일반논문-07-12-3-08

세분화된 탐색 범위에서의 방향 지향적 전영역 고속 탐색 알고리즘

임 동 영[a)‡], 박 상 준[a)], 정 제 창[a)]

Direction-Oriented Fast Full Search Algorithm at the Divided Search Range

Dongyoung Lim[a)‡], Sang-Jun Park[a)], and Jechang Jeong[a)]

요 약

본 논문에서는 비디오 부호화의 움직임 예측에 사용되는 블록 정합 알고리즘의 계산량을 줄이는 고속 전영역 탐색 알고리즘을 제안한다. 블록 정합 알고리즘에서 사용되는 기존의 나선형 탐색 알고리즘은 탐색 영역의 중심에서 시작하여 탐색 지점을 화소 단위로 이동화면서 움직임 예측을 수행하기 때문에 움직임이 적은 영상에 적합하다. 본 논문에서는 움직임이 작은 영상 뿐 아니라 움직임이 많은 영상에서도 효율적인 움직임 예측을 하기 위해 다음과 같은 알고리즘을 제안한다. 먼저 초기 문턱 값을 계산함에 있어서 확장된 예측기를 사용하여 보다 최소값에 근사한 문턱값을 계산한다. 그리고 탐색영역을 블록으로 세분화 한 후 각 영역을 새로운 탐색 순서에 따라 움직임 예측을 수행하고 방향성에 따라 영역을 재분할한다. 재분할된 영역이 가지는 방향성에 따라 방향 지향적인 탐색 순서를 적용한다. 실험 결과에서 제안하는 알고리즘이 기존의 나선형 전영역 탐색 알고리즘에 비해 객관적인 화질의 열화 없이 계산량이 평균적으로 약 94% 감소하는 것을 확인할 수 있다.

ABSTRACT

We propose the fast full search algorithm that reduces the computational load of the block matching algorithm which is used for a motion estimation in the video coding. Since the conventional spiral search method starts searching at the center of the search window and then moves search point to estimate the motion vector pixel by pixel, it is good for the slow motion picture. However we proposed the efficient motion estimation method which is good for the fast and slow motion picture. Firstly, when finding the initial threshold value, we use the expanded predictor that can approximately calculate minimum threshold value. The proposed algorithm estimates the motion in the new search order after partitioning the search window and adapt the directional search order in the re-divided search window. At the result, we can check that the proposed algorithm reduces the computational load 94% in average compared to the conventional spiral full search algorithm without any loss of image quality.

Keywords : BMA, Full Search, Fast Algorithm, Motion Estimation, MSEA

Ⅰ. 서 론

블록 정합 알고리즘 (Block Matching Algorithm)은 주어진 탐색 영역에서 최소의 비용을 갖는 후보 블록을 찾는 알고리즘으로 영상의 시간적 중복성을 최소화하여 효율적인 비디오 부호화를 가능하게 한다. 블록 정합 알고리즘은 입력 영상을 고정된 크기의 매크로블록으로 나누고 이 매크로 블록에 가장 비슷한 블록을 이전 참조 영상에서 찾음으로써 움직임 벡터를 예측한다.

전영역 탐색 알고리즘 (Full Search Algorithm)은 블록 정

a) 한양대학교 전자통신컴퓨터공학과
Department of Electronics and Computer Engineering, Hanyang University
‡ 교신저자 : 임동영(limddong@gmail.com)
※본 연구는 서울시 산학연협력사업으로 구축된 서울 미래형콘텐츠컨버전스 클러스터 지원으로 수행되었습니다.

합 알고리즘의 가장 일반적인 형태로 탐색 영역의 모든 후보 블록의 비용을 계산하여 최소의 비용을 갖는 움직임 벡터를 구하는 알고리즘이다. 전영역 탐색 알고리즘은 최적의 움직임 벡터를 구하므로 비디오 부호화에 있어서 압축 효율을 극대화 하지만 탐색 영역내의 모든 후보 지점의 비용을 계산하므로 방대한 계산량을 동반하여 부호화기에 많은 부하를 발생시킨다.

이러한 문제점을 개선하기 위해서 지금까지 많은 고속 움직임 예측 알고리즘(Fast Motion Estimation Algorithm)들이 연구되어 왔다. 이들 고속 알고리즘들은 크게 두 그룹으로 나누어진다. 첫 번째 그룹은 영상의 화질 손상을 감안하여 예측을 하는 방식(Lossy Motion Estimation Algorithm)으로 대표적으로 TSS(Three Step Search)[1], DS(Diamond Search)[2], PMVFAST(Predictive Motion Vector Field Adaptive Search Technique)[3] 등과 같은 고속 움직임 예측 알고리즘이 이에 속한다. 이러한 고속 움직임 예측 알고리즘들은 탐색 영역내의 후보 블록들을 전부 탐색하지 않기 때문에 상당한 계산량 감소를 가져오지만 그에 따른 화질 저하를 초래한다.

두 번째 그룹은 영상의 화질 저하 없이 움직임 예측의 계산량을 줄이는 방식(Lossless Motion Estimation Algorithm)으로 SEA(Successive Elimination Algorithm)[4], MSEA(Multilevel Successive Elimination Algorithm)[5], PDE(Partial Distortion Elimination)[6] 알고리즘과 같은 고속 전영역 탐색(Fast Full Serch) 알고리즘들이 이에 속한다. 이러한 고속 전영역 탐색 알고리즘들은 탐색 영역 내에 있는 후보 블록들을 전부 탐색하기 때문에 전영역 탐색 알고리즘에 비해 화질 열화가 전혀 없으며 수학적인 방정식을 이용하여 불필요한 계산량을 줄임으로 속도 향상을 가져온다.

본 논문에서는 두 번째 그룹에 속하는 고속 전영역 탐색 알고리즘을 제안한다. 제안하는 알고리즘은 전영역 탐색 알고리즘이므로 영상의 객관적 화질의 손실 없이 계산량을 줄이는 것을 목표로 한다. 탐색 영역을 일정한 크기의 여러 영역으로 세분화 한 후 본 논문에서 제안하는 탐색 순서대로 블록 기반 탐색을 한다. 불가능한 후보 블록들을 초기에 제거하기 위해서 확장된 예측기를 사용하여 기존의 예측기보다 정확한 최소 정합 에러를 구한다. 그리고 주변의 움직임 정보를 이용하여 움직임의 방향성에 따라 영역을 재분할하고 각각의 영역에 속한 서브 블록의 탐색 순서는 그 영역이 가지는 방향성을 적용한 탐색 순서를 적용하여 화질의 열화 없이 계산량을 줄이는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 대표적인 고속 전영역 탐색 알고리즘에 대해 간단히 살펴본다. 3장에서는 영상의 방향적 특성을 이용하는 제안하는 알고리즘에 대해 자세히 살펴본다. 4장에서는 실험한 결과를 제시하고 5장에서 결론을 맺는다.

II. 기존의 고속 전영역 탐색 알고리즘

앞으로 설명하는 모든 알고리즘에서 후보 매크로 블록을 찾는 비용은 *SAD* (Sum of Absolute Difference) 를 이용한다.

1. SEA

SEA는 현재 프레임의 기준 매크로 블록의 화소의 합(Sum Norm)과 참조 프레임의 탐색 영역 내에 있는 후보 매크로 블록의 화소의 합의 차이를 구한 후, 문턱값과 비교하여 *SAD* 계산이 불필요한 후보를 미리 제거하는 알고리즘이다. 최초의 문턱값은 같은 위치에 있는 참조 프레임 내의 후보 매크로 블록과 기준 블록의 *SAD* 를 사용한다. 식 (1)은 SEA에서 이용하는 기본적인 수학 부등식을 나타낸다.

$$|a+b| \le |a| + |b| \tag{1}$$

이 식(1)을 블록의 *SAD* 와 블록 합에 적용하면 다음과 같이 표현할 수 있다.

$$|C_0 - R_0| \le SAD \tag{2}$$

식 (2)에서 C_0 와 R_0 는 각각 현재 매크로 블록의 블록합과 탐색 영역 내의 (m,n) 위치에 해당하는 후보 매크로 블록의 블록합을 나타내며 다음과 같다.

$$C_0 = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f_{cur}(x+u, y+v) \tag{3}$$

$$R_0 = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f_{ref}(x+m+u, y+n+v) \tag{4}$$

또한 식 (2)의 SAD 는 다음과 같다.

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \left| f_{cur}(i,j) - f_{ref}(i+x, j+y) \right| \tag{5}$$

여기서 $f_{cur}(i,j)$ 는 기준 매크로 블록의 화소값을 나타내며 $f_{ref}(i+x, j+y)$ 는 (x,y) 위치에서의 후보 매크로 블록의 화소값을 나타낸다.

SEA에서는 고속 계산 알고리즘(Fast Calculation Algorithm)을 사용하여 미리구해 놓은 C_0 와 R_0 를 이용하고 현재까지 구한 min SAD 와 비교한다. 여기서 min SAD (minmum SAD)는 탐색을 하면서 계산된 SAD 중 가장 작은 값을 나타낸다. $|C_0 - R_0|$ 의 값이 min SAD 보다 클 경우, 식 (2)를 만족하지 않으므로 현재 위치 (m,n) 에서는 SAD 계산을 하지 않고 다음 후보 매크로 블록을 찾는다. 그 반대의 경우, 즉, $|C_0 - R_0|$ 의 값이 min SAD 보다 작을 경우에는 현재 위치 (m,n) 에서 SAD 를 계산하고 이 값이 min SAD 보다 작으면 min SAD 에 갱신한다.

SEA 알고리즘은 다음과 같은 고속 계산 알고리즘을 이용하여 블록합 C_0 와 R_0 의 계산량을 줄인다. 그림 1은 참조 영상을 나타낸 것이며 각각의 정사각형은 하나의 화소를 나타낸다. 블록의 크기가 16x16일 때, 각 열의 첫 번째 화소 위치에서 가로 방향으로 16개의 화소의 합을 식(6)과 같이 계산한다.

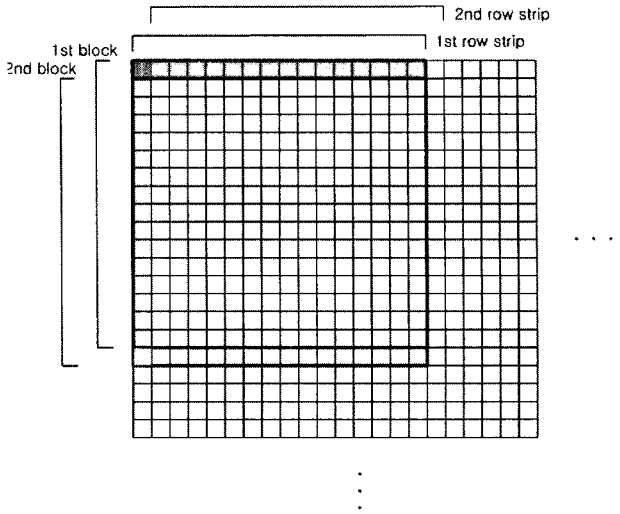


그림 1. 블록합 고속 계산 알고리즘
Fig. 1. Fast Calculation Method for Sum of Block

$$RowSum(x,y) = \sum_{i=0}^{15} f_{cur}(x+i, y) \tag{6}$$

$$\begin{aligned} RowSum(x,y) = {} & RowSum(x-1,y) \\ & - f_{cur}(x-1,y) + f_{cur}(x+15,y) \end{aligned} \tag{7}$$

여기서 $RowSum(x,y)$ 는 (x,y) 위치에서의 열의 합을 나타내며 $f_{cur}(x+i,y)$ 는 현재 프레임의 $(x+i,y)$ 위치에서의 화소값을 나타낸다.

두 번째 화소이후 위치에서는 식(7)을 이용한다. 앞서 구한 열의 합에서 첫 화소의 값을 빼주고 현재 열의 16개의 화소 중 마지막 화소의 값을 전 단계에서 구한 열의 합에 더해주어 다음 열의 합을 계산한다.

식 (7)에서 $RowSum(x,y)$ 과 $RowSum(x-1,y)$ 은 각각 (x,y) 와 $(x-1,y)$ 위치에서의 열의 합을 나타내며 $f_{cur}(x-1,y)$ 과 $f_{cur}(x+15,y)$ 는 현재 프레임에서 각각 $(x-1,y)$ 과 $(x+15,y)$ 위치에서의 화소값을 나타낸다.

블록합은 각 열의 합을 세로로 16개 더하여 구할 수 있다. 식 (8)과 (9)에 블록합의 계산식을 나타내었다. 여기서 $SumNorm(x,y)$ 은 (x,y) 위치에서의 블록합을 나타내며

$RowSum(x,y+i)$은 $(x,y+i)$ 위치에서의 열의 합을 나타낸다. 영상에서의 최상단에 위치하는 블록들의 블록합은 식 (8)을 이용하고 그 이외의 위치에서의 블록합은 열의 합을 구할 때 사용했던 방식을 적용한 식 (9)를 이용하여 구한다.

$$SumNorm(x,y)=\sum_{i=0}^{15}RowSum(x,y+i) \tag{8}$$

$$SumNorm(x,y)=\sum_{i=0}^{15}RowSum(x,y-1) -RowSum(x,y-1)+RowSu,(x,y+15) \tag{9}$$

2. MSEA

SEA를 세분화한 알고리즘인 MSEA는 매크로 블록을 정사각형의 서브 블록으로 나누어서 서브 블록의 합을 비교한다. 서브 블록의 합을 비교하게 되므로 식 (2)의 경계 조건은 더욱 세밀하게 되고 불가능한 후보 매크로 블록들을 많이 줄어들게 되므로 불필요한 SAD 계산량을 줄이게 된다.

$$\sum_{k=0}^{S_l-1}|C_l^{(k)}-R_l^{(k)}|\le\sum_{k=0}^{S_{l+1}-1}|C_{l+1}^{(k)}-R_{l+1}^{(k)}|\le SAD \tag{10}$$

식 (10)에서 l은 레벨(Level)을 의미하며 k는 해당 레벨에서의 서브 블록의 번호를 나타낸다. S_l은 레벨 l에 해당하는 서브 블록의 개수를 나타낸다. $C_l^{(k)}$ 와 $R_l^{(k)}$는 해당 레벨 l에서 각각 현재 매크로 블록과 후보 매크로 블록의 k 번째 서브 블록의 블록합을 의미한다. 식 (10)의 좌측은 레벨이 증가 할수록 현재 매크로 블록의 서브 블록과 후보 매크로 블록의 서브 블록 간 블록합의 차이의 절대 합이 증가함을 의미한다. 이는 경계 조건을 더욱 세밀하게 해주는 것을 의미한다. 그리고 매크로 블록에서 레벨이 4일 때 식(10)의 좌측에 있는 $\sum_{k=0}^{S_l-1}|C_l^{(k)}-R_l^{(k)}|$ 부분은 R 위치에서의 SAD를 의미한다.

앞서 소개한 SEA와 비교해 볼 때, SEA에서는 후보 매크로 블록 레벨이 하나만 존재하지만 MSEA에서는 후보 매크로 블록 레벨이 4가지 경우가 존재한다. MSEA에서는 여러 개의 레벨을 두고 후보 매크로 블록의 SAD 계산 여부를 식 (10)에 의하여 각 레벨 별로 (l=0,1,2,3) 판단한 뒤 계산한다.

MSEA는 계산량을 줄이기 위해 고속 계산 알고리즘을 사용한다. 먼저 레벨 3의 서브 블록 크기의 합을 계산하고, 레벨 2의 서브 블록의 합은 레벨3의 서브 블록을 4개 더하여 계산하는 방식을 이용한다.

3. PDE

PDE 알고리즘은 SAD계산시 불필요한 계산량을 줄이는 대표적인 알고리즘이다. 일반적인 블록 정합 알고리즘들이 블록의 SAD를 계산한 뒤 기존의 minSAD와 비교하는 반면 PDE 알고리즘은 부분적으로 계산된 SAD ($PartialSAD$) 와 min SAD를 비교한 후 $PartialSAD$가 min SAD 보다 더 크다면 남은 계산은 수행하지 않고 중단하는 방법이다. 식 (11)에 PDE의 대표적인 예인 행 단위로 SAD를 계산한 뒤 min SAD 와 비교하는 방법을 나타낸다.

$$PartialSAD(k)=\sum_{i=0}^{k}\sum_{j=0}^{N-1}|fcur(i,j)-fref(i+x,j+y)|, \quad k=0,1,...,N-1 \tag{11}$$

식 (11)에서 $PartialSAD(k)$는 k 번째 행까지 계산된 SAD를 의미하며 $k=N-1$이면 참조 영상의 탐색 영역에서 (x,y) 위치에서의 SAD가 된다. PDE 알고리즘은 다른 블록 병합 알고리즘과 동시에 쓰이는 것이 가능하며 전영역 탐색 및 다른 고속 탐색 알고리즘에서도 계산량을 줄이기 위해 사용된다.

Ⅲ. 제안하는 알고리즘

제안하는 알고리즘은 MSEA를 기반으로 하지만 탐색

영역을 세분화하고 방향 지향적인 탐색순서를 적용하여 전체적인 계산량을 줄인다. 그림 2는 제안하는 알고리즘의 블록 다이어그램을 나타낸다. 먼저 확장된 예측기를 사용하여 최소 정합 에러의 최소 근사치를 계산하고 분할 영역을 세분화 하여 각각의 방향성에 따른 탐색 순서를 적용하여 움직임 예측 알고리즘을 수행한다.

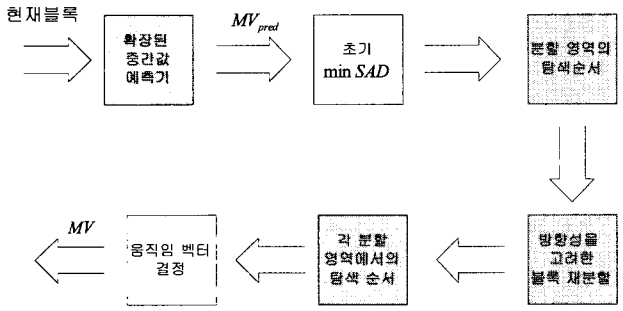


그림 2. MSEA기반 제안하는 알고리즘
Fig. 2. Proposed Algorithm based on MSEA

1. 확장된 중간값 예측기의 적용

고속 전영역 탐색 알고리즘에서 가장 중요한 것은 탐색 초기에 탐색 영역 내에서 최소 정합 에러를 찾는데 있다. 최소 정합 에러가 최소치에 근접할수록 SAD 계산량이 감소하여 최적화된 고속 전영역 탐색을 할 수 있다. 여기서 최소 정합 에러의 초기치를 최소화하기 위해 중간값 예측기(Median Predictor)를 사용한다.

이웃한 블록들 간에 움직임의 연관성이 존재하는 특징을 이용하면 현재 블록의 움직임 벡터를 효과적으로 예측하는 것이 가능하다. 최소 정합 에러는 현재 블록의 화소값과 좌측, 상단, 우상단 블록의 움직임 벡터의 중간값이 가리키는 블록의 화소값의 차이로 구한다. 여기서 한 프레임의 최상단의 블록들은 상단과 우상단 블록이 없으므로 전화면과 전전 화면의 움직임 벡터의 중간값을 사용한다.

하지만 중간값 예측기를 통해 계산된 화소에서의 최소 정합 에러는 최소치에 근사하지 않을 수 있다. 최소 정합 에러를 근사치에 비슷하게 하기 위해 중값값 예측기를 통해 예측한 벡터의 위치로부터 $N\times N$ 만큼 탐색 영역을 확장하여 움직임 탐색을 먼저 수행한다. 그림 3은 확장된 중간값 예측기를 나타낸다. 확장된 예측기는 기존의 예측기가 구한 초기 최소 정합 에러보다 최소에 근사한 최소 정합 에러를 가질 수 있는 가능한 후보들을 탐색할 뿐만 아니라 전체적인 계산량의 증가가 없기 때문에 확장된 예측기를 적용하여 계산된 최소 정합 에러를 적용하면 효율적으로 불가능한 후보들을 빨리 제거할 수 있다.

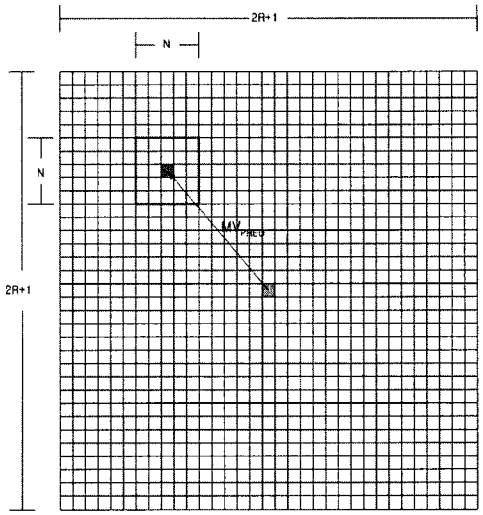


그림 3.확장된 중간값 예측기
Fig. 3. Expanded Median Predictor

2. 세분화된 탐색 영역을 이용한 고속 전영역 움직임 예측 알고리즘

움직임이 작은 영상일 경우 다수의 움직임 벡터가 (0,0)의 근처에 존재하므로 (0,0)을 시작으로 하는 기존의 나선형 탐색 알고리즘을 이용할 경우 효과적이다. 하지만 움직임이 많은 영상의 경우에는 (0,0)으로부터 움직임 벡터가 떨어져 있기 때문에 (0,0)을 시작으로 하는 나선형 탐색 알고리즘을 사용하면 계산 비용이 커지게 된다. 이를 개선하기 위해 다음과 같이 탐색 영역을 $N\times N$ 크기의 서브 블록으로 나누어 탐색한다.

그림 4와 같이 탐색 영역의 크기가 $\pm R$ 인 경우, 탐색 영역을 $N\times N$ 크기의 서브 블록 분할하면 총 $[(2R+1)/N]^2$

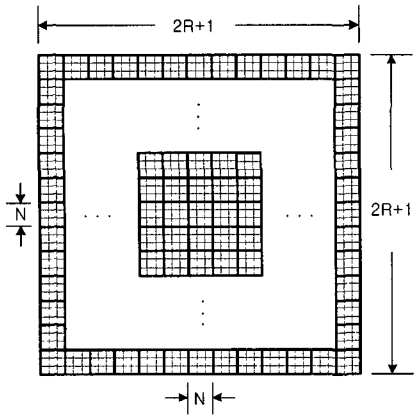


그림 4. 세분화된 탐색 영역
Fig. 4. Divided Search Range

개의 분할 영역으로 나뉜다. 우선 한 서브 블록 내의 모든 화소를 탐색한 후 서브 블록의 순서를 바꾸어 가면서 탐색을 한다. 기존의 탐색 알고리즘은 화소 단위로 탐색하는 반면 제안하는 탐색 알고리즘은 먼저 서브 블록의 탐색 순서를 기준으로 해당되는 서브 블록의 내부 화소를 모두 탐색한 후 다음 서브 블록의 화소를 탐색하는 방법을 사용한다. 탐색하는 서브 블록의 순서에는 여러 가지 방법이 있지만 본 논문에서는 다음 절에 제안하는 다이아몬드형 탐색 알고리즘을 사용한다. 이렇게 함으로써 가로 방향 움직임이 많은 영상의 경우 (0,0)을 시작으로 화소 단위로 탐색하는 나선형 탐색 알고리즘보다 빠르게 최소 정합 에러를 찾을 수 있다. 특히 가로 방향 움직임이 많은 영상의 경우 서브 블록으로 분할하여 탐색하는 알고리즘은 움직임 벡터가 (0,0)의 위치에서 어느 정도 떨어져 있기 때문에 (0,0)을 시작으로 하는 나선형 탐색 알고리즘보다 효과적으로 최소 정합 에러를 찾을 수 있다.

3. 영상의 특성을 적용한 탐색 순서

움직임 벡터가 (0,0)위치 부근에 있을 때에는 나선형 벡터가 효과적으로 움직임 예측을 수행한다. 하지만 움직임 벡터가 (0,0)위치 부근에 있지 않을 경우에는 최적의 움직임 벡터를 찾는데 많은 비용이 필요하게 된다. 그림 5는 여러 영상에서의 움직임 벡터 분포를 나타낸다.

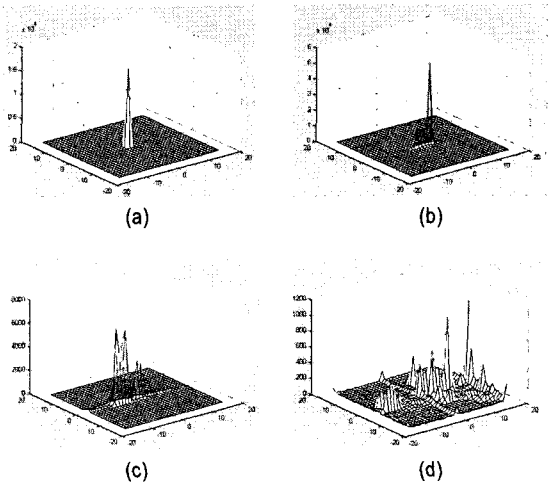


그림 5. 여러 영상에서의 움직임 벡터의 분포. (a) akiyo, (b) coastguard ,(c) container, (d) football
Fig. 5. Distribution of Motion Vector in Several Sequences. (a) akiyo (b) coastguard, (c) container, (d) football

그림 5를 보면 알 수 있듯이 움직임이 적은 영상인 'akiyo'의 경우에는 움직임 벡터가 (0,0)위치 부근에 밀집되어 있지만 움직임이 많은 영상인 'football' 의 경우에는 움직임 벡터가 (0,0)위치에서 떨어져 있기 때문에 (0,0)에서부터 탐색을 시작하는 나선형 탐색 알고리즘으로 탐색을 하면 (0,0)에서부터 멀리 떨어져있는 움직임 벡터를 찾기에 효과적이지 않다.

그림 5의 움직임 벡터의 분포를 통해 세로 방향 움직임보다 가로 방향 움직임이 더 많음을 알 수 있다. 사람의 움직임, 자동차의 이동 및 일반적인 사물의 움직임 등은 세로 방향의 움직임보다 가로 방향의 움직임이 많음을 알 수 있는데 이런 영상에서는 방향성 없이 중심에서부터 멀어지는 순서로 탐색하는 나선형 탐색 알고리즘을 적용하기보다는 가로 방향의 움직임 특성을 반영할 수 있는 그림 6(b)에 제안하는 다이아몬드형의 탐색 알고리즘을 적용하는 것이 효과적이다.

그림 6에 나선형과 다이아몬드형의 탐색 알고리즘을 나타내었다. 여러 개의 분할 영역들은 그림 6에 나타나 있는 탐색 순서대로 움직임 예측을 한다. 각 분할 영역 내에서는 두 가지의 탐색 순서를 적용할 수 있다.

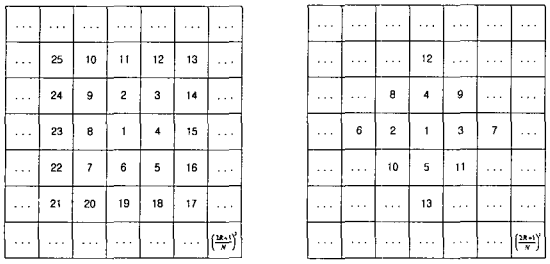


그림 6. 분할된 블록의 탐색 순서 (a) 나선형 (b) 다이아몬드형
Fig. 6. Search Order of Devided Block (a) Spiral Pattern (b) Diamond Pattern

일반적인 탐색 순서는 그림 6(a)와 같이 나선형으로 진행하면서 탐색하는 알고리즘이다. 하지만 앞에서 고찰하였듯이 영상에서 물체의 움직임은 대체로 가로 방향 움직임이 많기 때문에 본 논문에서는 가로 방향으로 탐색 영역을 넓힌 그림 6(b)의 다이아몬드형 탐색 알고리즘을 사용한다. 제안하는 다이아몬드형 탐색 알고리즘을 적용하면 기존의 탐색 알고리즘인 나선형 탐색 알고리즘보다 가로 방향으로 먼저 탐색을 하기 때문에 가로 방향 움직임이 많은 영상에서 보다 빠른 움직임 예측을 수행할 수 있다. 또 움직임이 가로 방향 움직임이 작은 영상을 예측할 경우를 이를 보완하기 위해 (0,0)을 중심으로 탐색하기 때문에 가로 방향 움직임이 적은 영상에도 적용하여 사용할 수 있다. 세로 방향 움직임이 많은 영상에 다이아몬드형 탐색 알고리즘을 적용할 경우 다이아몬드형을 세로 방향으로 적용하면 가로 방향 움직임이 많은 영상에서의 이점을 똑같이 얻을 수 있다.

4. 방향 지향적 탐색 순서

기존의 나선형 탐색 알고리즘은 방향성을 고려하지 않고 (0,0)의 위치에서부터 순차적으로 화소를 이동하면서 탐색한다. 만일 영상에 방향성이 존재한다면 방향성을 고려할 경우 후보 블록을 보다 빨리 탐색할 수 있으므로 방향성을 고려한 탐색 순서를 제안한다. 현재 블록의 위치에서의 방향성은 중간값 예측기의 값을 사용한다.

하나의 예로 중간값 예측기의 값이 그림 7과 같은 방향성을 나타낼 경우 서브 블록 내에서의 탐색 순서는 각 영역의 방향성을 고려하여 적용한다. 즉, 탐색 범위(R=16)안에서

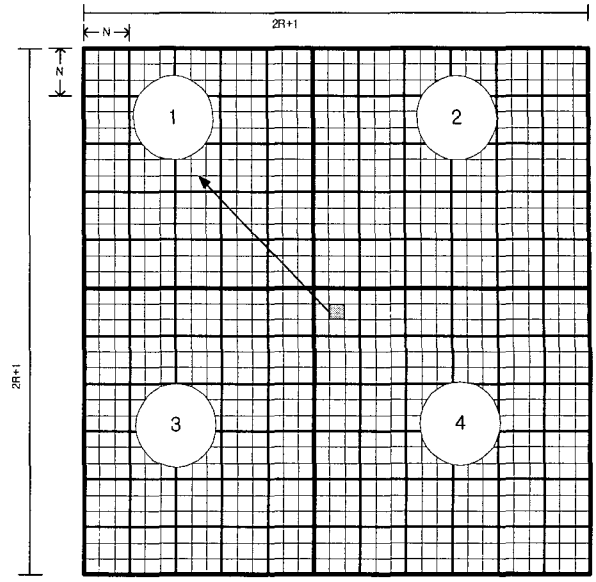


그림 7. 탐색 순서 결정을 위한 블록 재분할
Fig. 7. Redivided Block for Deciding Search Order

분할 영역(N=3)은 중간값 예측기가 나타내는 방향에 따라 그림 7과 같이 4가지 구역으로 재분할되고 각각의 재분할된 영역 안에서의 서브 블록은 그 영역이 가진 방향성에 따라 탐색 순서를 적용한다.

그림 8은 재분할된 각각의 영역에서의 탐색 순서를 나타낸다. 중간값 예측기가 나타내는 방향성이 그림 7과 같을 때 1번 영역과 4번 영역은 방향을 따라 좌상 단부터 탐색을 하고 2번 영역은 좌측부터, 3번 영역은 최상단부터 탐색을 한다.

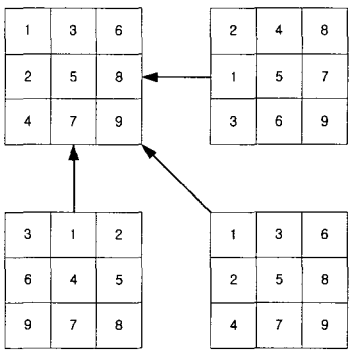


그림 8. 각 영역에서의 탐색 순서
Fig. 8. Search Order in the Each Region

Ⅳ. 실험 결과 및 분석

알고리즘의 성능을 확인하기 위해 CIF 크기의 30Hz 표준영상인 "Table"(300프레임), "akiyo"(300프레임), "bus"(150프레임), "coastguard"(300프레임), "container"(300프레임), "football"(90프레임), "foreman"(300프레임), "hall and monitor"(330프레임), "mother and daughter"(300프레임)를 알고리즘에 적용하였다. "akiyo"와 "mother and daughter"영상은 다른 영상에 비해 움직임이 작은 반면 "bus"와 "coastguard" 그리고 "container"는 가로 방향 성분의 움직임이 특히 많다. "football"과 "foreman"은 다른 영상에 비해 전체적으로 움직임이 많다.

실험결과는 객관적인 화질인 PSNR과 테스트 영상 전체 프레임에서 *SAD* 계산의 횟수, 다른 후보 블록을 탐색하는 횟수(Early Termination), 계산되는 전체 행의 횟수로 나타내었다. 제안하는 알고리즘은 탐색영역(R=16)과 서브 블록의 크기(N=3)를 고정하여 고속 전영역 탐색 알고리즘을 수행하였다.

1. 확장된 예측기

제안한 예측기의 성능을 알아보기 위해 모든 조건은 동일하게 하고 예측기의 종류에 변화를 주었다. 탐색 영역(R=16)의 분할된 영역(N=3) 내에서 예측기를 사용하지 않거나 기존의 예측기 및 확장된 예측기를 차례대로 적용하여 실험하였다. 기존의 예측기는 단순히 예측된 중간값만을 사용하여 예측하므로 최소 정합 에러를 계산함에 있어서 정확하지 않다.

따라서 확장된 예측기 사용에 따른 *SAD* 계산량을 통해 확장된 예측기가 기존의 예측기보다 정확함을 표 1을 통해 알 수 있다. 예측기를 사용 하지 않은 알고리즘에 비해 최대 67%까지 감소하고, 일반적으로 좌측, 상단, 우상단의 중간값을 사용하는 중간값 예측기에 비해 최대 0.35% 감소함을 알 수 있다. 이점으로 보아 확장된 예측기를 적용하여 구한 최소 정합 에러가 최소치에 보다 근사함을 알 수 있다.

표 1. 예측기 사용 여부에 따른 계산량
Table 1. Calculation Cost according to the Predictor

분류	예측기 적용안함	예측기	확장된 예측기	계산량 감소비	
				적용안함	예측기
akiyo	2242675	2183742	2180561	2.77	0.15
bus	15752990	5138848	5125792	67.46	0.25
coastguard	11591428	4831788	4826024	58.37	0.12
container	18395785	18382581	18380800	0.08	0.01
football	30016972	17947958	17893241	40.39	0.30
foreman	41436575	28598657	28536483	31.13	0.22
hall_monitor	63560379	63488516	63467252	0.15	0.03
mother and daughter	155866912	155134109	155095707	0.49	0.02
news	9872150	9389317	9366803	5.12	0.24
table	15540350	10848567	10810227	30.44	0.35

2. 방향 지향적 탐색 알고리즘

탐색 영역(R=16)의 분할된 영역(N=3)에서 방향 지향적 탐색 알고리즘의 성능을 비교한다. 그림 9에 나타나있는 세로축 수치는 각 행에서 *PartialSAD*의 값이 기존의 min*SAD* 보다 클 경우를 제거되는 행의 수를 나타내고 가로의 수치는 행을 나타내는데 제일 끝 행에 문턱값에 의해 제거되지 않고 *SAD*를 계산하는 행을 추가하여 나타내었다. 즉, *PartialSAD* 의 값이 min*SAD* 보다 크다면 불가능한 후보 블록이라 판단하게 되며 더 이상 계산하지 않고 다음 후보를 찾아서 예측을 시작하므로 불필요한 계산량을 줄일 수 있다. 제안하는 알고리즘은 전영역 탐색 알고리즘의 하나이므로 행의 수가 감소하여도 객관적인 화질의 기준인 PSNR은 변하지 않는다.

제안된 알고리즘의 경우 여러 개의 서브 블록으로 나누어 경계조건을 더 세밀하게 하였을 뿐만 아니라 MSEA 알고리즘 보다 정확한 최소 문턱값을 찾아서 적용하였기 때문에 불가능한 후보들이 많이 제거되어 계산되는 행의 수가 적은 것을 볼 수 있다.

영상의 움직임 특성을 적용하였기 때문에 가로 움직임이 많은 영상인 "bus", "football", "foreman"과 "table" 영상에서는 현저한 계산량의 감소를 볼 수 있다.

그림 9에 제시되어 있는 수치를 이용하여 각 알고리즘의

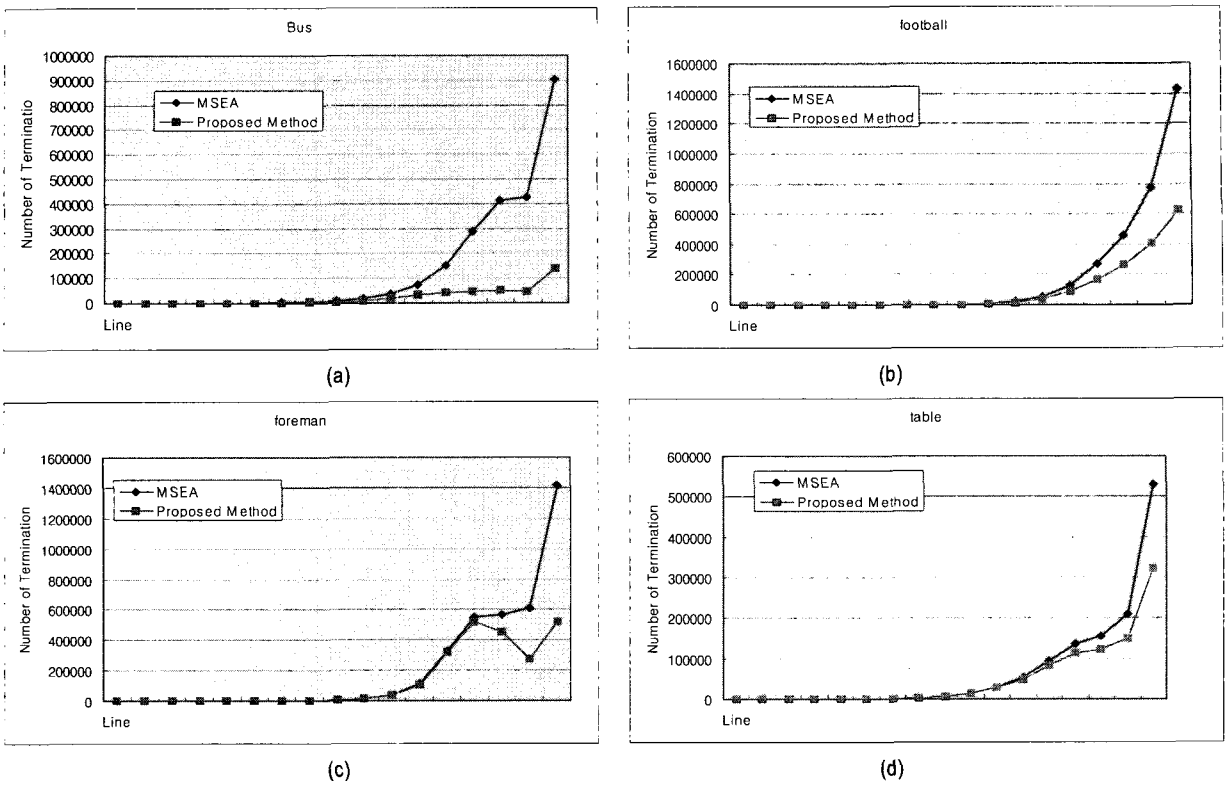


그림 9. 여러 영상의 각 행에서 제거되는 행의 수 (a) bus, (b) football, foreman, (d) table

Fig. 9. The Numbers of Row which is eliminated at Each line (a) bus, (b) football, (c) foreman, (d) table

성능을 비교해보기 위해 라인($Line$)마다 가중치를 주어 전체 계산량을 구한다. 다음 식 (12)를 이용하여 라인마다 가중치를 곱해주어 총계산량을 구한다.

$$\text{전체 계산량} = \sum_{w=1}^{16} w * Line(w) \qquad (12)$$

식 (12)에서 w는 각 행별 가중치를 의미하고, $Line(w)$는 w번째 행에서 제거되는 행의 수를 나타낸다. 16번째 제거되는 행의 경우에는 SAD의 계산량과 동일하므로 제거되지 않는 행의 수도 더해주어 가중치 계산을 한다. 표 2는 라인별 가중치를 두어 계산을 한 총 계산량을 나타낸다. 표 2를 보면 알 수 있듯이 제안하는 알고리즘이 기존의 나선형 탐색 알고리즘 보다 최대 99%만큼, MSEA 알고리즘 보다 최대 82%만큼 계산량이 감소함을 알 수 있다.

표 2. 가중치를 적용한 전체 계산량

Table. 2. Total Cost of Weighted Calculation

	나선형 탐색 알고리즘	MSEA	제안하는 탐색순서	계산량의 감소비율	
				나선형 탐색 알고리즘	MSEA
akiyo	199683508	2224547	2179069	98.91	2.04
bus	323290143	28293908	5032396	98.44	82.21
coastguard	551432409	9944412	4810476	99.13	51.63
container	451222067	18712665	18367112	95.93	1.85
football	310307693	36114054	17989925	94.20	50.19
foreman	599426633	40521568	28562601	95.24	29.51
hall and monitor	997629965	63509084	63462947	93.64	0.07
mother and daughter	593049162	156099347	155118079	73.84	0.63
news	280560266	10024907	9306789	96.68	7.16
table	502154211	15097892	10819792	97.85	28.34

V. 결 론

본 논문에서는 탐색 영역을 $N \times N$ 개의 분할 영역으로 나눈 후 재분할하여 방향 지향적 탐색 알고리즘을 수행하였다. 이렇게 함으로써 기존의 탐색 알고리즘이 가지고 있는 국소적인 탐색 특성을 개선하였다. 가로 방향 움직임이 적은 영상은 물론 가로 방향 움직임이 많은 영상에서 객관적인 화질의 열화 없이 계산량을 확연히 줄일 수 있었다. 확장된 예측기를 사용하여 최적의 최소 정합 에러를 구하였고 기존의 행 단위 PDE알고리즘을 적용하여 불가능한 후보 블록을 빨리 제거함으로써 불필요한 계산량이 줄어들어 예측을 보다 효율적으로 수행할 수 있음을 검증하였다. 영상에서의 움직임 특성을 반영한 탐색 영역내의 탐색 순서뿐만 아니라 탐색 영역을 재분할하여 방향 지향적인 탐색 알고리즘을 제안함으로써 최소 정합 에러를 구하는데 필요한 계산량을 기존의 나선형 전영역 탐색 알고리즘과 비교하였을 때 최대 99%까지 줄일 수 있었다.

참 고 문 헌

[1] R. Li, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. On Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 438-42, Aug'94.

[2] Shan Zhu and Kai-Kuang Ma, "A New Diamond Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. On Image Processing, vol. 9, No. 2, pp. 287 - 290, Feb. 2000.

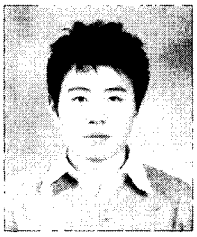
[3] A.M. Tourapis, O.C. Au, and M.L. Liou, "Fast Block-Matching Motion Estimation using Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)," in ISO/IEC/ JTC1/SC29 /WG11 MPEG2000/M5866, Noordwijkerhout, NL, Mar'00.

[4] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," IEEE Trans. Image Processing, vol. 4, pp. 105 - 107, Jan. 1995.

[5] X,Q. Gao, C.J. Duanmu, and C.R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," IEEE Trans. Image Processing, vol. 9, pp. 501 - 504, Mar. 2000.

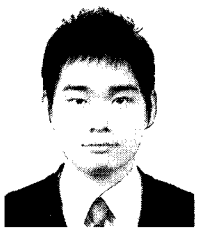
[6] 김종남, "영상 복잡도와 다양한 정합 스캔을 이용한 고속 전영역 움직임 예측 알고리즘," 정보과학회논문지 : 소프트웨어 및 응용 제 32권 제 10호, pp. 949 - 955, Oct. 2005.sd

저 자 소 개



임 동 영

- 2006년 2월 : 한양대학교 전자전기컴퓨터공학부 졸업
- 2006년 3월~현재 : 한양대학교 전자통신컴퓨터공학과 석사 과정
- 주관심분야 : Image Processing, Image Compression, Image Enhancement, H.264, MVC



박 상 준

- 2006년 2월 : 한양대학교 전자컴퓨터공학부 졸업
- 2006년 3월~현재 : 한양대학교 전자통신컴퓨터공학과 석사 과정
- 주관심분야 : 영상 처리, 화질 개선, 영상 압축, SVC

저 자 소 개

정 제 창

- 1980년 2월 : 서울대학교 전자공학과 졸업
- 1982년 2월 : KAIST 전기전자 공학과 석사
- 1990년 : 미국 미시간대학 전기 공학과 공학박사
- 1980년~1986년 : KBS 기술연구소 연구원(디지털 TV 및 뉴미디어 연구)
- 1990년~1991년 : 미국 미시간대학 전기공학과 연구교수 (영상 및 신호처리 연구)
- 1991년~1995년 : 삼성전자 멀티미디어 연구소 (MPEG, HDTV, 멀티미디어 연구)
- 1995년~현재 : 한양대학교 전자통신컴퓨터공학과 교수 (영상통신 및 신호처리 연구실)
- 1998년 11월 27일 : 과학기술자상 수상
- 1998년 12월 31일 : 정보통신부장관상 표창
- 주관심분야 : 영상처리 및 영상압축