되고 있는 R-D 최적화 기법[9]은 움직임 추정시 단순히 SAD만 비교하지 않고, 다음 식 (2)를 통해 비트율과 화질열화를 모두 고려한다.

$$J(m,\lambda_M) = SAD_{4\times4}(B_C, B_{REF}(m)) + \lambda_M \cdot R(m-pmv_{16\times16}) \quad (2)$$

여기서 B_C는 현재 블록을, 그리고 m은 움직임 벡터를, 그리고 $pmv_{16\times16}$는 16x16 블록의 예측 벡터를 가리킨다. λ_M은 Lagrangian 계수로써 $\sqrt{0.85 \cdot 2^{QP/3}}$의 값을 갖으며, $R(m-p)$는 움직임 벡터와 예측 벡터와의 DPCM을 통해 부호화해야 할 움직임 벡터의 최종 비트량이다[3]. 또한 $SAD_{4\times4}(B_C, B_{REF}(m))$는 주어진 4x4 블록 B_C와 참조 영상에서 움직임 벡터 위치의 블록 $B_{REF}(m)$간 SAD값으로 다음 수식 (3)을 통해 계산된다.

$$SAD_{4\times4}(B_C, B_{REF}(m)) = \sum_{y=1}^{4}\sum_{x=1}^{4}\left\{\left|B_C(x,y) - B_{REF}(x-m_x, y-m_y)\right|\right\} \quad (3)$$

그러므로 수식 (3) 에서 구한 SAD값과 움직임 벡터의 비트수를 통해 얻어진 식 (2)의 $J(m,\lambda_M)$를 최소화하는 위치를 찾음으로써 R-D 관점에서 최적의 움직임 벡터를 얻게 된다.

과정 2) 과정 1에서 구한 각 4x4 블록의 SAD값을 이용하여 모든 블록 단위에 대하여 LDSP 형태의 움직임 벡터를 추정한다. 즉, (i,j)를 한 매크로블록 내의 4x4 블록의 좌표위치라고 하면 MxN 블록에 대한 SAD는 수식 (4)와 같이 구해진다.

$$SAD_{M\times N}^{(i,j)}(B_C, B_{REF}(m)) = \sum_{q=j}^{j+N/4}\sum_{p=i}^{i+M/4}\left\{SAD_{4\times4}^{(p,q)}(B_C, B_{REF}(m))\right\} \quad (4)$$

그러므로 수식 (4)를 적용하여 모든 블록 단위에 대해 추가로 움직임 추정 필요 없이 4x4 단위 움직임 추정시 구한 4x4 단위 SAD 값을 활용하여 나머지 블록 단위의 SAD값을 모두 구하고, 다시 (2)의 수식을 통해 각 블록 단위마다 움직임 벡터를 구한다. 이러한 1,2 과정을 참조 영상의 개수만큼 반복함으로써 각 참조영상 및 블록 크기에 대한 다이아몬드 탐색 첫 단계의 움직임 벡터를 모두 찾게 된다.

과정 3) 이렇게 찾은 모든 움직임 벡터들 가운데 최적 움직임 벡터를 찾기 위해 먼저 8x8 하위모드에서의 각 8x8 블록별로 최적 움직임 벡터를 찾는다. 각 8x8 블록내에서는 동일한 참조 영상을 사용하기 때문에 8x8, 8x4, 4x8, 4x4 각 블록별로 동일한 참조 영상을 사용한 경우 R-D 관점에서 최적의 움직임 벡터를 식 (5)를 통해 찾을 수 있다.

$$J_{best8\times8,stage1}(\{\hat{m}\},\lambda_M) = \min_{\substack{M\times N,\\ ref\_frame}}\left\{\sum_{p=1,q=1}^{8/M,8/N} J_{M\times N,stage1}^{(p,q,ref\_frame)}(m^{(p,q)},\lambda_M)\right\} \quad (5)$$
$$(M, N \le 8)$$

과정 4) 다이아몬드 탐색의 첫번째 단계에서의 최적 움직임 벡터를 수식 (6)을 통해 찾는다.

$$J_{best,stage1}(\{\hat{m}\},\lambda_M) = \min_{M\times N}\left\{\sum_{i=1,j=1}^{16/M,16/N}\min_{ref(i,j)}\left\{J_{M\times N,stage1}^{(i,j,ref(i,j))}(m^{(i,j)},\lambda_M)\right\}\right\} \quad (6)$$

각 (i,j)위치 움직임 벡터의 R-D 비용($J_{M\times N,stage1}^{(i,j,ref(i,j))}$)을 최소로 하는 참조 영상을 찾은 다음, 이러한 참조 영상을 갖는 각 움직임 벡터의 R-D 비용을 각 블록 크기별(16x16, 16x8, 8x16, 8x8)로 합산하여 이 가운데 최소값을 찾아 $J_{best,stage1}(\{\hat{m}\},\lambda_M)$로 한다. 여기서 8x8 하위모드의 경우에는 과정 3에서 구한 $J_{best8\times8,stage1}$들을 합산하여 8x8단위 R-D 비용을 구한다.

한 예로 그림 7과 같이 각 블록 단위별로 각 블록 위치의 최적 참조 영상을 갖는 움직임 벡터를 찾았다면, 이들 움직임 벡터들을 각 블록 크기별로 합산한 값들($J_{16\times16}$, $\sum J_{16\times8}$, $\sum J_{8\times16}$, $\sum J_{best\ 8\times8,stage1}$)중 최소값을 갖는 것이 최적의 움직임 벡터가 된다.

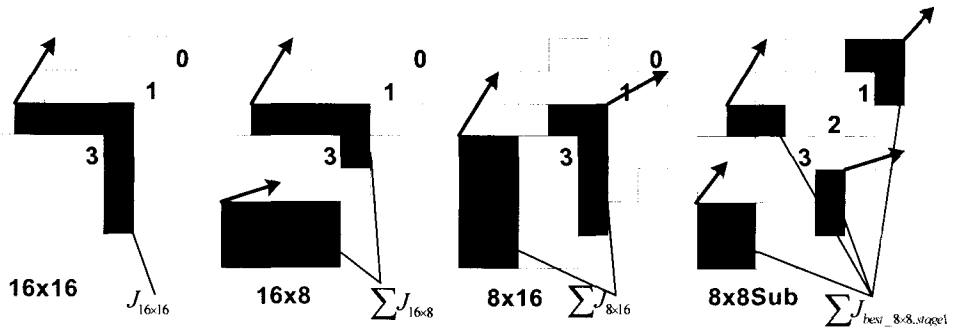


그림 7. 16x16에서 8x8 크기까지의 각 블록 단위 움직임 추정
Fig. 7. Motion Estimation with block size from 16x16 to 8x8

과정 5) 과정 4까지 다이아몬드 탐색의 첫번째 단계가 모두 끝나면 그 다음 단계는 선택적으로 수행한게 된다. 다이아몬드의 두번째 단계부터는 과정 4에서 결정된 최적 움직임 벡터의 경우에만 수행하고 나머지 움직임 벡터의 경우에는 바로 다이아몬드의 최종 단계를 수행하여 정수화소 단위 움직임 벡터를 결정한다. 즉, 최적 움직임 벡터인 경우에는 일반적인 다이아몬드 탐색을 그대로 적용하고 그외 경우에는 간략화하여 2개의 단계(LDSP+SDSP)만으로 된 다이아몬드 탐색만을 수행하게 되는 것이다.

과정 6) 정수화소 단위 움직임 벡터 수행이 끝나면 이러한 움직임 벡터를 중심으로 1/4 화소단위 움직임 추정을 추가로 하게 된다. 단, 여기서 추가적으로 수행되는 1/4 화소단위 움직임 추정은 각 블록의 R-D 비용 $J_{M\times N}^{(i,j)}(m^{(i,j)},\lambda_M)$이 부등식 (7)의 조건을 만족하는 경우에 한한다.

$$J_{M\times N}^{(i,j)}(m^{(i,j)},\lambda_M) - J_{best.stage1}(\hat{m},\lambda_M)\cdot\frac{M\times N}{16\times16} < THR \tag{7}$$

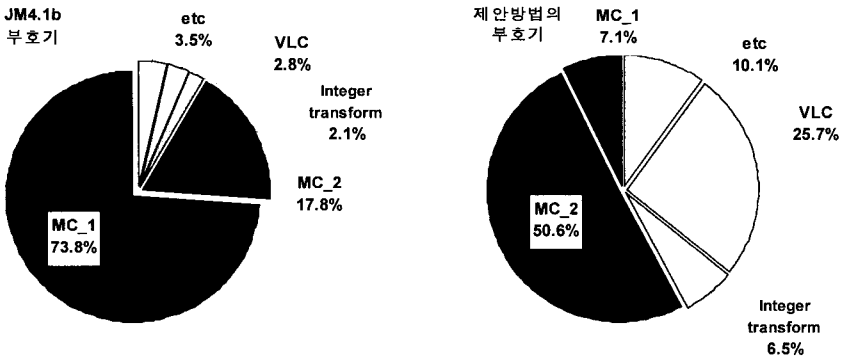
그러므로 MxN 블록단위에서 각 움직임 벡터 $m^{(i,j)}$의 R-D 비용이 앞의 과정 4에서 구한 R-D 비용보다 어떤 일정값 이상 크게 되면 그 블록의 움직임 벡터 $m^{(i,j)}$에 대해서는 1/4 화소 움직임 탐색 과정을 생략하게 된다. 여기서 참고로 R-D 비용 $J_{best.stage1}(\{\hat{m}\},\lambda_M)$은 매크로블록 단위의 R-D 비용이기 때문에 블록 크기에 따라 이 값을 보정해 주도록 하며, 두 값의 차이에 해당하는 임계치 *THR*은 실험적으로 얻은 수치를 사용한다.

Ⅳ. 실험결과

제안된 DS 기반의 새로운 움직임 추정 알고리즘의 성능을 비교하기 위한 실험 조건이 표 2에 나타나 있다. 표 2의 실험 조건은 현재 H.264 표준화 과정에서 사용되고 있는 실험 조건에 기반한 것이다. 본 실험에서는 JM 4.1b 부호화기를 이용하였고 IPPP 구조에 대해서만 실험하였다. 엔트로피 부호화는 CAVLC를 이용하였으며 최적화된 부호화기를 위해 R-D 최적화 및 Hadamard 변환을 이용한 모드 결정법을 이용하도록 하였다. 그리고 선택적 움직임 추정을 위해 사용된 임계치(*THR*) 값은 실험적으로 얻은

표 2. 실험 조건
Table 2. Test Condition

	NEWS	CONTAINER	FOREMAN	SILENT	PARIS	MOBILE	TEMPETE
영상크기	QCIF	QCIF	QCIF	QCIF	CIF	CIF	CIF
Skip	2	2	2	1	1	0	0
프레임율(Hz)	10	10	10	15	15	30	30
총프레임수	100	100	100	150	150	300	300
QP	28, 32, 36, 40						
기타	R-D 최적화, Hadamard 변환 사용,						



(a) JM4.1b 부호화기의 상대적 비중 (b) 제안 방법을 이용한 부호화기의 상대적 비중

그림 8. 제안 방법과 기존의 각 부호화 기법별 수행 시간 비교

Fig. 8. Comparison of the processing time between JM4.1b and proposed method

192*(QP-12)를 이용하였다.

그림 8은 제안된 움직임 추정 기법을 이용한 경우 부호화기의 복잡도를 기존과 비교하여, 전체 수행 시간 가운데 비교적 많은 수행시간을 요구하는 각 부호화 기법별로 상대적 비중을 나타낸 것이다. 그림 8에서 MC_1은 정수 단위 움직임 탐색이고 MC_2는 1/4 화소 단위 움직임 탐색을 의미한다. MC_1의 경우, 기존 부호화기에서는 전역 탐색 알고리즘을 사용하고 있으나 제안 방법은 DS 기법을 기반으로 하고 있다. 움직임 탐색 영역을 32로 한 경우 기존 정수 단위 움직임 탐색인 MC_1은 73.8%로써 가장 많은 부분을 차지하였으나 제안 방법에서는 전체 부호화기에 7.1%밖에 차지하지 않는다. 반면 MC_2의 경우에는 17.8%에서 50.6%로 상대적으로 증가하여 가장 많은 비중을 차지하는 것으로 나타났으나 MC_1이 줄어듦에 따라 나머지 경우도 상대적으로 늘어난 것을 볼 수 있다. VLC 및 정수변환은 매크로블록의 모드를 결정하기 위한 R-D 최적화에서 실제 발생 비트율을 계산하기 위해 필요하기 때문에 R-D 최적화를 사용하게 되면 비교적 많은 비중을 차지하게 된다. 특히 VLC의 경우에는 내용기반 적응적 부호화 기법(CAVLC)으로 인해 상대적으로 그 비중이 크게 증가하였다. 제안 기법을 적용하더라도 여전히 1/4 화소 단위 부호화가 절반 정도의 많은 비중을 차지하고 있어 이를 좀더 효과적으로 부호화하기 위한 기법이 필요함을 볼 수 있다.

표 3은 제안 기법을 이용한 부호화기와 기존 전역 탐색 기반의 JM4.1b 부호화기와의 성능을 비교한 것이다. 기본적으로 움직임 백터의 탐색 영역을 ±32로 실험한 결과, 부호화 효율 측면에서는 BDBR[10]을 기준으로 평균 3% 가량의 비트율이 증가하는 것으로 나타났지만 복잡도 측면에서는 평균 5배 정도의 수행 속도 향상을 가져오는 것으로 나타났다. 표 3을 보면 결과가 영상에 따라 약간씩 차이를 보였는데 움직임이 빠르고 복잡한 영상일수록 고속 움직임 추정 기법의 성능이 떨어지며 그렇지 않은 영상에 대해서는 1% 정도의 미미한 성능의 차이가 존재하는 것을 볼 수 있다. 반면 수행 속도는 영상에 관계없이 일정하게 증가하고 있는 것을 볼 수 있다.

표 3. JM4.1b와 제안 기법의 성능 비교

Table 3. Performance comparison between JM4.1b and the proposed algorithm

	BDBR[%]	BDPSNR[dB]	speedup
CONTAINER	1.42	-0.07	5.05
NEWS	2.12	-0.12	5.04
FOREMAN	6.41	-0.35	5.02
SILENT	3.60	-0.17	5.01
PARIS	4.55	-0.23	4.99
MOBILE	1.29	-0.12	4.99
TEMPETE	1.22	-0.05	5.00
average	2.94	-0.16	5.01

V. 결 론

본 논문에서는 H.264의 움직임 보상을 위한 선택적 움직임 추정 알고리즘을 제안하였다. H.264 표준안이 기존에 비해 높은 부호화 효율을 얻도록 하는 주 요소인 복수 참조 영상을 이용한 다양한 블록 단위 움직임 보상은 H.264가 기존 부호화기에 비해 높은 복잡도를 갖도록 하는 원인이

되고 있었다. 그러나 기존의 DS 기반 고속 움직임 추정 알고리즘을 H.264 부호화시 그대로 적용하는 데에는 한계가 있기 때문에 먼저 간단한 탐색을 통해 최적의 움직임 벡터를 예측하고 이를 기준으로 차별적, 선택적 움직임 추정을 수행하는 기법을 제안하였다. 본 논문에서는 기존의 고속 움직임 추정 가운데 대표적인 DS(Diamond Search) 알고리즘을 적용하였지만 이 기법보다 더 효율적이고 향상된 고속 움직임 추정 기법들[6][7]도 충분히 적용될 수 있으며 이 경우 부호화 효율이나 연산량 감소의 측면에서 보다 더 향상된 결과를 가져올 것으로 예상한다.

참 고 문 헌

[1] "Text of ISO/IEC 14496-10 FCD Advanced Video Coding," *ISO/IEC JTC1/SC29/WG11*, N4920, Jul. 2002.

[2] "AHG report on editorial convergence of MPEG-4 reference software," *ISO/IEC JTC1/SC29WG11*, m8041, Mar. 2002.

[3] "Joint Model Number 1, Revision 1(JM-1r1)," *Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG*, JVT_A003r1, Dec. 2001.

[4] "A Computational Complexity Comparison of MPEG4 and JVT Codecs," *ISO/IEC JTC1/SC29WG11*, m8696, July 2002.

[5] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching," *IEEE Transactions on Image Processing*, Vol. 9, No. 2, pp. 287-290, Feb. 2000.

[6] A. M. Tourapis, O. C. Au, M.L. Liou, G. Shen and I. Ahmad, "Optimizing the MPEG-4 Encoder - Advanced Diamond Zonal Search," *IEEE International Symposium on Circuits and Systems*, (ISCAS2000), Vol. 3, pp. 674-677, May 2000.

[7] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation," *SPIE Conference on Visual Communications and Image Processing'01*, Vol. 4310, pp. 883-892, Jan. 2001.

[8] 전병우, 최웅일, "H.264 엔트로피 부호화 기법", *한국방송공학회논문지, 제 7권 3호*, pp. 54-64, Dec. 2002.

[9] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, Vol. 15, pp. 74-90, Nov. 1998.

[10] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-curves," *ITU-T Q.6/16*, Doc. #VCEG-M33, Mar. 2001.

저 자 소 개

최 웅 일

- 2000년 2월 : 성균관대학교 전기전자 컴퓨터 공학부 졸업(공학사)
- 2002년 2월 : 성균관대학교 정보통신공학부 졸업(공학석사)
- 2002년 3월~현재 : 성균관대학교 정보통신공학부 박사과정
- 주관심분야 : 영상압축, 멀티미디어 응용

전 병 우

- 1985년 2월 : 서울대학교 전자공학과 졸업(공학사)
- 1987년 2월 : 서울대학교 전자공학과 졸업(공학석사)
- 1992년 12월 : Purdue Univ, School of Elec. 졸업(공학박사)
- 1993년~1997년 8월 : 삼성전자 신호처리연구소 수석연구원
- 1997년 9월~현재 : 성균관대학교 정보통신공학부 부교수
- 주관심분야 : 멀티미디어, 영상압축, 영상인식, 신호처리