초기 탐색 위치의 효율적 선택에 의한 고속 움직임 추정

정회원 남 수 영*, 김 석 규**, 임 채 환***, 김 남 철****

Fast Motion Estimation Using Efficient Selection of Initial Search Position

Soo-Young Nam*, Seog-Gyu Kim**, Chae-Whan Lim***,
and Nam-Chul Kim**** *Regular Members*

요 약

본 논문에서는 효과적으로 선택된 초기 탐색 위치를 이용한 움직임 추정의 고속 알고리즘을 제안하였다. 제안된 알고리즘에서는 2×2화소 블록 평균으로 부표본화 영상에서 움직임 벡터를 얻어 원영상 비율로 확대하고, 주위 블록의 움직임으로부터 예측 움직임 벡터를 구하여, 이 중에서 정합오차가 작은 것을 초기 탐색 위치로 선택한다. 그리고 선택된 초기 탐색 위치를 중심으로 회전 탐색을 시작하여, 연속 소거 알고리즘으로 탐색할 후보 블록을 선택하고, 부분 정합 왜곡 소거법을 사용하여 블록간 정합오차 계산량을 줄이면서, 고속으로 움직임 벡터를 추정한다. 알고리즘의 실제 적용에 있어서는 선택된 초기 탐색 위치를 중심으로 회전 탐색 패턴의 탐색 범위를 조절하거나, 매크로 블록 당 복잡도를 제한하여 계산량을 줄일 수 있다. 실험 결과, 제안된 알고리즘은 전역탐색 블록정합 알고리즘에 대하여 0.2dB 이하의 미소한 평균 PSNR 저하만을 발생하면서, FBMA 복잡도의 3% 이하의 평균 복잡도를 소요하였다. 이것은 3단계 탐색법에 대하여 40% 이하의 계산량이다. 그리고 실험 영상들의 각 프레임에 대해서도 비슷한 성능을 보임을 확인하였다.

ABSTRACT

In this paper, we present a fast algorithm for the motion estimation using the efficient selection of an initial search position. In the method, we select the initial search position using the motion vector of the subsmpled images and the predicted motion vector from the neighbor blocks. We do the spiral search pattern at the selected search position, select the candidate blocks using the successive elimination algorithm and eliminate the complexities of the matching by the partial distortion elimination, so do the fast motion estimation. The search complexity can be reduced with quite a few PSNR loss by controlling the search range in the spiral search pattern or limiting the complexity per a macroblock. The experiment results show that the complexity of the proposed algorithm is about 3% less than the full search algorithm, that is about 40% less than the three-step search, with the PSNR loss of just 0.2dB from the full search algorithm. The experiment result of each frame shows that similar performance.

* 경북대학교 전자전기공학부(salgoo@vcl.knu.ac.kr),
** 삼성전자 정보통신총괄 무선사업부(cwlim@tel.samsung.co.kr),
논문번호 : K01011-0108, 접수일자 : 2001년 1월 8일
*** 대구서부공업고등학교 전자통신과(ksgjw@kebi.com)
**** 경북대학교 전자전기공학부(nckim@ee.knu.ac.kr)

Ⅰ. 서 론

동영상 부호화에서는 영상의 시간적 중복을 줄이는 방법으로 움직임 벡터 추정 및 보상(motion vector estimation and compensation) 기술을 이용한다. 특히, 움직임 벡터 추정은 동영상 부호화에서 가장 많은 계산량을 필요로 하는 부분이기 때문에 부호화 계산량을 줄이기 위한 주요 대상이 되고 있다. H.263[1], MPEG[2] 등의 다양한 비디오 부호화 표준에서 움직임 벡터 추정에 사용되는 블록정합 알고리즘(block matching algorithm)은 연속된 두 프레임에서 영상을 일정한 크기의 작은 블록 단위로 나눈 다음, 현재 프레임의 각 블록에 대하여 이전 프레임의 동일위치나 이웃위치 블록과의 정합도에 기초하여 움직임을 추정한다. 전통적인 블록정합 알고리즘인 전역탐색 블록정합 알고리즘(FBMA : full search BMA)[3]은 탐색 영역(search range) 안에서 모든 움직임 후보 벡터들을 탐색하기 때문에 많은 계산량을 요구하므로 동영상 부호화에 큰 부담이 되고 있다. 지금까지 이러한 계산량의 부담을 줄일 수 있는 고속 알고리즘들이 많이 연구되어 왔다.

FBMA에 대해 성능저하 없이 계산량만을 줄이는 알고리즘으로 연속소거 알고리즘(SEA : successive elimination algorithm)[4], 부분정합 왜곡소거법(PDE : partial distortion elimination)[5] 등이 있다. SEA는 현재 프레임의 참조 블록(reference block)과 이전 프레임의 후보 블록(candidate block)들 각각의 평균을 구하고, 블록 평균의 절대차가 초기 정합오차보다 큰 후보 블록들을 제거한다. PDE는 참조 블록과 후보 블록 사이의 정합오차를 계산하는 과정에서 부분 정합오차가 이전까지 구해진 최소 정합오차보다 커지면 블록 내의 나머지 계산을 하지 않고 다음 후보 블록과의 정합오차 계산으로 넘어간다. 위의 방법들을 적용할 경우, 후보 블록을 탐색하는 패턴을 어떻게 정하는 가에 따라 더욱 효과적으로 계산량을 줄일 수 있는데, 이에는 회전 탐색 패턴(spiral search)[6], 다이아몬드 탐색 패턴(diamond search)[7] 등이 있다. 이들은 초기 탐색 위치를 중심으로 정사각형 또는 다이아몬드 모양으로 회전하면서 탐색하는 방법이다. 다소의 성능 저하를 감수하면서 계산량을 대폭 줄일 목적으로 만들어진 고속 알고리즘으로는 3단계 탐색법(TSS : three-step search)[8], 4단계 탐색법(four-step search)[9], 2차원 대수 탐색법(2-dimensional logarithm search)[10] 등 다수가 있다. 그러나 후자의 고속 알고리즘과 같이 계산량을 크게 줄이는 방법은 FBMA에 대해 성능저하가 크고, 전자와 같이 성능저하가 없는 고속 알고리즘은 계산량을 줄이는 데 한계가 있다. 따라서 약간의 성능 저하를 허용하면서, 계산량에 있어서는 후자의 고속 알고리즘과 같거나 그 이상으로 줄임으로써 실시간 동영상 전송에 적합한 알고리즘의 연구가 필요하다.

본 논문에서는 초기 탐색 위치를 효율적으로 선택한 후, 탐색영역 내에서 회전 탐색하면서 SEA와 PDE를 사용하여 FBMA에 대하여 0.2dB 정도의 미소한 평균 PSNR 저하를 보이면서, 3% 이하의 평균 계산량이 소요되는 고속 알고리즘을 제안한다. 초기 탐색 위치는 이전 프레임과 현재 프레임을 2×2화소 블록 평균으로 부표본화한 영상으로부터 얻은 움직임 벡터를 원영상 크기로 확대한 벡터와 참조 블록의 움직임과 주위 블록들이 갖는 움직임 사이의 높은 상관성을 바탕으로 주위 블록들의 움직임에서 중간 값을 얻어 구한 예측 움직임 벡터 각각에 대응하는 위치 중에서 정합오차가 작은 것을 선택한다. 그리고 선택된 초기 탐색 위치를 중심으로 회전 탐색을 시작하여, SEA로 탐색할 후보 블록을 선택하고, PDE로 블록 간 정합오차의 계산량을 줄여 고속의 움직임 추정을 한다. 실험으로부터 제안된 방법에서 얻은 초기 탐색 위치가 FBMA에서 얻은 움직임 벡터에 해당하는 위치에 매우 근접함을 확인하였고, 제안된 알고리즘이 적은 성능저하를 발생하면서 계산량을 대폭 줄일 수 있음을 확인하였다.

Ⅱ. 블록 정합에 의한 움직임 추정

2.1 블록정합 알고리즘

블록정합 알고리즘에서는 그림 1과 같이 영상을 일정한 크기의 작은 블록 단위로 나눈 다음 현재 프레임(t프레임)의 참조 블록 X와 이전 프레임((t-1) 프레임)의 후보 블록 $Y_{i,j}$ 사이의 정합오차(또는 유사도)를 비교하여 최소값(또는 최대값)을 갖는 상대 위치를 찾아 물체의 움직임을 찾는다. 만약, 이동 변위가 (0,0)라면 X는 이전 프레임에 바로 대응되는 블록 $Y_{0,0}$와 가장 작은 정합오차를 가진다.

블록정합 알고리즘에서 현재 프레임의 참조 블록과 이전 프레임의 후보 블록 사이의 정합오차 계

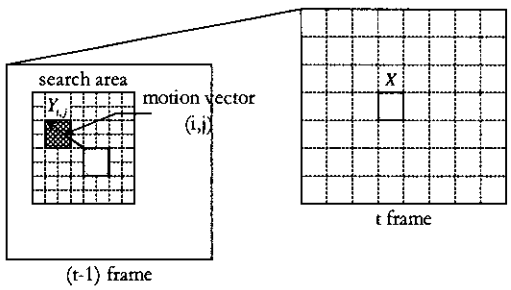


그림 1. 블록정합 알고리즘에서의 움직임 벡터 검출

산에는 일반적으로 다음과 같이 MAD(mean absolute difference)을 사용한다.

$$MAD(i,j) = \langle |X - Y_{i,j}| \rangle \quad (1)$$

(1)식에서 이동변위 (i,j)는 $-S \le i,j \le S$의 탐색 영역 내로 제한되고, $\langle \cdot \rangle$ 연산은 블록 X 및 Y에 관하여 다음과 같이 수행된다.

$$\langle |X - Y| \rangle = \frac{1}{B^2} \sum_{k=0}^{B-1} \sum_{l=0}^{B-1} |X(k,l) - Y(k,l)|$$

여기서 $X(k,l)$과 $Y(k,l)$은 각각 블록 X와 블록 Y의 (k,l) 위치 화소값을 나타내며, B는 블록의 가로 및 세로 길이를 나타낸다. 정합오차의 척도로서 MSE(mean square error)가 사용되기도 하지만, 복잡도 때문에 일반적으로 움직임 추정을 위한 척도로 (1)식을 사용한다.

2.2 연속소거 알고리즘

Li와 Salari에 의해 제안된 연속소거 알고리즘[4]은 현재 프레임의 참조 블록과 이전 프레임의 후보 블록들 각각의 평균을 구하고, 참조 블록 평균과 후보 블록 평균의 절대차를 구하여 초기 정합오차와 비교, 어떤 부등식을 만족하지 않는 후보 블록들을 제거해가며 최소 MAD를 갖는 움직임 벡터를 탐색한다.

연속소거 알고리즘에서 사용된 부등식[1]은 다음과 같다.

$$|\langle X \rangle - \langle Y_{i,j} \rangle| \le \langle |X - Y_{i,j}| \rangle \quad (2)$$

(2)식에서 우변은 (1)식의 $MAD(i,j)$를 나타내므로 (2)식은 다음과 같이 쓸 수 있다.

$$|\langle X \rangle - \langle Y_{i,j} \rangle| \le MAD(i,j) \quad (3)$$

탐색영역 내에서 이전까지 구한 $MAD(i,j)$ 중 최소인 값을 MAD_0라 할 때, 다음의 관계를 만족하는 후보 블록만이 최소 MAD를 가질 수 있으므로, 이 관계를 만족하는 후보 블록에 대해서만 탐색을 진행한다.

$$MAD(i,j) \le MAD_0 \quad (4)$$

즉, (3)식과 (4)식을 결합하여 얻은 다음의 관계를 만족하는 후보 블록만 탐색한다.

$$|\langle X \rangle - \langle Y_{i,j} \rangle| \le MAD_0 \quad (5)$$

Li와 Salari의 연속소거 알고리즘에서는 최소 MAD를 갖는 최적의 블록을 찾을 때, (5)식의 조건을 만족하는 후보 블록에 대해서만 탐색을 진행하기 때문에 FBMA에 대하여 PSNR을 저하시키지 않으면서, 계산량을 줄일 수 있다.

2.3 부분정합 왜곡소거법

부분정합 왜곡소거법[5]에서는 블록간 정합오차를 계산하는 과정에서 부분 정합오차가 이전까지의 최소 정합오차보다 크면, 그 블록에 대한 정합오차 계산을 그만두고 다음 후보 블록으로 옮겨 탐색을 계속한다. 이때, 부분 정합오차는 다음과 같이 주어진다.

$$PMAD_{i,j}(m,n) = \frac{1}{B^2} \sum_{k=0}^{m} \sum_{l=0}^{m} |X(k,l) - Y_{i,j}(k,l)| \quad (6)$$

(6)식은 참조 블록 X와 (i,j)의 움직임 벡터를 갖는 후보 블록 $Y_{i,j}$에 대하여 (m,n)위치 화소까지의 부분 정합오차를 나타내고, 각 블록에서 대응되는 위치의 화소 값을 절대차 누적시킨 것이므로 단조증가 함수이다. 따라서 다음과 같이 부분 정합오차가 이전까지 구한 최소 정합오차 MAD_0보다 크면, 이 후보 블록의 정합오차 $MAD(i,j)$는 MAD_0보다 항상 크다.

$$PMAD_{i,j}(m,n) \ge MAD_0 \quad (7)$$

그러므로 (7)식을 만족하는 후보 블록은 최적의 움직임 벡터를 가질 수 없다. 따라서 현재의 후보 블록을 버리고 다음 후보 블록에서 탐색을 계속한다. 부분 정합 왜곡 소거법의 실제 적용에 있어서, 각 화소의 위치마다 (7)식을 적용하여 판단을 하는

것은 부담이 되기 때문에 블록의 각 줄마다 (6)식의 부분 정합오차를 계산한 후에 (7)식을 적용하는 것이 유리하다.

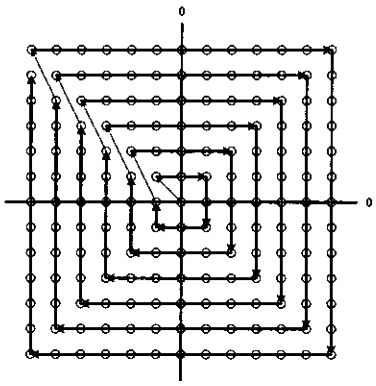


그림 2. 회전 탐색 패턴

2.4 회전 탐색 패턴

회전 탐색 패턴[6]은 움직임 벡터가 탐색영역의 중심에 많이 분포함을 이용하여 탐색영역의 중심으로부터 그림 2와 같이 정사각형 모양으로 회전하면서 후보 블록을 탐색한다. 이 때, 탐색영역의 중심이란 현재 프레임의 참조 블록과 대응되는 이전 프레임 후보 블록의 위치를 말한다. 즉, 움직임 벡터 (0,0)를 갖는 후보 블록의 위치이다. 최소 정합오차를 갖는 후보 블록의 위치가 탐색영역의 중심에 가깝게 있을 때, 회전 탐색 패턴을 연속소거 알고리즘이나 부분정합 왜곡소거법과 함께 사용하면, (5)식과 (7)식에서 MAD_0가 최소 정합오차에 빨리 근사해지고 부등식이 제한하는 범위를 더욱 줄여 계산량을 줄일 수 있다.

Ⅲ. 초기 탐색 위치 선택을 이용한 고속 움직임 추정

일반적으로 회전 탐색에서 초기 탐색 위치는 탐색영역의 중심 (0,0)벡터에 대응되는 위치를 사용한다. 그러나 움직임이 큰 영상의 경우에는 (0,0)벡터의 위치를 중심으로 탐색을 시작하는 회전 탐색은 최소 정합오차를 갖는 움직임 벡터를 찾는 데 많은 계산량이 요구된다. 그러므로 움직임을 예측하여 초기 탐색 위치를 적절히 선택하고, SEA나 PDE를 사용하면, 최소 정합오차를 갖는 움직임 벡터를 빨리 찾을 수 있으면서, (5)식과 (7)식이 제한하는 범위를 효과적으로 줄일 수 있으므로 계산량을 대폭 줄일 수 있다. 제안된 방법에서는 2×2화소 블록 평균으로 부표본화된 프레임으로부터 얻은 움직임 벡터와 주위 블록의 움직임을 이용한 예측 움직임 벡터 중에서 작은 정합오차를 갖는 것을 초기 탐색 위치로 선택하고, 이 위치를 시작으로 회전 탐색을 진행하여 움직임을 추정함으로써 계산량을 대폭 줄이면서 성능저하는 크지 않도록 하였다.

2.1 초기 탐색 위치의 선택

부표본화를 이용한 움직임 벡터의 선택은 그림 3과 같이 이루어진다. 먼저 현재 프레임과 이전 프레임을 2×2화소 블록 평균을 이용하여 수평 및 수직 방향으로 각각 2 : 1 부표본화 하여 얻은 영상에 대하여, 블록정합 알고리즘으로 각 블록의 움직임 벡터를 구한다. 이 때, 정합되는 블록의 크기 및 탐색 영역은 원영상의 것들에 비해 1/4로 축소된 상태이다. 탐색 순서는 회전 탐색 패턴으로 하며, 2화소 간격으로 후보 블록을 먼저 탐색한 후에 가장 작은 정합 오차를 갖는 후보 블록을 중심으로 나머지 후보 블록을 탐색하는데, 이러한 탐색 순서는 움직임이 큰 영상에 대하여 상당히 효과가 있으며, 후보 블록의 탐색에는 SEA와 PDE를 사용한다. 다음으로, 이렇게 얻은 부표본화 된 영상의 움직임 벡터를 두 배로 확대하여 초기 움직임 벡터 (x_1, y_1)를 얻는데, 이것은 원 블록의 대략적 움직임을 잘 나타내게 된다. 그러나 블록의 움직임이 작거나 복잡한 영상에 대해서는 잘못된 결과를 낼 수도 있다.

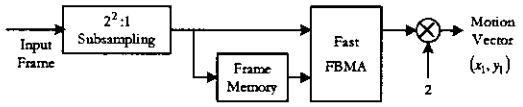


그림 3. 부표본화를 이용한 움직임 벡터 선택

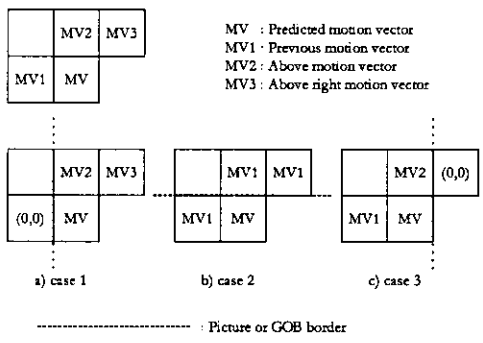


그림 4. 예측 움직임 벡터 선택을 위한 주위 움직임 벡터 결정[1]

한편, H.263 등의 동영상 부호화 표준에서는 주위 블록의 움직임을 이용한 예측 움직임 벡터[1]를 사용하는데, 이것은 참조 블록과 주위 블록의 움직임이 높은 상관성을 갖는다는 것을 바탕으로 하며, 주위 블록의 움직임 벡터에서 가운데 값을 선택하여 예측 움직임을 얻는다. 그림 4는 이때 사용되는 주위 블록의 움직임 벡터들을 나타낸다. 여기서 점선은 프레임(picture or GOB)의 경계를 나타내며, a), b), 그리고 c)는 각각 참조 블록이 프레임의 경계에 위치하여 주위 블록이 없는 경우에 주위 움직임 벡터를 선택하는 방법을 보여준다. 참조 블록의 예측 움직임 벡터 MV의 수직, 수평 성분은 각각 다음과 같이 구한다.

$$x_2 = Median(x_{MV1}, x_{MV2}, x_{MV3})$$
$$y_2 = Median(y_{MV1}, y_{MV2}, y_{MV3}) \quad (8)$$

여기서 x_{MVi}및 y_{MVi}는 주위 블록의 움직임 벡터 MV_i의 x및 y 성분을 나타낸다. 이렇게 얻어진 예측 움직임 벡터는 여러 블록들이 같거나 유사한 방향성을 갖는 경우에 매우 효과적이다. 그러나 움직이는 물체가 작아서 각 블록의 움직임간에 상관성이 적거나 움직임이 매우 큰 영상에서는 잘못된 예측을 하는 경우도 있다.

따라서 블록의 대략적 움직임을 잘 타나내는 부표본화 된 영상을 이용한 움직임 벡터와 주위 블록의 움직임을 이용한 예측 움직임 벡터를 모두 사용하여 초기 탐색 위치를 선택하는 것이 더 효과적이다. 제안된 방법에서는 위에서 얻은 두 가지 움직임 벡터들 중에서 다음과 같이 작은 정합오차를 갖는 움직임 벡터 (x, y)에 대응되는 위치를 초기 탐색 위치로 이용하였다. 여기서 (x_1, y_1)은 부표본화를 이용하여 얻은 움직임 벡터를, (x_2, y_2)는 주위 블록을 이용한 예측 움직임 벡터를 나타낸다.

$$(x, y) = \underset{i=1,2}{\arg\min}\ MAD(x_i, y_i) \quad (9)$$

그림 5는 10프레임/초의 연속 영상 STEFAN. QCIF 33개 프레임들에 대하여 FBMA로 얻은 움직임 벡터의 분포와 제안된 방법으로 얻은 초기 움직임 벡터(초기 탐색 위치)와 FBMA로 얻은 움직임 벡터 사이의 차벡터 분포를 나타낸다. 그림 5(a)는 FBMA로 얻은 움직임 벡터의 분포로서, x축은 움직임 벡터의 수평 성분 절대값, y축은 움직임 벡터의 수직 성분 절대값, 그리고 z축은 전체 매크로 블록 수에 대한 같은 움직임 벡터를 갖는 매크로 블록 수의 백분율[%]을 나타낸다. 그림 5(b)는 제안된 방법으로 얻은 초기 움직임 벡터와 FBMA로 얻은 움직임 벡터의 차벡터 분포로서, x축은 차벡터의 수평 성분 절대값, y축은 차벡터의 수직 성분 절대값, 그리고 z축은 전체 매크로 블록 수에 대한 같은 차벡터를 갖는 매크로 블록 수의 백분율[%]을 나타낸다. 그림 5에서 제안한 방법으로 얻은 초기 움직임 벡터들은 FBMA로 얻은 움직임 벡터들로부터 $[-2,+2]\times[-2,+2]$의 영역 내에 대부분(80% 이상) 분포하는데, 이것은 제안된 방법이 움직임 벡

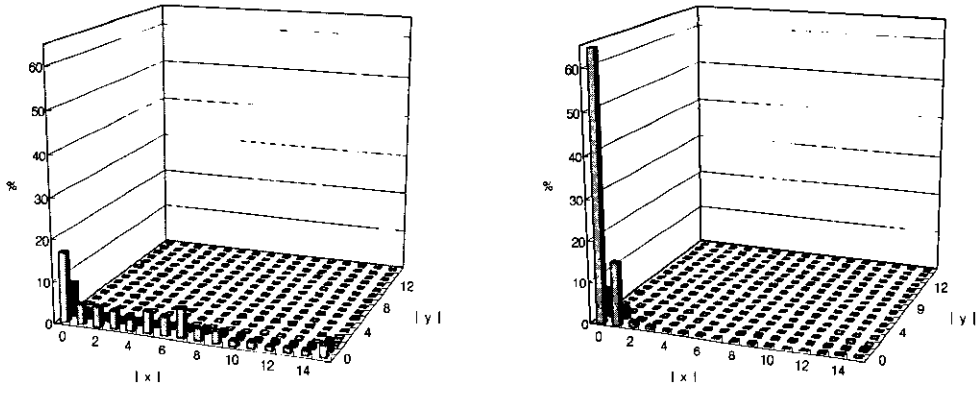


그림 5. 연속 영상 STEFAN에 대한 움직임 벡터 분포 :
(a) FBMA로 얻은 움직임 벡터 분포,
(b) 제안된 방법으로 얻은 초기 움직임 벡터와 FBMA로 얻은 움직임 벡터의 차벡터 분포

터에 상당히 근접한 초기 탐색 위치를 선택하는 방법으로 매우 효과적임을 나타낸다.

3.2 제안된 알고리즘의 흐름도

제안된 알고리즘에서는 그림 6과 같이 한 프레임에 대한 움직임 벡터 추정을 시작하기 전에 전처리 과정으로 후보 블록들의 평균 $\langle Y_{i,j}\rangle$을 구한다. 각 참조 블록에 대하여 블록 평균 $\langle X\rangle$을 구하고, 2×2 화소 블록 평균을 이용한 부표본화로 얻은 영상으로부터 움직임 벡터 (x_1, y_1)을 얻어 $MAD(i,j)$을 먼저 구한다. 그리고, 이웃 블록의 움직임을 이용하여 예측 움직임 벡터 (x_2, y_2)을 구하여 $MAD(x_2, y_2)$을 구하고, 움직임 벡터 (x_1, y_1)와 (x_2, y_2) 중에서 작은 정합 오차를 갖는 것을 선택하여 초기 움직임 벡터 (x, y)로 한다. 이렇게 얻은 초기 탐색 위치를 중심으로 회전 탐색을 시작하며, 이전까지 얻은 최소 정합오차 MAD_0는 $MAD(x, y)$로 한다. 회전 탐색 패턴으로 SEA를 이용하여 MAD_0를 포함하는 부등식으로 탐색할 후보 블록을 선택하고, 선택된 후보 블록과의 정합 오차를 계산하는 과정에서 블록내 각 줄에 대하여 PDE를 적용한다.

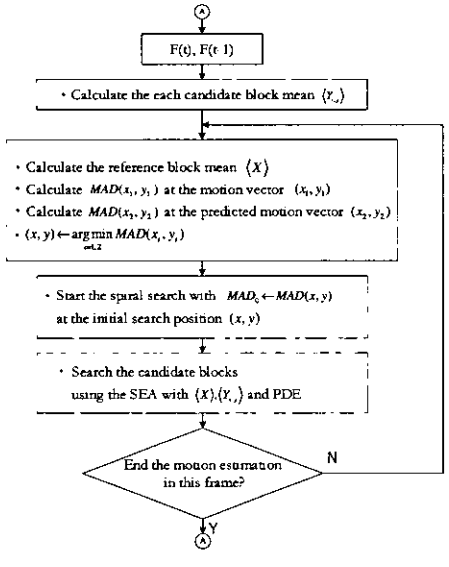


그림 6. 제안된 알고리즘의 흐름도

Ⅳ. 실험 결과 및 고찰

본 논문에서 제안된 방법과 기존 방법의 성능 평가를 위하여 설정된 실험환경은 표 1과 같다. 그리

표 1. 실험 환경

컴퓨터 사양	Pentium II 650MHz
사용 언어	Borland C++ Builder 4.0
프레임 크기	QCIF(176 × 144)
블록 크기	16×16 화소
탐색 범위	[-15,+15]×[-15,+15]
프레임 속도	10프레임/초
시험 연속 영상 당 처리된 프레임 수	33프레임

고 성능평가의 척도로 다음과 같이 원영상에 대한 보상된 영상의 PSNR을 사용하였고, 복잡도 평가 척도로 프레임 당 정합에 필요한 덧셈(또는 뺄셈)의 연산수와 절대치 연산수를 사용하였다.

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \tag{10}$$

여기서 MSE는 다음과 같이 주어지는 량이다.

$$MSE = \frac{1}{A\times B}\sum_{i=0}^{A-1}\sum_{j=0}^{B-1}\{X(i,j) - \hat{X}(i,j)\}^2 \tag{11}$$

(11)식에서 $X(i,j)$는 현재 프레임의 (i,j)위치의 화소값을, $\hat{X}(i,j)$는 움직임 추정 및 보상에 의하여 얻은 현재 프레임의 (i,j)위치 화소값을 나타내며, A와 B는 각각 프레임의 가로, 세로 화소 수를 나타낸다.

실험에서는 제안된 알고리즘을 다음의 두 가지 방식으로 적용하여 실험하였다. 첫 번째는 선택된 초기 탐색 위치를 중심으로 탐색 범위를 제한하여 평균적인 성능과 복잡도를 조절하는 방식이고, 두 번째는 매크로 블록 당 움직임 추정의 처리 속도가 일정 범위 내에 있도록 복잡도를 제한하는 방식이며, 이 두 방식은 제안된 방법으로 선택된 초기 탐색 위치가 FBMA에 의해서 얻어진 움직임 벡터에 매우 근접함을 바탕으로 한다. 초기 탐색 위치를 중심으로 탐색 범위를 제한하는 방식은 회전 탐색 패턴의 탐색 범위를 $[-S,+S]\times[-S,+S]$라고 할 때, S를 제한하여 복잡도와 성능을 조절할 수 있으며, 영상 부호화가 적용되는 프로그램, 혹은 전송 상태에 따라 S를 선택하여 적용할 수 있다. 각 프레임에 대하여 처리 속도가 일정하도록 하는 방식은 부표본화 된 영상에서 움직임 벡터를 얻기 위하여 소요되는 복잡도를 고려하여, 각 매크로 블록의 움직임을 추정하는 데 요구되는 복잡도를 일정하게

표 2. 각 움직임 추정 알고리즘의 평균 PSNR[dB]과 평균 복잡도 비교

Algorithms \ Image Sequences	CARPHONE.QCIF					STEFAN.QCIF				
	Average PSNR [dB]	Average Complexity				Average PSNR [dB]	Average Complexity			
		Additions		Absolutes			Additions		Absolutes	
		[10^5]	[%]	[10^5]	[%]		[10^5]	[%]	[10^5]	[%]
FBMA	31.53	395.7	100	198.2	100	22.40	395.7	100	198.2	100
SEA-PDE	31.53	27.6	7.0	13.4	6.8	22.40	95.8	24.2	47.6	24.0
TSS	31.06	27.7	7.0	13.9	7.0	21.81	27.8	7.0	13.9	7.0
Proposed Algorithm I ($S=6$)	31.51	13.9	3.5	6.3	3.2	22.37	26.6	6.7	12.7	6.4
Proposed Algorithm I ($S=1$)	31.41	5.5	1.4	2.0	1.0	22.33	9.7	2.4	4.1	2.1
Proposed Algorithm II ($N_{row}=250$)	31.51	9.3	2.4	3.8	1.9	22.25	10.7	2.7	4.5	2.3

함으로써 프레임별로 처리되는 속도가 일정하도록 하는 방식으로, 일정한 처리 속도를 요구하는 경우에 적합하다. 이때, 복잡도는 부분 정합 왜곡 소거법에 의하여 계산되는 블록의 줄 수 N_{row}을 제한함으로써 조절하였다.

표 2는 10프레임/초의 연속 영상 CARPHONE과 STEFAN에 대하여 FBMA와 PDE를 병행하는 SEA (SEA-PDE), TSS, 그리고 두 가지 방식으로 제안된 알고리즘의 실험 결과를 나타낸다. 여기서 제안된 알고리즘의 성능은 탐색 범위를 $S=6$, $S=1$로 제한한 알고리즘 I과 계산되는 블록의 줄 수 N_{row}을 250 이하로 제한한 알고리즘 II에 대하여 조사하였고, 복잡도는 각 알고리즘에 대하여 정합에 소요된 프레임당 평균 덧셈 연산수와 절대치 연산수, 그리고 각 연산수의 FBMA 연산수에 대한 백분율[%]로 나타내었다. 표 2에서 SEA-PDE은 평균 PSNR에 있어서 두 연속 영상 모두에 대하여 FBMA와 같은 평균 PSNR을 나타내고 있지만, 덧셈과 절대치 연산수 모두에 있어서 영상 CAR- PHONE은 FBMA 복잡도의 약 7%가 소요되고, 영상 STEFAN은 FBMA 복잡도의 약 24%가 소요되었다. TSS은 덧셈과 절대치 연산수 측면의 평균 복잡도 모두가 두 영상에 대하여 FBMA 복잡도의 7%가 소요되고, 평균 PSNR은 두 영상 모두에 대하여 FBMA보다 0.45dB 이상 저하되었다.

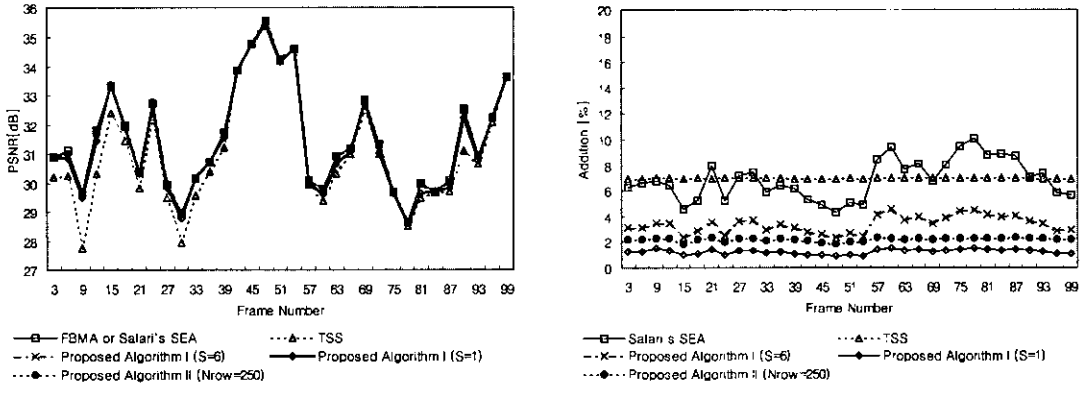
표 2의 실험 결과에서 $S=6$를 적용한 알고리즘 I은 FBMA에 대한 PSNR 저하가 두 영상 모두에 대해 0.03dB 이하였고, 덧셈과 절대치 연산수 모두에 있어서 영상 CARPHONE에 대하여 FBMA 복잡도의 약 3.3%가 소요되었고, 영상 STEFAN에 대하여 FBMA 복잡도의 약 6.5%가 소요되었는데, 이것은 TSS 복잡도보다 적어진 양이다. $S=1$로 탐색 범위를 더욱 줄이면, FBMA에 대한 평균 PSNR 저하가 두 영상 모두에 대해서 0.12dB 이하로 나타났고, 덧셈과 절대치 연산수 모두 영상 CARPHONE에 대해서 FBMA 복잡도의 약 1.2%로 감소하였고, 영상 STEFAN에 대해서 FBMA의 2.3%로 감소하였으며, 이것은 TSS 복잡도의 32% 이하로 줄어든 양이다. 실험 결과로부터 S를 줄이면, 평균 PSNR이 다소 저하되면서 평균 복잡도가 감소함을 알 수 있는데, S는 요구되는 PSNR과 복잡도에 따라서 선택적으로 사용할 수 있다. 그러나 탐색 범위를 제한하는 알고리즘에서 같은 탐색 범위를 적용하는 경우, 영상에 따라서 평균 복잡도가 배 이상 차이가 남을 알 수 있다. 이에 비하여, 매크로 블록 당 복잡도를 제한한 알고리즘 II는 표 2에서 FBMA에 대한 평균 PSNR 저하가 영상 CARPHONE에 대해서 0.02dB로 나타났고, 영상 STEFAN에 대해서 0.15dB로 나타났으며, 제안된 알고리즘의 덧셈과 절대치 연산수 측면의 평균 복잡도는 두 영상 모두에 대하여 FBMA 복잡도의 3% 이하, TSS 복잡도의 37% 이하로 감소하였다. 알고리즘 II의 실험 결과에서 영상 STEFAN에 대한 PSNR 저하가 $S=1$인 알고리즘 I의 실험 결과보다 커지기는 했지만, 두 영상에 대한 평균 복잡도 차이가 적어 졌음을 알 수 있는데, 이것은 일정한 처리 속도를 요구하는

경우에 적합하다. 표 2의 실험 결과로부터 각 알고리즘의 덧셈 연산수와 절대치 연산수의 FBMA에 대한 비율이 같음을 알 수 있다.

그림 7은 연속 영상 CARPHONE 33개 프레임들에 대하여 각 알고리즘에 대한 프레임별 실험 결과를 나타낸다. 그림 7(a)은 각 알고리즘에 대한 프레임별 PSNR을 나타내며, 수평축은 영상의 프레임 번호를, 수직축은 PSNR[dB]를 나타낸다. 그림 7(b)은 각 알고리즘의 프레임별 덧셈 연산수를 나타내며, 수평축은 영상의 프레임 번호를, 수직축은 FBMA 덧셈 연산수에 대한 각 알고리즘의 복잡도 백분율[%]을 나타낸다. SEA-PDE의 프레임별 PSNR은 그림 7(a)에서 보듯이 FBMA의 PSNR과 같고, 프레임별 덧셈 연산수는 그림 7(b)에서 보듯이 프레임에 따라서 FBMA 덧셈 연산수의 4%에서 10%까지 크게 다르다. TSS은 그림 7(a)에서 보듯이 각 프레임의 PSNR이 FBMA에 대하여 저하되는 정도가 프레임에 따라서 크게 다른데 최고 2dB까지 발생하지만, 덧셈 연산수는 모든 프레임에 대하여 FBMA 덧셈 연수사의 약 7%로 나타났다.

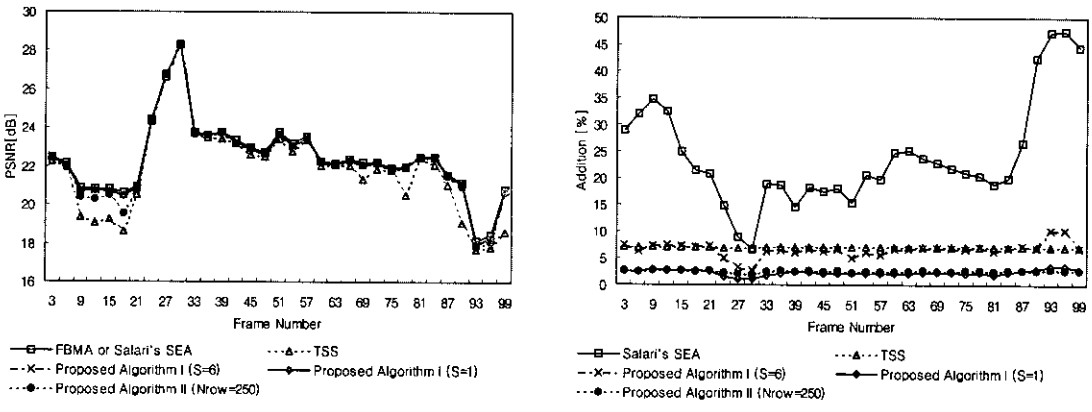
그림 7(a)에서 $S=6$인 알고리즘 I은 각 프레임의 PSNR이 FBMA에 대하여 0.1dB 이하로 미소하게 저하되면서, 그림 7(b)에서 보듯이 프레임별 덧셈 연산수는 FBMA 덧셈 연산수의 2.5%~4.5%로 감소하였고, 이것은 TSS 덧셈 연산수에 대하여 40%~65%로 줄어든 양이지만, TSS 덧셈 연산수의 25%에 해당하는 프레임별 변화를 나타내었다. 여기서 $S=1$로 탐색 영역을 더욱 줄이면, 각 프레임의 PSNR은 FBMA에 대하여 0.4dB 이하로 저하되면서, 그림 7(b)에서 보듯이 프레임별 덧셈 연산수는 FBMA 덧셈 연산수의 1%~1.5%, TSS 덧셈 연산수의 13%~21%로 줄었고, 프레임별 변화는 TSS 덧셈 연산수의 8% 정도로 나타났다. 알고리즘 II는 FBMA에 대한 PSNR 저하가 그림 7(a)에서 보듯이 0.15dB 이하로 나타나면서, 프레임별 덧셈 연산수는 FBMA 덧셈 연산수의 2%~2.5%, TSS 덧셈 연산수의 26%~33%로 감소하였고, 프레임별 변화도 TSS의 5% 정도로 감소함으로써, 알고리즘 I에 비하여 적은 복잡도 변화를 나타내었다. CARPHONE에 대한 실험에서, 각 알고리즘에 대한 절대치 연산수의 FBMA 절대치 연산수에 대한 비율은 표 2와 같이 각 프레임에 대해서도 덧셈 연산수에 대한 비율과 유사하여, 그림 7(b)와 같은 개형을 나타냈다.

그림 8은 연속 영상 STEFAN의 33개 프레임들에 대하여 각 알고리즘에 대한 프레임별 실험 결과를 나타낸다. 그림 8(a)은 각 알고리즘에 대한 프레임별 PSNR을 나타내며, 수평축은 영상의 프레임 번호를, 수직축은 PSNR[dB]를 나타낸다. 그림 8(b)은 각 알고리즘의 프레임별 덧셈 연산수를 나타내며, 수평축은 영상의 프레임 번호를, 수직축은 FBMA의 덧셈 연산수에 대한 각 알고리즘의 덧셈 연산수 백분율[%]을 나타낸다. SEA-PDE의 프레임별 PSNR은 그림 8(a)에서 보듯이 FBMA의 PSNR과 같고, 프레임별 덧셈 연산수는 그림 8(b)에서 보듯이 프레임에 따라서 크게 다른데, FBMA덧셈 연산수의 7%에서부터 47%까지 변화하였다. 이것은 일정 처리 속도를 요구하는 경우에 문제가 될 수 있다. TSS은 그림 8(a)에서 보듯이 각 프레임의



(a) (b)

그림 7. 연속 영상 CARPHONE에 대한 각 알고리즘의 프레임별 실험 결과 비교 :
(a) PSNR[dB] 비교, (b) 덧셈 연산수 비교



(a) (b)

그림 8. 연속 영상 STEFAN에 대한 각 알고리즘의 프레임별 실험 결과 비교 : (a) PSNR[dB] 비교, (b) 덧셈 연산수 비교

PSNR이 FBMA에 대하여 저하되는 정도가 프레임에 따라서 크게 달라, 최고 2dB까지 발생하지만, 덧셈 연산수는 모든 프레임에 대하여 FBMA 덧셈 연수사의 약 7%로 나타났다.

그림 8(a)에서 $S=6$인 알고리즘 I은 각 프레임의 PSNR이 대부분 FBMA에 대하여 0.15dB 이하로 미소하게 저하되면서, 그림 8(b)에서 보듯이 프레임별 덧셈 연산수는 FBMA 덧셈 연산수의 3%~10%가 소요되었고, 이것은 TSS 덧셈 연산수의 40%~140%가 소요된 것이며, 프레임별로 다소의 차이를 보였다. $S=1$로 탐색 범위를 더욱 제한하면, 그림 8(a)에서 보듯이 각 프레임의 PSNR이 대부분 FBMA에 대하여 0.25dB 이하로 저하되면서, 그림 8(b)에서 보듯이 프레임별 덧셈 연산수는 FBMA 덧셈 연산수의 1%~3.5%로 감소하였고, 이것은 TSS 덧셈 연산수의 14%~51%로 감소한 양이지만 프레임별 변화가 TSS의 25% 정도로 나타났다. 알고리즘 II은 프레임별 PSNR이 그림 8(a)에서 보듯이 9, 12, 18번째 프레임에 대하여 0.5dB 정도로 다소 저하되는 것을 제외하면, 대부분의 프레임에 대하여 0.2dB 이하로 미소하게 저하되면서, 그림 8(b)에 나타나듯이 프레임별 덧셈 연산수가 FBMA 덧셈 연산수의 2%~2.8%, TSS 덧셈 연산수의 27%~40%로 감소되었고, 프레임별 변화도 TSS의 13% 정도로 감소하여, 알고리즘 I에 비하여 일정한 처리 속도를 보였다. STEFAN에 대한 실험에서도 각 알고리즘에 대한 절대치 연산수의 FBMA 절대치 연산수에 대한 비율은 매 프레임마다 덧셈 연산수에 대한 비율과 유사하게 나타났다.

그림 7과 그림 8에서 탐색 범위를 제안하는 알고리즘이 영상에 따라서, 그리고 프레임에 따라서 크게 다른 복잡도를 나타냄을 알 수 있다. 영상 FOREMAN와 영상 SUZIE에 대하여 추가로 실험해 본 결과, $S=6$인 알고리즘 I은 네 가지 실험 영상에 대하여 평균 PSNR이 FBMA에 대하여 0.05dB 이하로 저하되면서, 평균 복잡도가 FBMA 복잡도의 3.4%~6.6%, TSS 복잡도의 48%~95%로 감소하였으나, 영상별 평균 복잡도 변화가 TSS 복잡도의 50% 정도로 커졌고, $S=1$인 알고리즘 I은 네 가지 실험 영상에 대하여 평균 PSNR이 FBMA에 대하여 0.16dB 이하로 저하되면서, 평균 복잡도가 FBMA의 1.3%~2.3%, TSS의 18%~33%로 감소하였으나, 영상별 평균 복잡도 변화가 TSS의 15% 정도로 나타났다. 이에 비하여 알고리즘 II는 네 가지 실험 영상에 대하여 평균 PSNR이 FBMA에 대하여 0.15dB 이하로 저하되면서, 평균 복잡도가 FBMA의 2.2%~2.6%, TSS의 31%~37%로 감소하였으며, 영상별 평균 복잡도 변화도 TSS의 6% 정도로 감소하여, 일정한 처리 속도를 요구하는 경우에 알고리즘 II가 적합함을 보였다.

표 3은 10프레임/초의 연속 영상 CARPHONE과 STEFAN에 대하여 각 알고리즘의 처리시간을 비교한 결과를 나타낸다. 여기서 복잡도는 각 알고리즘에 대하여 프레임당 평균 처리시간과 FBMA의 평균 처리시간에 대한 각 알고리즘의 평균처리 시간 백분율[%]로 나타내었다. 표 3의 실험 결과에서 FBMA 평균 처리시간에 대한 각 알고리즘의 처리시간 백분율은 표 2의 결과와 유사하게 나타났다.

표 3. 각 움직임 추정 알고리즘의 평균 CPU 시간 비교

Image Sequences / Algorithms	CARPHONE		STEFAN	
	[sec]	[%]	[sec]	[%]
FBMA	0.861	100	0.861	100
SEA-PDE	0.087	10.1	0.257	29.8
TSS	0.061	7.1	0.061	7.0
Proposed Algorithm I ($S=6$)	0.035	4.1	0.068	7.8
Proposed Algorithm I ($S=1$)	0.016	1.87	0.026	3.0
Proposed Algorithm II ($N_{row}=250$)	0.028	3.28	0.029	3.3

PDE-SEA나 제안된 알고리즘들이 표 2의 결과와 다소 차이를 보이는 것은 연산수의 비교에서 고려되지 못 한 비교문 등이 FBMA나 TSS에 비하여 많이 발생하기 때문이다.

V. 결 론

움직임 추정에 있어서, 기존에 제안된 SEA은 FBMA에 비하여 PSNR 저하가 없지만 계산량이 많은 경우가 있고, TSS은 계산량이 적어 실시간 영상 부호화에 적합하지만 FBMA에 비하여 PSNR 저하가 큰 경우가 있다. 본 논문에서는 부표본화를 이용하여 얻은 움직임 벡터와 주위 블록의 움직임을 이용한 예측 움직임 벡터를 이용하여 초기 탐색 위치를 효율적으로 선택함으로써, FBMA에 대하여 미소한 평균 FBMA를 나타내면서, 평균 복잡도를 TSS 복잡도보다 감소시키는 고속 알고리즘을 제안하였다. 제안된 알고리즘에서는 선택한 초기 탐색 위치를 중심으로 하여, 회전 탐색 패턴으로 SEA를 사용하여 후보 블록을 선택하고, 선택된 후보 블록과의 정합오차를 계산하는 과정에서 PDE를 사용하여 계산량을 대폭 줄였다.

실험에서 제안된 알고리즘은 선택된 초기 탐색 위치를 중심으로 탐색 범위를 제한하는 알고리즘 I과 매크로 블록 당 복잡도를 제한하는 알고리즘 II로 적용되었다. 알고리즘 I은 실험 결과에서 탐색 범위를 $S=6$로 하면, 평균 PSNR이 FBMA에 비하여 0.05dB 이하로 감소하면서, 평균 복잡도는 FBMA 복잡도의 7% 이하로 감소하여, TSS 복잡도보다 감소하였고, $S=1$로 탐색 범위를 더욱 제한하면, 평균 PSNR이 FBMA에 비하여 0.16dB 이하로 감소하면서, 평균 복잡도는 FBMA 복잡도의 2.5% 이하, TSS 복잡도의 35% 이하로 감소함을 확인하였다. 이 실험 결과에서 알고리즘 I이 S를 조절함에 따라, PSNR과 복잡도를 조절할 수 있음을 확인하였다. 알고리즘 II의 경우, 평균 PSNR이 FBMA에 비하여 0.15dB 이하로 감소하면서, 평균 복잡도는 FBMA 복잡도의 3% 이하, TSS 복잡도의 37% 이하로 감소하였으며, 실험 영상들의 프레임별 복잡도가 TSS의 30%~40% 정도로 나타났다. 그리고 알고리즘 II의 프레임별 복잡도 변화가 TSS의 10% 이하로 감소하여, 일정한 처리 속도를 요구하는 경우에 적합하게 이용될 수 있음을 확인하였다. 연산수에 의한 복잡도 비교뿐만 아니라, 처리 시간에 대한 실험에서도 유사한 결과를 얻었다.

참 고 문 헌

[1] ITU-T DRAFT H.263, "Video doding for low bitrate communication," May 1996.

[2] ISO-IEC JTC1/SC2/WG11, "Preliminary text for MPEG video coding standard," ISO, Aug. 1990.

[3] Y. H. Fok and O. Au, "A fast block matching algorithm in feature domain," *Proc. of IEEE Workshop on Visual Signal Processing and Communications*, Melbourne, pp. 199-202, Sep. 1993.

[4] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. on Image Processing*, vol. 4, no. 1, pp. 105-107, Jan. 1995.

[5] H. S. Wang and R. M. Mersereau, "Fast algorithms for the estimation of motion vectors," *IEEE Trans. on Image Processing*, vol. 8, no. 3, pp. 435-438, Mar. 1999.

[6] T. Zahariadis and D. Kalivas, "Fast algorithm for the estimation of block motion vectors," *Proc. ICECS on Electronics, Circuits, and Systems '96*, vol. 2, pp. 716 -719, 1996.

[7] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. on Image Processing*, vol. 9, no. 2, pp. 287 -290, Feb. 2000.

[8] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion

estimation," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 7, pp.429-433, Apr. 1977.

[9] L. M. Po and W. C. Ma, "A novel four-stop search algorithm for fast block motion estimation," *IEEE Trans. on Circuits Syst. video Technol.*, vol. 4, pp. 313-317, June 1996.

[10] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Commun.*, vol.COMM-29, pp. 1799-1808, Dec. 1981.

[11] D. M. Young and R. T. Gregory, *A Survey of Numerical Mathematics.* New York: Dover: pp.759-101, vol. 2, 1998.

남 수 영(Soo-Young Nam) 정회원

2000년 2월 : 경북대학교 전자공학과 (공학사)
2000년 3월~현재 : 경북대학교 전자공학과 석사과정
<주관심 분야> 영상처리, 영상통신

김 석 규(Seog-Gyu Kim) 정회원

1988년 2월 : 영남대학교 전자공학과 (공학사)
2000년 2월 : 경북대학교 산업대학원 통신공학전공 (공학석사)
2000년 3월~현재 : 대구서부공업고등학교 전자통신과
<주관심 분야> 영상처리, 영상압축,

임 채 환(Chae-Whan Lim) 정회원

1993년 2월 : 경북대학교 전자공학과 (공학사)
1995년 2월 : 경북대학교 전자공학과 (공학석사)
2000년 2월 : 경북대학교 전자공학과 (공학박사)
2000년 3월~현재 : 삼성전자 정보통신총괄 무선사업부
<주관심 분야> 영상처리, 영상분할, 영상압축, 컴퓨터 비전

김 남 철(Nam Chul Kim) 정회원

1978년 2월 : 서울대학교 전자공학과 (공학사)
1980년 2월 : 한국과학기술원 전기 및 전자공학과 (공학석사)
1984년 2월 : 한국과학기술원 전기 및 전자공학과 (공학박사)
1984년 3월~현재 : 경북대학교 전자전기공학부 교수
1990년 1월~1994년 12월 : 생산기술연구원 HDTV 신호처리분과위원
1994년 1월~1996년 12월 : 한국통신학회 편집위원
1994년 1월~1996년 12월 : 대한전자공학회 편집위원
1996년 1월~현재 : 한국방송공학회 편집위원,이사
1997년 1월~1998년 12월 : 한국통신학회 신호처리 연구회 위원장
<주관심 분야> 영상처리, 영상압축, 영상복원, 컴퓨터 비전