논문 2005-42SP-2-16

H.264에 적용을 위한 SEA 기반 고속 움직임 탐색 기법

(Fast motion estimation scheme based on Successive Elimination Algorithm for applying to H.264)

임 찬*, 김 영 문*, 이 재 은*, 강 현 수**

(Chan Lim, Young-Moon Kim, Jae-Eun Lee, and Hyun-Soo Kang)

요 약

본 논문에서는 연속제거 알고리즘(successive elimination algorithm)을 기반으로 하여 H.264 부호화기의 복잡도에서 가장 큰 비중을 차지하는 가변 블록에 대한 움직임 추정을 효율적으로 생략함으로써 고속으로 움직임 벡터를 탐색하는 기법을 제안한다. 제안된 기법은 7가지 모드의 가변 블록에 대하여 기존의 SEA를 계층적으로 적용한다. 즉, SEA를 사용해서 4x4 블록 단위로 SAD 또는 sum norm을 조합하고 이것을 각 모드의 최소 SAD 값과 비교 검색함으로써 불필요한 SAD 계산을 줄이는 방식이다. 그러므로 SEA의 SAD와 sum norm의 부등 관계에서 경계범위를 좁게 만들 수 있다. 단위 블록의 크기를 4x4 이하로 할 경우에는 경계 범위를 더욱 좁게 만들 수 있으나 계산량이 증가하는 단점이 있다. 제안된 기법을 적용했을 때에 각 실험영상에 따른 전체적인 계산량은 H.264의 고속 전역 탐색 방식에 비하여 약 60% ~ 70%의 일관된 감소가 있었다.

Abstract

In this paper, we propose a new fast motion estimation algorithm based on successive elimination algorithm (SEA) which can dramatically reduce heavy complexity of the variable block size motion estimation in H.264 encoder. The proposed method applies the conventional SEA in the hierarchical manner to the seven block modes. That is, the proposed algorithm can remove the unnecessary computation of SAD by means of the process that the previous minimum SAD is compared to a current SAD for each mode which is obtained by accumulating sum norms or SAD of 4x4 blocks. As a result, we have tighter bound in the inequality between SAD and sum norm than in the ordinary SEA. If the basic size of the block is smaller than 4x4, the bound will become tighter but it also causes to increase computational complexity, specifically addition operations for sum norm. Compared with fast full search algorithm of JM of H.264, our algorithm saves 60 to 70 % of computation on average for several image sequences.

Keywords : SEA, SAD, sum norm, variable block-based motion estimation

I. 서 론

움직임 추정과 보상은 디지털 비디오 부호화 시스템에 있어서 중요한 요소들이다. 영상 부호화 시스템을 위한 움직임 추정 방식으로는 현재 영상에 있는 블록과 시간적 중복성이 가장 많은 블록을 이전 영상에서 찾는 방식인 블록 정합 알고리즘(block matching algorithm)이 널리 사용되고 있다.[1][2] 특히 H.264의 움직임 추정을 위한 블록 크기는 16x16에서부터 4x4에 이르기까지 다양하다. 이러한 다양한 크기와 모양의 가변 블록 움직임 추정(variable block-based motion estimation) 기법은 부호화 효율에 있어서 많은 이득을 제공한다. 즉 비교적 배경의 변화가 적고 영상 내 객체의 크기가 크면서 변화가 적은 영상의 경우에는 큰 블록으로 움직임이 추정될 확률이 크지만, 복잡하고 변화가 심한 배경과 객체의 크기가 작은 영상은 작은 블록으로 움직임이 추정될 확률이 크다. 가변 블록의 모드별 블록 간의 유사성을 판단하기 위한 척도로서 SAD (sum of absolute difference)가 주로 사용된다. 전역 탐색(full-search) 방식은 대표적인 가변 블록 움직임 추정 알고리즘으로서 탐색 영역 내의 모든 후보 블록에 대하여 탐색을 수행

* 학생회원, 중앙대학교
(Chung-Ang University)
** 정회원, 충북대학교
(ChungBuk University)
※ 본 연구는 정보통신부 ITRC와 교육부의 두뇌한국 21사업(BK21)의 연구지원으로 수행되었습니다.
접수일자: 2005년1월4일, 수정완료일: 2005년2월11일

하므로 최적의 움직임 추정이 가능하지만 동시에 탐색 영역이 전역이기 때문에 계산량이 증가하는 문제점을 지닌다.[3][4] 이러한 문제점을 해결하기 위한 방식 중의 하나로서, 불필요한 계산량을 줄이면서 빠르게 최적의 해를 찾는 방식인 SEA (successive elimination algorithm)가 제안되었다.[5] 이 기법은 현재의 목표 블록과 탐색 영역 내 후보 블록간의 평균 성분 (sum norm)을 이용하여 두 블록 간의 불필요한 SAD 계산을 생략함으로써 계산량을 절감하는 이점이 있다. 본 논문에서는 동영상의 압축효율을 높이기 위하여 7가지(4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16)의 가변 블록 중 최적의 모드를 결정하는 방식에 대하여 SEA를 적용함으로써 고속으로 움직임을 탐색하는 기법을 제안한다. 4x4 모드를 기본 단위로 각 모드별로 탐색 영역 내에서 최적의 위치를 찾기 위한 최소한의 4x4 블록 SAD를 계산하고 각 모드의 나머지 부분은 현재 블록과 후보 블록에 대한 4x4 블록의 평균 성분으로 대체하여 모드에 따라 4x4 블록의 합으로 계산한다. 이 기법은 SEA의 SAD와 평균 성분의 부등 관계에서 4x4 보다 큰 블록에 대하여 이미 계산된 4x4 블록의 SAD와 평균 성분을 조합하여 경계값을 정함으로써 후보 블록에 대한 SAD 계산의 필요여부를 세밀하게 판별하도록 하는 것이다. 즉 가능성 없는 후보 블록은 순차적으로 제거하고 가능성 있는 후보 블록 (4x4)만을 블록 크기에 따라 검색하여 SAD를 계산함으로써 전체적인 계산량을 줄이고 탐색 영역 내의 모드별 최적의 움직임 벡터 (motion vector)를 구한다. 여기서 후보 블록의 가능성 여부를 판별하기 위한 검사를 모드별로 여러 번 수행해야 하기 때문에 이에 따른 계산량의 증가를 고려하여 최종적인 계산량을 얻는다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 JM 7.3의 전역 탐색 알고리즘 및 가변 블록 움직임 추정에 대한 간략한 소개와 개선점을 언급하고, Ⅲ장에서는 연속 제거 알고리즘을 이용한 가변 블록 움직임 추정을 설명한다. Ⅳ장에서는 제안 기법에 대한 실험 결과를 보여주고, Ⅴ장에서는 결론을 맺는다.

Ⅱ. 전역 탐색 알고리즘을 이용한 가변 블록 움직임 추정

전역 탐색 블록 정합 (full-search block matching) 알고리즘은 움직임 추정에 대한 최적의 해법으로 고려될 수 있다. 이 기법은 목표 블록에 대하여 탐색 영역

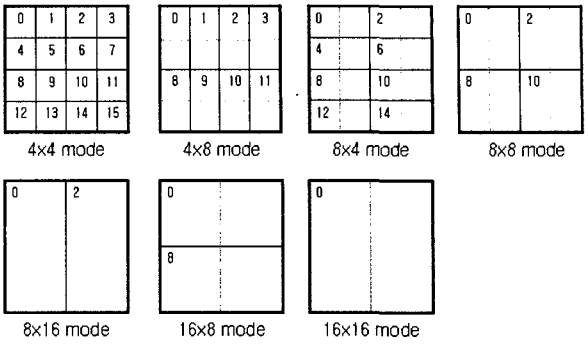

그림 1. H.264 (JM 7.3)의 움직임 추정을 위한 7가지 모드와 색인

Fig. 1. The 7 modes and the indexes for motion estimation in H.264.

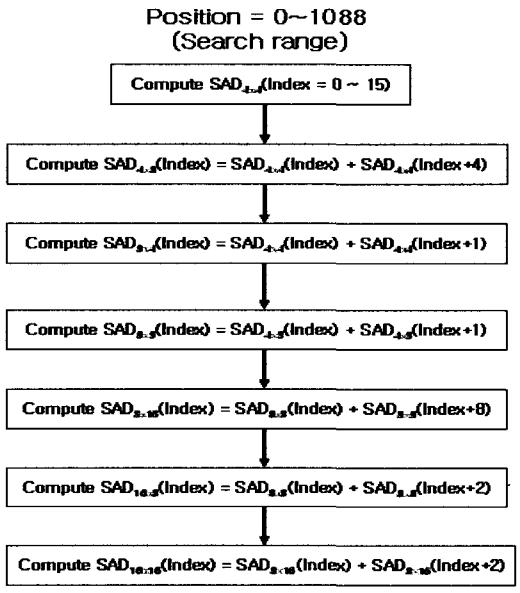

그림 2. H.264 (JM 7.3)의 고속 전역 탐색 알고리즘 흐름도

Fig. 2. The process of Fast full search algorithm in H.264.

내에서 모든 후보 블록을 탐색함으로써 최적의 블록 정합을 찾기 때문이다. 그러므로 최소 오류를 제공하지만 계산량에 대한 부담이 크므로 처리 과정에서 소모되는 시간이 많다.[4][6] 이런 문제점을 해결하기 위하여 고속 움직임 추정 알고리즘에 대한 연구가 이루어져 왔다.

H.264의 가변 블록 움직임 추정 기법은 그림 1에 표시된 것처럼 다양한 크기 (P8x8 - 8x8, 8x4, 4x8, 4x4 - 8x16, 16x8, 16x16)의 움직임 추정 구조를 가질 수 있는 방식이다. H.264에서는 다중의 영상 참조가 가능하기 때문에 크기가 8x8 이상인 블록들은 서로 다른 참조 영상을 가질 수 있다.[2][7][8]

그림 1은 JM 7.3 에서 제안된 16x16 MB 당 각 모드

별 색인을 나타낸다. 이런 다양한 구조의 움직임 추정은 영상 간의 시간적 상관성뿐만 아니라 영상 내부의 공간적 상관성을 고려해서 부호화가 가능한 효율적인 기술이다. 예를 들면 영상 내 단일 객체의 규모가 크고 배경의 변화가 적으면 객체의 움직임을 추정하기 위하여 큰 블록 모드가 결정될 확률이 크고 객체가 작고 움직임이 많아서 변화가 심한 경우에는 작은 블록 단위로 움직임 추정이 이루어질 가능성이 크다. 그러나 이 기법은 H.264의 부호화 과정 중에서 많은 시간을 소모하기 때문에 부호화 속도 향상을 위한 고속 움직임 추정 알고리즘이 요구된다. JM 7.3 에서의 전역 탐색 알고리즘을 이용한 가변 블록 움직임 추정은 그림 2의 흐름도로 나타내었다.

Ⅲ. 연속 제거 알고리즘을 이용한 가변 블록 움직임 추정

블록 정합 알고리즘의 목적은 이전 영상의 탐색 영역에서 현재 영상의 블록에 대한 최적의 참조 블록을 찾는 것이다. 움직임 추정 시에 가장 유사한 블록을 찾기 위한 정합 기준으로서 SAD 가 주로 사용된다. 연속 제거 알고리즘은 평균 성분과 SAD의 부등 관계를 이용하여 불필요한 SAD 계산을 생략하고 평균 성분으로 대체하여 전체적인 계산량을 줄이는 기법이다.

1. 연속 제거 알고리즘 (SEA)

독자의 이해를 돕고자 [5]의 내용을 설명하면 다음과 같다. 블록의 크기가 NxN이고 탐색 영역을 (2N+1) x(2N+1)로 가정했을 때 최적의 움직임 벡터를 얻기 위하여 탐색 영역 내에서 탐색 과정을 제한하는 부등식을 세웠다.

$$\begin{aligned} &||f(i,j,t)| - |f(i-x,j-y,t-1)|| \\ &\le |f(i,j,t) - f(i-x,j-y,t-1)| \end{aligned} \quad (1)$$

f(i, j, t)가 t번째 영상의 위치 (i, j)에서의 픽셀 값이고 x, y (-N ≦ x, y ≦ N) 가 움직임 벡터이면 수학적인 부등관계로부터 식(1)을 얻을 수 있다.

$$\begin{aligned} &|f(i,j,t)| - |f(i-x,j-y,t-1)| \\ &\le |f(i,j,t) - f(i-x,j-y,t-1)| \end{aligned} \quad (2)$$

$$\begin{aligned} &|f(i-x,j-y,t-1)| - |f(i,j,t)| \\ &\le |f(i,j,t) - f(i-x,j-y,t-1)| \end{aligned} \quad (3)$$

식 (1)은 위의 두 부등식 (2), (3)과 같이 나타낼 수 있다. 한 블록 내의 모든 픽셀에 대하여 식(2)와 식(3) 양쪽의 합을 구하면 식(4), 식(5)와 같다.

$$\begin{aligned} &\sum_{i=1}^{N}\sum_{j=1}^{N}|f(i,j,t)| - \sum_{i=1}^{N}\sum_{j=1}^{N}|f(i-x,j-y,t-1)| \\ &\le \sum_{i=1}^{N}\sum_{j=1}^{N}|f(i,j,t) - f(i-x,j-y,t-1)| \end{aligned} \quad (4)$$

$$\begin{aligned} &\sum_{i=1}^{N}\sum_{j=1}^{N}|f(i-x,j-y,t-1)| - \sum_{i=1}^{N}\sum_{j=1}^{N}|f(i,j,t)| \\ &\le \sum_{i=1}^{N}\sum_{j=1}^{N}|f(i,j,t) - f(i-x,j-y,t-1)| \end{aligned} \quad (5)$$

식(4)에서 첫 번째 합은 참조 블록의 평균 성분을, 두 번째는 움직임 벡터 (x, y)의 정합 후보 블록의 평균 성분을 나타내고 우변의 합은 움직임 벡터 (x, y)의 SAD를 나타낸다. 그러므로 식(4)와 식(5)를 식(6)과 식(7)로 표현할 수 있다.

$$R - M(x,y) \le SAD(x,y) \quad (6)$$

$$M(x,y) - R \le SAD(x,y) \quad (7)$$

움직임 벡터 (m, n)의 초기 정합 후보 블록에 대하여 SAD(m, n)을 얻었다고 한다면 움직임벡터 (x, y)의 더 적합한 후보 블록을 탐색하기 위하여 다음과 같은 관계식에 적용한다.

$$SAD(x,y) \le SAD(m,n) \quad (8)$$

$$R - M(x,y) \le SAD(m,n) \quad (9)$$

$$M(x,y) - R \le SAD(m,n) \quad (10)$$

위의 식들을 정리하면 부등식 (11)을 얻을 수 있다.

$$R - SAD(m,n) \le M(x,y) \le R + SAD(m,n) \quad (11)$$

식(11)은 SEA 알고리즘에 적용된다. SEA에서 최적의 정합을 얻기 위한 탐색은 단지 평균 성분이 식(11)을 만족하는 블록에서만 실행되어진다. 식(11)을 만족하는 블록들은 탐색 영역의 전체 블록들 보다 그 수가 확실히 적을 것이다. 그러므로 이 알고리즘은 최적의 위치를 배제하지 않고 탐색 과정의 계산적인 복잡도를 크게 줄일 수 있다. 이 알고리즘의 효율은 각 블록에 대한 평균 성분의 빠른 계산과 효과적인 초기 움직임 추정에

달려있다.

2. 제안된 가변 블록 움직임 추정 알고리즘

H.264 / MPEG-4 AVC 비디오 부호화 표준에서 시간적으로 압축된 매크로 블록 (MB)의 트리 구조 (tree-structured) 블록 크기는 움직임을 추정하는데 사용되고 각 16x16 MB은 16x16, 16x8, 8x16, 8x8 블록 모드로 부호화될 수 있다. 8x8 모드가 선택된다면 각 8x8 블록은 8x8, 8x4, 4x8, 4x4 블록으로 독립적으로 나누어질 수 있다. 그러므로 그림 1과 같이 모두 7가지의 다른 블록 크기가 만들어지고 이 블록 크기에 대하여 각 16x16 MB은 1, 2, 2, 4, 8, 8, 16개의 블록을 가진다. 움직임 추정을 위하여 H.264의 JM 7.3에서는 고속 전역 탐색 (fast full search) 알고리즘이 사용되었다. 그림 2의 흐름도에서 알 수 있듯이 각 4x4 블록의 SAD가 계산되고 다른 6가지 블록 모드의 SAD는 이미 계산되어진 4x4 블록의 SAD 합에 의해 구해진다. 4x4 블록 모드로 계산된 SAD를 재사용 할지라도 7가지 모드 전부에 대한 계산량은 16x16으로 크기가 고정된 하나의 MB에 대하여 전역 탐색 움직임 추정 알고리즘을 사용하는 것보다 증가할 것이다. 탐색 영역이 +/-16 픽셀이고 무제한(unrestricted) 움직임 벡터 모드가 활성화되면 전역 탐색 움직임 추정 알고리즘이 사용되는 탐색 점의 수는 1089이다. 그러므로 탐색 점이 1089 일 때 4x4 블록의 SAD를 사용한 7가지 모드의 SAD 계산량은 16x16 MB 단독의 SAD 계산량보다 크다. 즉 가변 블록의 정합은 크기가 고정된 블록의 정합에 비하여 계산량이 많다. 본 논문에서는 SEA를 사용함으로써 각 모드에 따라 4x4 블록 단위로 불필요한 SAD 계산을 줄이는 기법을 제안한다. 각 모드별로 SAD 계산이 필요한 4x4 블록을 탐색하는 방법은 앞에서 언급된 SEA의 SAD와 평균 성분과의 비교 검색과 가변적인 블록 크기의 적절한 조합으로 이루어진다. 8x8 블록의 부 블록 (sub-block) 인 4x4 블록에 [9]의 증명을 사용하면 식 (12), (13)이 성립한다.

$$\left|\sum_{i=1}^{4}(R_i - M_i(x,y))\right| = |R - M(x,y)| \le \sum_{i=1}^{4}|R_i - M_i(x,y)| \tag{12}$$

$$\sum_{i=1}^{4}|R_i - M_i(x,y)| \le |R_1 - M_1(x,y)| + SAD_2(x,y) + SAD_3(x,y) + |R_4 - M_4(x,y)| \le \sum_{i=1}^{4} SAD_i(x,y) = SAD(x,y) \tag{13}$$

16x16 MB의 다른 부 블록 (16x8, 8x16, 8x4, 4x8)에 적용해도 위의 사실이 성립함을 알 수 있다. 그림 3에서 알 수 있듯이 SAD 계산의 필요여부를 결정하기 위하여 식(12)는 8x8 블록을 부 블록 (4x4)으로 나누어서 각 블록 평균 성분들의 차에 대한 절대치를 구하여 합한 경우에는 각 블록 평균 성분들의 차에 대한 합을 구하여 절대치를 취한 것보다 좁은 경계 범위를 얻을 수 있다는 것을 의미한다. 식(13)에 의하면 일부 4x4 블록의 평균 성분을 4x4 모드의 비교 과정에서 이미 계산되어진 SAD로 대체하고 나머지 4x4 블록의 평균 성분과 더하여 8x8 블록의 경계 값을 구함으로써 더욱 좁은 경계 범위를 얻을 수 있다. 그러므로 모드가 4x4 보다 크다면 순수하게 4x4 블록의 평균 성분으로만 조합한 경우보다 4x4 블록의 평균 성분을 SAD로 대체하여 조합한 경우가 경계 범위가 좁고 해당 모드에서의 4x4 블록에 대한 SAD의 추가적인 계산을 줄일 수 있다. 하지만 제한을 두지 않고 더 작은 부 블록으로 나눈다면 식 (13)에서 더하기 연산의 계산량이 증가한다는 단점이 있다.

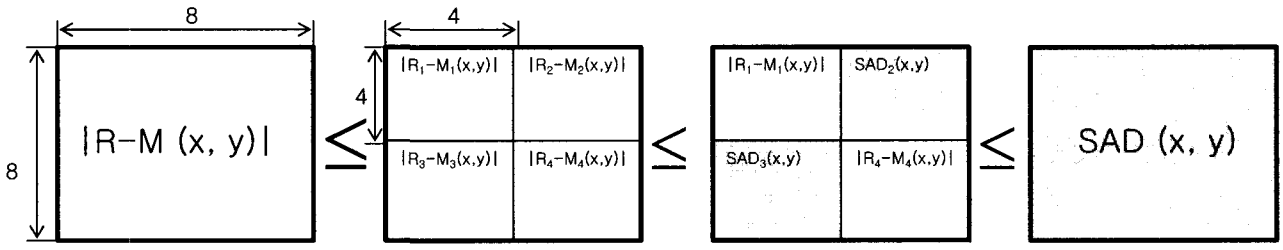

그림 3. 4x4 블록의 SAD와 sum norm의 조합에 의한 8x8 블록의 경계값 대소비교
Fig. 3. The comparison of the bound of 8x8 block by the accumulation of SAD and sum norm for 4x4 blocks.

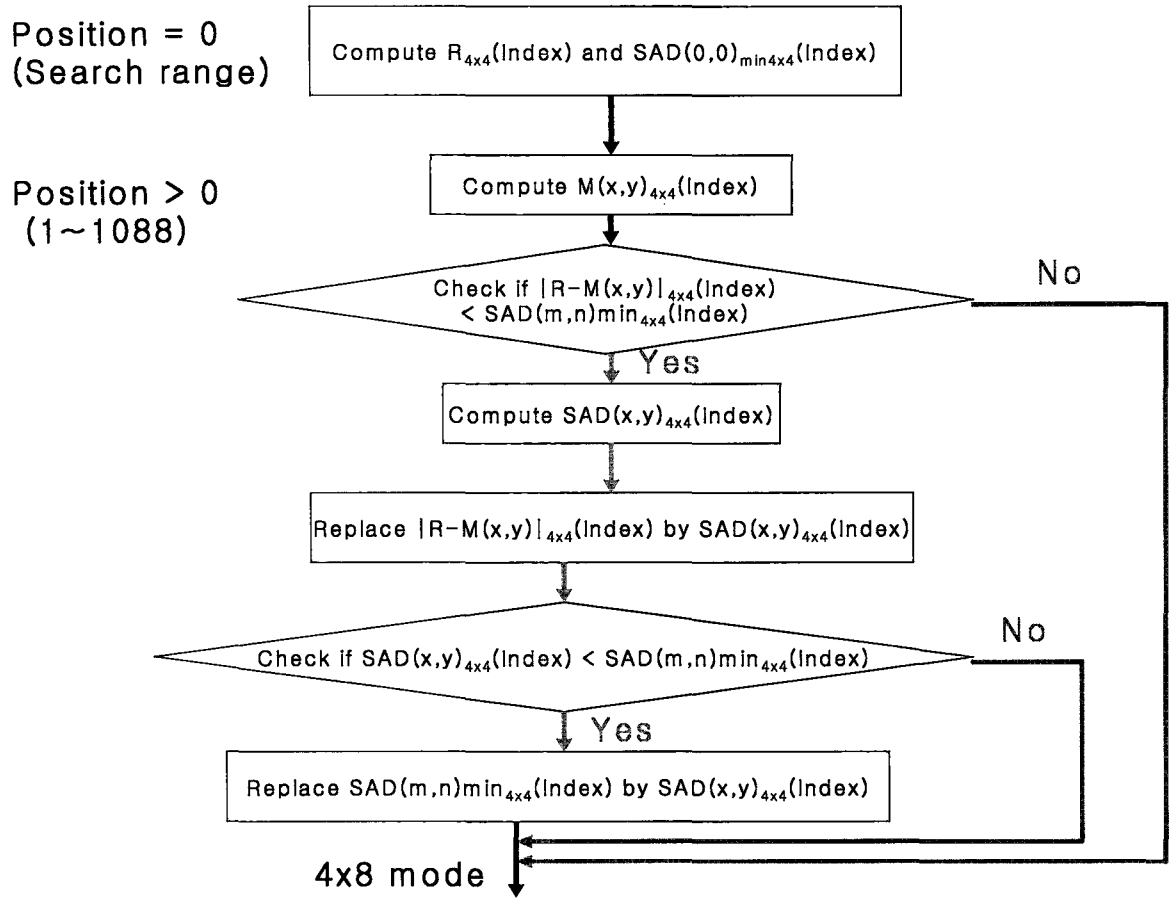

그림 4. 제안된 알고리즘에서 SAD의 계산 여부 를 결정하기 위한 4x4 블록 탐색 흐름도
Fig. 4. The search process for 4x4 block to decide whether SAD computation is or not needed in the proposed algorithm.

본 논문에서는 SEA를 적용하여 얻은 4x4 블록에 대한 SAD 및 평균 성분을 사용하여 다른 큰 블록들의 경계범위를 좁게 만듦으로써 모드별 움직임 탐색에 필요한 SAD의 계산량을 전체적으로 감소시킨다.

그림 4에서 알 수 있듯이 다음은 4x4 블록 단위로 SEA를 적용하여 SAD 계산이 불필요한 4x4 블록을 제거하는 절차이다. SAD(0,0)min은 처음 탐색위치에서의 SAD 값인데 비교 대상이 없으므로 초기의 최소값으로 간주한다.

Step 1. 탐색위치가 0 이면 4x4 블록의 각 색인별로 R_{4x4}와 $SAD(0,0)_{min4x4}$를 구한다.

Step 2. 탐색위치가 0 보다 크면 탐색영역에서 4x4블록의 색인별로 $M(x,y)_{4x4}$를 구한다.

Step 3. $|R-M(x,y)|_{4x4}$과 $SAD(m,n)_{min4x4}$를 비교한다.

3.1. $|R-M(x,y)|_{4x4}$이 크면 Step 8로 간다.

3.2. $SAD(m,n)_{min4x4}$가 크면 Step 4로 간다.

Step 4. 그 위치에서 $SAD(x,y)_{4x4}$를 구한다.

Step 5. $|R-M(x,y)|_{4x4}$를 $SAD(x,y)_{4x4}$로 대체한다.

Step 6. $SAD(x,y)_{4x4}$와 $SAD(m,n)_{min4x4}$를 비교한다.

6.1. $SAD(x,y)_{4x4}$가 크면 Step 8로 간다.

6.2. $SAD(m,n)_{min4x4}$가 크면 Step 7로 간다.

Step 7. $SAD(m,n)_{min4x4}$를 $SAD(x,y)_{4x4}$로 대체한다.

Step 8. 현재 탐색 위치의 SAD_{min4x4}를 이전 탐색위치의 SAD_{min4x4}로 갱신한다.

다음은 그림 5 및 그림 6 에서와 같이 4x4 블록의 움직임 추정을 4x8 블록에 재사용하는 절차에 대한 예를 나타낸다.

Step 1. 탐색위치가 0 일 때 4x4 블록에서 계산된 SAD 을 사용하여 4x8 블록 색인(I0)의 $SAD(0,0)_{min4x8}$를 구한다.

Step 2. 탐색위치가 0 보다 클 때 4x4 블록에서 계산된 SAD와 평균성분의 차에 대한 절대치를 사용하여 탐색영역에서 다음을 구한다.

$|R-M(x,y)|_{4x4}(I0)+SAD(x,y+4)_{4x4}(I4)$

Step 3. 4x4 블록 색인(I0)의 $|R-M(x,y)|_{4x4}$와 4x4 블록 색인(I4)의 SAD 합을 4x8 블록 색인(I0)의 $SAD(m,n)_{min4x8}$와 비교한다.

3.1. $|R-M(x,y)|_{4x4}(I0)+SAD(x,y+4)_{4x4}(I4)$ 이 크면 Step 8로 간다.

3.2. $SAD(m,n)_{min4x8}$가 크면 Step 4로 간다.

Step 4. 그 탐색 위치에서 $SAD(x,y)_{4x4}(I0)$를 구하고 $SAD(x,y+4)_{4x4}(I4)$와 합하여 다음을 구한다.

$SAD(x,y)_{4x8}(I0),(|R-M(x,y)|_{4x4}(I0)$를 $SAD(x,y)_{4x4}(I0)$로 대체)

Step 5. $SAD(x,y)_{4x8}$와 $SAD(m,n)_{min4x8}$를 비교한다

5.1. $SAD(x,y)_{4x8}$가 크면 Step 7로 간다.

5.2. $SAD(m,n)_{min4x8}$가 크면 Step 6으로 간다.

Step 6. $SAD(m,n)_{min4x8}$를 $SAD(x,y)_{4x8}$로 대체한다.

Step 7. 현재 탐색 위치의 SAD_{min4x8}를 이전 탐색 위치의 SAD_{min4x8}로 갱신한다.

Step 8. 4x8 블록의 각 색인별로 반복한다.

Step 9. 8x4, 8x8, 8x16, 16x8, 16x16 모드별로 반복한다.

그림 6과 같이 각 모드별로 앞의 절차를 반복하면 최적 모드 결정을 위한 움직임 탐색에 있어서 SAD 계산이 필요한 4x4 블록의 수를 알 수 있다. 그림 6에서 표시된 부분은 모드별 탐색에 의하여 SAD가 계산된 블

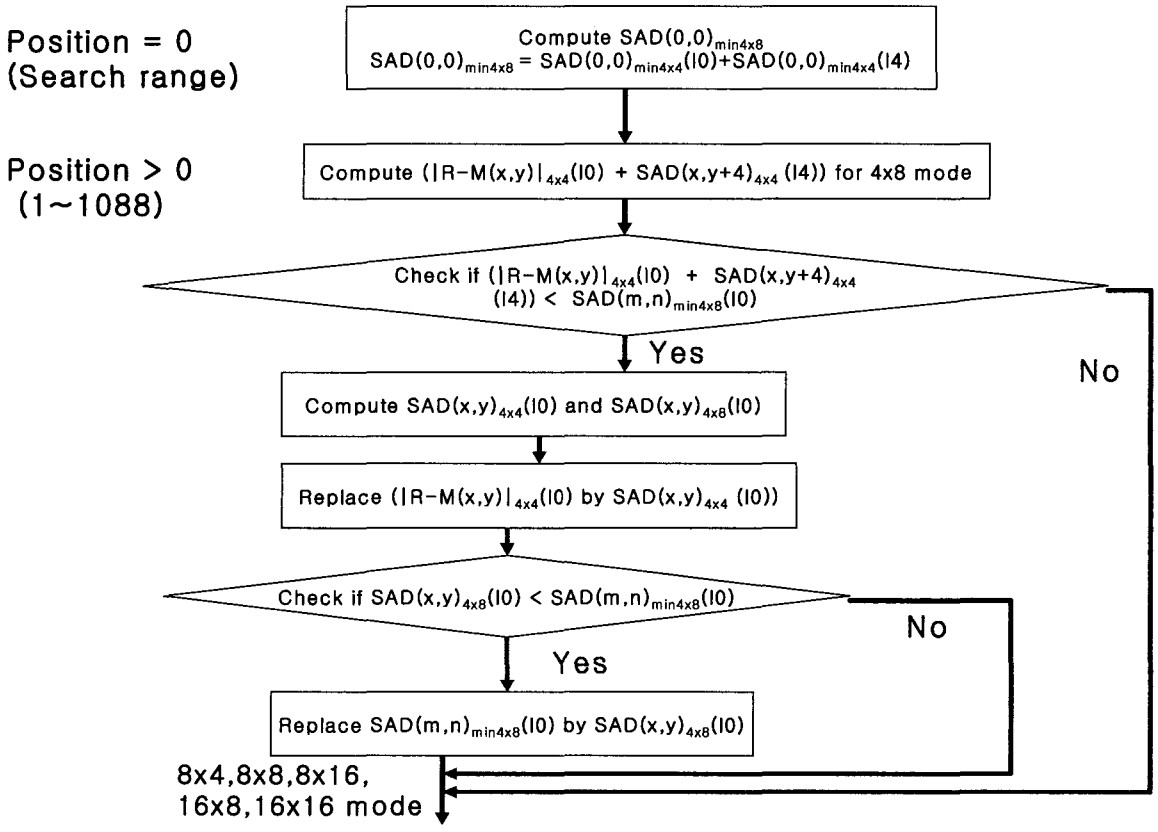

그림 5. 제안된 알고리즘에서 SAD의 계산 여부를 결정하기 위한 4x8 블록 탐색 흐름도. (I0)과 (I4)는 블록별 색인을 표시

Fig. 5. The search process for 4x8 block to decide whether SAD computation for 4x4 block is or not needed in the proposed algorithm. (I0) and (I4) are the indexes in each mode.

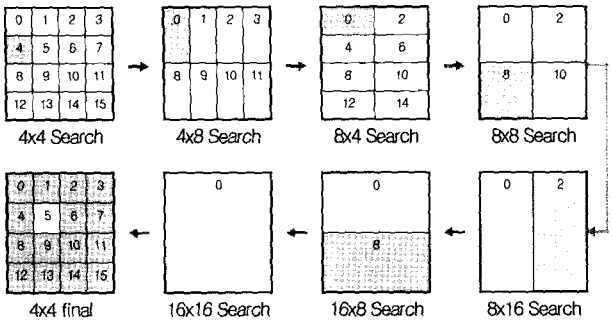

그림 6. 제안된 알고리즘에서 SAD 계산이 필요한 4x4 블록의 모드별 탐색 과정

Fig. 6. The search process for each mode to need the computation of SAD in the proposed algorithm.

록을 나타낸다. 설명의 편의를 위하여 각 모드별 탐색을 거치면서 대부분의 블록에서 SAD가 계산된 예를 나타내었다. 그러나 실제 실험에서는 표 1에서와 같이 영상에 따라서 약 10% ~ 20%의 4x4 블록 SAD 계산율을 얻을 수 있었다.

실험 영상에 따른 전체적인 계산량은 JM 7.3의 고속 전역 탐색 방식에 비하여 약 60% ~ 70%의 감소가 있다. 이 때 SEA를 통하여 계산된 각 모드별 SAD 값과 JM 7.3의 고속 전역 탐색 방식에 의해 계산된 SAD의 모드별 값을 탐색 영역에 따라서 비교하여 최소치를 구하면 동일한 움직임벡터를 얻는다. 그러므로 본 논문에서 제안된 알고리즘의 타당성을 증명할 수 있다.

IV. 실험결과 및 고찰

제안 알고리즘의 성능 평가는 H.264의 JM 7.3에서 제안된 고속 전역 탐색 방식과의 연산량 비교를 통하여 이루어졌다. 이를 위하여 덧셈 (ADD), 뺄셈 (SUB), 비교 (COMP)는 1개의 연산으로, 절대치 (ABS)는 2의 보수를 취하는 방식으로 간주하여 2개의 연산으로 계산함으로써 총 연산량을 측정하였다. 실험은 다섯 개의 QCIF (176x144) 형식의 표준 동영상 (Foreman, Coast guard, Mother & daughter, Stefan, Table Tennis)에 대하여 실행되었다. 입력 동영상은 100 프레임을 사용하였고 참조 영상은 바로 이전의 영상 한 장이다. 움직임 추정을 위하여 사용된 탐색 점은 N=16 일 때 (2N+1) * (2N+1) = 1089 개다.

표 1은 각 실험 영상들에 대한 결과를 보여준다. 표 1의 수치는 입력 영상이 100프레임 일 때 탐색 영역을 따라서 매 16x16 MB에 대하여 각 블록 모드 별로 SAD가 계산된 4x4 블록의 수를 전체 4x4 블록의 수에 대한 백분율로 나타낸 것이다. 즉 최적의 모드를 결정하기 위하여 모드별로 움직임을 탐색함에 있어서 16x16 MB 당 SAD 계산이 필요한 모드별 4x4 블록의 평균을 의미한다.

표 1. 실험 영상(100 frames)에서 16x16 MB 당 각 모드별로 SAD가 계산된 4x4 블록 수의 전체 평균을 백분율 (%)로 환산한 결과

Table 1. The ratio of the total number of 4x4 blocks with SAD computation to the total number of 16x16 MBs according to each mode.

image / mode	Foreman	Coast guard	Mother_ daughter	Stefan	Table tennis
4x4	7.772	16.986	6.721	13.241	19.227
4x8	0.457	0.364	0.626	0.587	0.665
8x4	0.421	0.369	0.481	0.614	0.606
8x8	0.296	0.239	0.451	0.435	0.446
8x16	0.284	0.234	0.332	0.38	0.459
16x8	0.281	0.228	0.276	0.384	0.407
16x16	0.24	0.222	0.22	0.331	0.439
4x4total	8.376	17.483	7.496	14.067	20.109

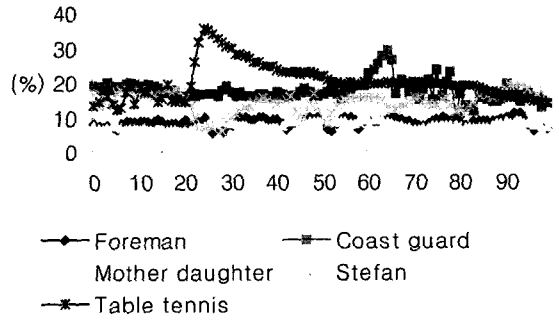

그림 7. 각 동영상에서 프레임별로 SAD가 계산된 4x4블록의 평균치에 대한 백분율(%)

Fig. 7. The average of the number of 4x4 blocks per frame that SAD was computed for each image sequence.

그림 7은 각 영상에서 SAD가 계산된 4x4 블록의 평균치를 프레임 별로 나타낸 것이다. 표 1에서 알 수 있듯이 SAD가 계산된 4x4 블록의 개수가 Foreman 과 Mother & daughter 영상의 경우에는 적고 이에 비하여 Table tennis와 Coast guard 영상은 비교적 많다. Table tennis와 Coast guard 영상의 경우에는 전체적으로 평탄한 영역을 많이 가지면서 객체의 움직임이 완만한 반면에 그림 7에서 나타난 것처럼 Table tennis의 경우에는 약 25번째 프레임에서 SAD가 계산된 4x4 블록의 개수가 급격히 증가하는 것을 알 수 있다. Table tennis 영상을 살펴보면 25번째 프레임을 전후로 하여

화면이 zoom out 되면서 객체의 크기가 급격히 변화하는 것을 알 수 있다. Coast guard 영상은 65번째 프레임 부근에서 화면 전체가 흔들리고 두 개의 객체가 겹쳐지는 것을 알 수 있다. 반대로 Foreman과 Mother & daughter 영상은 객체의 부분적인 움직임은 있으나 전체적으로 안정된 영상을 나타냈다. Stefan의 경우에는 지속적인 객체의 움직임은 있으나 크기의 변화는 없었다. 따라서 다음과 같은 결론을 유추할 수 있다. 영상 내에서 화면의 급격한 변화, 이를테면 배경이나 객체의 움직임 및 크기의 변화가 심할 경우에는 화면이 평탄하고 움직임이 완만한 영상에 비하여 SAD를 계산해야하는 블록의 개수가 증가한다. 이것은 SEA에서의 SAD와 평균성분의 부등 관계에서 알 수 있듯이 객체 또는 화면 자체의 변화가 심한 영상에서는 더욱 세밀한 움직임 탐색을 위하여 SAD의 계산이 필요하다는 것을 의미한다.

표 2는 1개의 4x4 블록 SAD의 연산량과 평균 성분의 연산량을 각 단계별로 나누어 비교한 것이다. 단계별 수치에서 나타나듯이 SAD와 평균 성분은 그 연산량에 있어서 많은 차이가 있다. 탐색 영역 2 ~ 1088 에서 M의 연산 횟수는 이전에 계산된 데이터에 의하여 8회의 연산만 추가하면 된다. 즉 4x4 블록 1개가 이전의 탐색 점에서 15회의 ADD 연산을 수행했다면 1개의 픽셀만큼 이동한 현재의 탐색 점에서는 새롭게 추가된 4개의 새로운 픽셀에 대해서만 ADD 연산을 하고 불필요한 4개의 이전 픽셀에 대해서는 SUB 연산을 수행하면 된다. 그래서 탐색 영역 2 ~ 1088 에서 필요한 각 평균 성분의 연산량은 15회가 아닌 8회가 된다. 표 3-1과 표 3-2 는 JM 7.3에 제시된 고속 전역 탐색 방식의 연산량과 SEA를 이용한 제안된 방식의 연산량을 비교하여 보여준다. 제안된 방식은 영상 내에서 각 모드별로 객체의 움직임을 탐색하기 위하여 SAD 계산이 필요한 4x4 블록의 개수를 구하는 절차를 표 3-1과 표 3-2에 제시하였다. 그리고 표 3-2에 제시된 바와 같이 5개의 연산 과정을 기준으로 알고리즘을 실행했을 때 전체적으로 소모되는 연산량을 측정하였다.

즉 먼저 4x4 블록 단위로 평균 성분을 구하고 기존에 계산된 SAD의 최소치와 비교하여 대소를 결정한다. 평균 성분이 기존 SAD의 최소치 보다 작은 경우에는 그 위치에서 평균 성분을 대신하여 4x4 블록의 SAD를 구하고 이 값을 다시 기존 SAD의 최소치와 대소 비교하여 기존 SAD의 최소치를 갱신한다. 4x4 보다 큰 블록의 경우도 위의 과정을 반복하되 이미 계산된 4x4 블록에 대한 평균 성분과 SAD 값을 기본 단위로 각 블록 모드에 따라서 합의 연산을 취하여 중복되는 계산을 줄이고 대소 비교에 필요한 경계 범위를 좁게 유지한다.

표 3-1은 고속 전역 탐색 방식의 전체적인 연산량과 제안된 방식의 절차 중에서 모든 영상에서 공통적으로 소모되는 연산량을 나타내고 표 3-2는 제안된 방식의 절차 중에서 실험 영상에 따라서 별도로 소모되는 개별적인 연산량을 나타낸다. 표 3-2와 그림 8의 최종 결과에서 나타난 것처럼 제안된 알고리즘이 H.264의 고속 전역 탐색 알고리즘에 비하여 약 60 ~ 70 %의 계산량 감소가 있다는 것을 알 수 있다.

표 2. 4x4 블록 1개에 대한 SAD의 연산량과 sum norm 연산량 비교: 각 단계별로 나누어 연산량 표시

Table 2. The comparison of the computational cost of SAD and that of sum norm per 4x4 block: the computationalmcost for each step was represented in this table.

		SAD_{4x4}	$\|R-M\|_{4x4}$			
Search Range		0 ~ 1088	0	1 ~ 1088	1	2 ~ 1088
ADD (+)	SAD	15				
	R		15			
	M				15	4
SUB (-)		16		1		4
ABS (| |)		16		1		

표 3-1. 고속 전역 탐색 방식(JM 7.3)의 연산량과 제안된 방식의 연산량

Table 3-1. The computational cost of Fast full search algorithm vs the common computational cost of the proposed algorithm.

	FFSA (JM 7.3)	Proposed + SEA (Common)	
		① |R-M|	② COMP (|R-M| vs SAD_{min})
SAD_{4x4}	1097712	1008	
$\|R-M\|_{4x4}$		155872	
4x4 (+)			17408
4x8 (+)	8712	8712	8704
8x4 (+)	8712	8712	8704
8x8 (+)	4356	4356	4352
8x16 (+)	2178	2178	2176
16x8 (+)	2178	2178	2176
16x16 (+)	1089	1089	1088
Total	1124937	228713 (Common)	

표 3-2. 제안된 방식에서 실험 영상에 따른 추가 연산량 (소수점 이하 반올림)
③ SAD4x4, ④ SAD+|R-M| ⑤ COMP (SAD+|R-M| vs SADmin)
Table 3-2. The computational overhead of the proposed method for each sequence.

	Proposed + SEA (Individual)														
Image	Foreman			Coast guard			Mother daughter			Stefan			Table		
Operation	③	④	⑤	③	④	⑤	③	④	⑤	③	④	⑤	③	④	⑤
4x4 (+)	91854	1353		191709	2957		82215	1170		154287	2305		220563	3347	
4x8 (+)		80			64			110			102			116	
8x4 (+)		74			64			84			108			106	
8x8 (+)		26			20			40			38			38	
8x16 (+)		12			10			14			16			20	
16x8 (+)		12			10			12			16			18	
16x16(+)		6			4			4			8			10	
Individual	93417			194818			83649			156878			224218		
Common + Individual	322130			423531			312362			385591			452931		
(Proposed x100) / FFSA(%)	28.6			37.6			27.8			34.3			40.3		

V. 결 론

본 논문에서는 H.264의 가변 블록 움직임 추정 방식의 계산량을 줄이는 알고리즘으로서 SEA 기반 고속 움직임 탐색 기법을 제안하였다. 제안 알고리즘은 SEA에서의 SAD와 평균 성분(sum norm)의 부등 관계를 이용하여 불필요한 SAD의 계산을 절감하였다. 그리고 최적움직임 벡터를 결정하기 위하여 16x16 MB 당 각 모드별로 모든 후보 블록에 대하여 SAD를 계산하는 대신에 4x4 블록 단위로 움직임 탐색에 필요한 최소한의 SAD만을 계산함으로써 가변 블록을 이용한 탐색 방식의 전체적인 계산량을 절감할 수 있었다. 즉 SEA의 SAD와 평균 성분의 부등 관계에서 경계 범위를 좁게 취하기 위하여 식 (12)와 식 (13)에 제시된 것처럼 각 4x4 블록에 대한 평균 성분의 합 뿐만아니라 덧붙여 이미 계산된 SAD를 조합하여 더 큰 모드의 경계값을 구함으로써 4x4 블록 단위의 SAD 계산 횟수를 최소화하였다.

제안 알고리즘은 다양한 움직임 특성을 가진 여러 가지 영상임에도 불구하고 H.264의 JM 7.3에서 제안된 고속 전역 탐색 알고리즘에 비하여 약 60 ~ 70%의 일관된 계산 이득을 나타내었다. 즉 실험 영상 내의 배경과 객체의 움직임 및 크기 등의 변화에 따라 최대 70% 이상의 계산량을 절감 할 수 있었다. 이상의 실험 결과로부터 제안 알고리즘이 H.264의 고속 전역 탐색 알고리즘

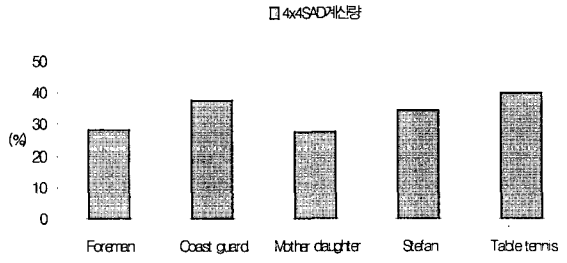

그림 8. H.264의 고속 전역 탐색 알고리즘과 SEA를 이용한 제안 알고리즘의 전체적인 계산량 비교: 고속 전역 탐색 알고리즘에 대한 제안 알고리즘의 계산량의 비를 백분율(%)로 표시
Fig. 8. The total computational cost of the proposed algorithm compared with that of fast full search algorithm of H.264: The ratio of the total computational cost of the proposed algorithm to that of fast full search algorithm of H.264.

보다 적은 계산량으로 가변 블록에 대한 최적의 움직임 벡터를 얻을 수 있음을 확인하였다.

참 고 문 헌

[1] M. Brunig, W. Niehsen, "Fast full-search blockmatching", *IEEE Trans. on Circuits and Systems for video Technology*, Vol.11, No.2, pp. 241-247, Feb. 2001.
[2] S. Y. Huang, J. R. Chen, J. S. Wang, K. R.

Hsieh, and H. Y. Hsieh, "Classified variable block size motion estimation algorithm for image", *IEEE International Conference on Image Processing*, Vol.3, pp. 736-740, Nov 1994.
[3] A. Ahmad, N. Khan, S. Masud, and M. A. Maud, "Selection of variable block sizes in H.264", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, Vol.3, pp. 173-176, May 2004.
[4] T. G. Ahn, Y. H. Moon, and J. H. Kim, "Fast Full-Search Motion Estimation Based on Multilevel Successive Elimination Algorithm", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.14, No.11, pp. 1265-1269, Nov. 2004.
[5] W. Li, E. Salari, "Successive elimination algorithm for motion estimation", *IEEE Trans. on Image Processing*, Vol.4, No.1, pp. 105-107, Jan. 1995.
[6] T. M. Oh, Y. R. Kim, W. G. Hong, and S. J. Ko, "A fast full search motion estimation algorithm using the sum of partial norms", *International Conference on Consumer Electronics*, pp. 236-237, June 2000.
[7] Z, Zhou, M. T. Sun, and Y.. F. Hsu, "Fast variable block-size motion estimation algorithm based on merge and slit procedures for H.264 / MPEG-4 AVC", *International Symposium on Circuits and Systems*, Vol.3, pp. 725-728, May 2004.
[8] W. Choi, J. Lee, S. Yang, and B, Jeon, "Fast motion estimation and mode decision with variable motion block sizes", *Proc. of SPIE, Visual Communications and Image Processing (VCIP)*, July 2003.
[9] Y. Noguchi, J. Furukawa, and H. Kiya, "A fast full search block matching algorithm for MPEG-4 video", *International Conference on Image Processing*, Vol.1, pp. 61-65, 1999.

저 자 소 개

임 찬(학생회원)
2000년 아주대학교 전자공학과 학사졸업.
2000년 LG전자 연구원
2005년 현재 중앙대학교 첨단영상대학원 영상공학과 석사과정.
<주관심분야 : 영상통신, 부호화, 영상처리>

이 재 은(학생회원)
2001년 중앙대학교 기계공학부 학사졸업.
2001년 LG CNS 연구원
2005년 현재 중앙대학교 첨단영상대학원 영상공학과 석사졸업.
<주관심분야 : 사운드, 부호화, 영상처리>

김 영 문(학생회원)
2003년 중앙대학교 컴퓨터공학과 학사졸업.
2005년 중앙대학교 첨단영상대학원 영상공학과 석사졸업.
<주관심분야 : 영상처리, 영상통신, 신호처리>

강 현 수(정회원)
1999년 한국과학기술원 전기 및 전자공학과 박사졸업.
1995년 하이닉스반도체(주) 선임연구원
2001년 한국전자통신연구원 선임연구원
2002년 3월 ~ 2005년 2월 중앙대학교 첨단영상대학원 영상공학과 조교수
2005년 3월 현재 충북대학교 전기전자컴퓨터공학부 조교수
<주관심분야 : 영상처리, 부호화, 컨텐츠보호기술, 사운드>