# 세분화된 탐색 영역을 이용한 고속 전영역 움직임 예측 알고리즘

준회원 박 상 준*, 진 순 종*, 정회원 정 제 창*

# A Fast Full Search Motion Estimation Algorithm using Partitioned Search Window

Sang-Jun Park*, Soonjong Jin* *Associate Members*, Jechang Jeong* *Regular Member*

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 방법은 탐색 영역의 중심에서 시작하여 탐색 지점을 화소 단위로 이동하면서 움직임 예측을 수행 하기 때문에 움직임이 적은 영상에 적합하다. 본 논문에서는 탐색 영역을 작은 영역으로 세분화한 후 각 영역을 새로운 탐색 순서에 따라 움직임 예측을 수행함으로써 움직임이 많은 영상을 효과적으로 탐색할 수 있다. 또한 움직임 벡터 판정시 영상의 복잡도에 따라 최적의 순서로 비용을 계산하여 복잡도를 줄이는 방법을 제안한다. 실험 결과에서 제안하는 알고리즘이 기존의 나선형 전역 탐색 방법에 비해 예측화질의 열화 없이 최대 99%까지 계산량을 감소시키는 것을 확인할 수있다.

Key Words : Motion Estimation, Fast Full Search, Block Matching Algorithm, Partitioned Search Window, MSEA

ABSTRACT

We propose the fast full search algorithm that reduces the computation of the block matching algorithm which is used for motion estimation of 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 vectorpixel by pixel, it is goodfor the slow motion pictures. However the proposed method is good for the fast and slow motion because it estimates the motion in the new search order after partitioning the search window. Also, when finding the motion vector, this paper presents the method that reduces the complexity by computing the matching error in the order which is determined by local image complexity. The proposed algorithm reduces the computation up to 99% for block matching error compared with the conventional spiral full search algorithm without any loss of image quality.

## Ⅰ. 서 론

블록 정합 알고리즘(Block Matching Algorithm)은 주어진 탐색 영역에서 최소의 비용을 갖는 후보 블록을 찾는 방법으로 영상의 시간적 중복성을 최소화하여 효율적인 비디오 부호화를 가능하게 한다. 블록 정합 알고리즘은 입력 영상을 고정된 크기의 매크로블록으로 나누고 이 매크로블록에 가장 비슷

※ 본 연구는 서울시 산학연 협력사업으로 구축된 서울 미래형 콘텐츠 컨버전스 클러스터 지원으로 수행되었습니다.
* 한양대학교 전자통신컴퓨터공학과 영상통신및신호처리 연구실(sjp21c@ece.hanyang.ac.kr)
논문번호 : KICS2006-09-367, 접수일자 : 2006년 9월 01일, 최종논문접수통보일자 : 2006년 12월 21일

한 블록을 이전 참조 영상에서 찾음으로써 움직임 벡터를 예측한다. 일반적으로 비디오 부호화기에서 매크로블록의 크기를 16×16으로 정하는데, 블록의 크기가 이보다 크면 움직임 벡터의 수가 줄어들어 계산량은 줄어들지만 세밀한 영역을 제대로 예측하지 못하게 되어 예측 에러가 증가하게 된다. 반대로 블록의 크기가 이보다 작으면 세밀한 영역을 예측할 수가 있으므로 예측 에러는 줄어들지만 움직임 벡터의 개수가 증가하여 계산량이 증가하게 된다.

전역 탐색 방법(Full Search Algorithm)은 블록 정합 알고리즘의 가장 일반적인 형태로 탐색 영역의 모든 후보 블록의 비용을 계산하여 최소의 비용을 갖는 움직임 벡터를 구하는 방법이다. 전역 탐색 방법은 최적의 움직임 벡터를 구하므로 비디오 부호화에 있어서 압축 효율을 극대화하지만 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 손실을 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 영역 내의 후보 블록들을 전부 탐색하지 않기 때문에 상당한 계산량 감소를 가져오지만 그에 따른 화질 저하도 초래한다.

두 번째 그룹은 영상의 화질 저하 없이 움직임 예측의 연산량을 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 Search) 알고리즘들이 이에 속한다. 이러한 고속 전영역 탐색 알고리즘들은 탐색 영역 내에 있는 후보 블록들을 전부 탐색 하기 때문에 전역 탐색 방법에 비해 화질 열화가 전혀 없으며 수학적인 방정식을 이용하여 불필요한 계산량을 줄임으로 속도 향상을 가져온다.

본 논문에서는 두 번째 그룹에 속하는 고속 전영역 탐색 알고리즘을 제안하며, 따라서 영상의 화질 손실 없이 계산량을 줄이는 것을 목표로 한다. 탐색 영역을 일정한 크기의 여러 영역으로 나눈 뒤 제안하는 탐색 순서대로 기존의 MSEA를 사용한다. 그리고 주변의 움직임 정보를 이용하여 현재 블록의 움직임을 예측하므로 기존의 알고리즘보다 계산량을 줄일 수 있으며 또한 움직임 벡터 판정시 영상의 복잡도에 따라 최적의 순서로 비용을 계산하여 복잡도를 줄이는 방법을 제안한다.

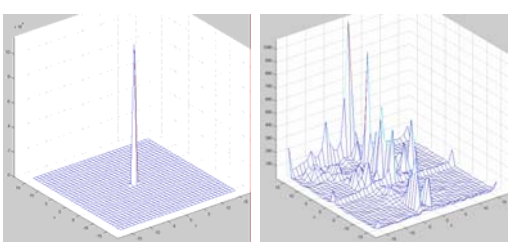
## Ⅱ. 적응적인 탐색 순서

기존의 고속 전영역 탐색 알고리즘에서 많이 사용하는 나선형 탐색 방법은 탐색 영역의 중심에서 시작하여 탐색 지점을 나선형으로 이동하면서 움직임 예측을 수행한다. 이러한 나선형 탐색 방법은 최적의 움직임 벡터가 (0,0) 위치 부근에 있을 때 효과적으로 움직임 예측을 수행한다. 즉, 그림 1(a)와 같이 움직임 벡터가 존재하는 영상에 적합하다.

"Akiyo" 영상의 경우 움직임이 적기 때문에 움직임 벡터가 (0,0) 주위에 몰려 있는 것을 볼 수 있다. 이러한 경우 기존의 나선형 탐색 방법을 이용함으로써 매우 적은 비용으로 움직임 예측을 할 수 있다. 그러나 모든 영상의 움직임 벡터가 그림 1(a)와 같이 분포하지 않는다. "Akiyo" 영상과 달리 "football Ⅰ" 영상과 같이 움직임이 많은 영상의 움직임 벡터의 분포를 그림 1(b)에 나타낸다.

"football Ⅰ" 영상의 경우 그림 1(b)에서 볼 수 있듯이 움직임 벡터가 (0,0)으로부터 멀리 떨어져 있는 부분에 많이 분포하는 것을 알 수 있다. 이렇게 움직임이 많은 영상의 경우, 기존의 나선형 탐색 방법을 사용하면 움직임을 예측을 하는데 많은 비용이 든다.

"football Ⅰ" 영상과 같이 움직임이 많은 영상의 경우 다수의 움직임 벡터가 (0,0)으로부터 멀리 떨어져서 존재하므로 탐색 영역을 여러 개의 영역으로 분할하여 탐색을 하는 것이 더 효율적이다. 그



(a) akiyo (b) football Ⅰ

그림 1. 영상의 움직임 벡터 분포

이유는 기존의 나선형 탐색 방법은 (0,0)을 시작으로 화소 단위로 탐색을 하기 때문에 움직임이 많은 영상에서는 움직임 벡터를 찾는데 비용이 커지는 반면에, 탐색 영역을 여러 개의 분할영역으로 나누어서 탐색을 하면 (0,0)으로부터 멀리 떨어진 움직임 벡터를 적은 비용으로 찾는 것이 가능해지기 때문이다.

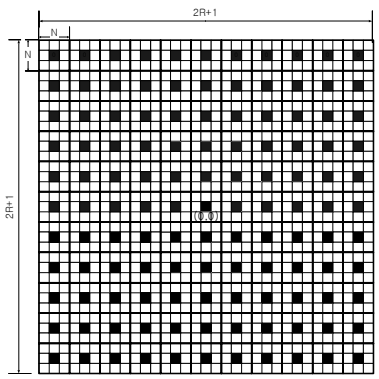



그림 2. 탐색 영역을 여러 개의 영역으로 분할한 그림

그림 2와 같이 탐색 영역의 크기가 $\pm R$인 경우, $N\times N$ 크기의 $[(2R+1)/N]^2$개의 분할 영역으로 탐색 영역을 분할 할 수 있다. 그림 2에서 흑색점은 각각의 분할 영역(partitioned region)에서 탐색이 시작이 되는 점이다. 기존의 나선형 탐색 순서로 하는 움직임 예측 방법은 그림 2에서의 중앙점(0,0)을 시작으로 한 화소씩 나선형으로 이동하면서 예측을 하지만 제안하는 방법은 여러 개의 분할된 영역 단위로 그림 3에 제시되어 있는 순서대로 이동하면서 MSEA를 이용하여 움직임 예측을 수행한다. 이렇게 함으로써 움직임이 많은 영상에서 보다 빠르게 최소 정합에러를 찾을 수 있으며 결과적으로 SAD 계산량을 현저히 줄일 수 있다.

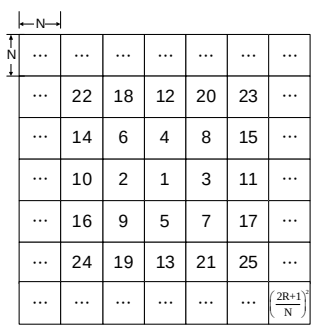
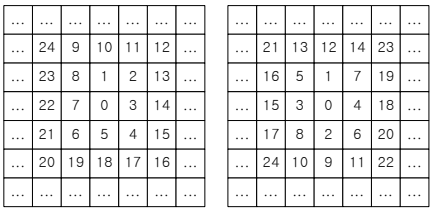



그림 3. 분할 영역들의 탐색 순서




(a) 나선형 (b) 십자형

그림 4. 분할 영역 내($N\times N$)에서 2가지 탐색 순서

여러 개의 분할 영역들은 그림 3에 나타나 있는 탐색순서대로 움직임 예측을 한다. 일반적인 영상이 가로, 세로 방향으로 움직임이 많은 점을 착안하여 그림 3에 나타나 있는 탐색순서를 제안한다.

각 분할 영역 내에서는 두 가지의 탐색 순서를 적용할 수 있다. 첫 번째 탐색 순서는 그림 4(a) 와 같이 나선형으로 탐색하는 것이고 두 번째 탐색 순서는 그림 4(b)와 같이 위, 아래, 왼쪽, 오른쪽, 좌상단, 우하단, 우상단, 좌하단 순으로 한다. 일반적인 영상인 경우에 가로 방향과 세로 방향의 움직임이 많기 때문에 제안하는 방법에서는 그림 4(b)의 탐색 순서를 사용한다.

## Ⅲ. 움직임 벡터 예측기 및 영상의 복잡도 적용

### 3.1 중간값 예측기 적용

앞서 언급 했듯이 고속 전영역 탐색 알고리즘에서 가장 중요한 것은 최대한 탐색 영역내에서 초기에 최소 정합 에러를 찾아서 SAD 계산량을 줄이는데 있다. 여기서 초기에 최대한 빠르게 최소 정합 에러를 찾기 위해서 움직임 벡터를 예측하였으며 중간값 예측기(median predictor) 를 사용한다.

이웃한 블록들간에는 움직임의 연관성이 존재하며 이를 이용하여 현재 블록의 움직임 벡터를 효과적으로 예측하는 것이 가능하다. 각각의 블록들은 하나의 움직임 벡터를 갖고 있으며 예측되는 현재 블록의 움직임 벡터는 바로 전 블록, 윗 블록, 우상단 블록의 움직임 벡터의 중간값으로 설정한다. 그림 5는 참조하는 움직임 벡터들에 대한 공간적인 위치를 나타낸다.


그림 5. 중간값 예측기가 참조하는 움직임 벡터들

### 3.2 영상의 복잡도를 이용한 적응적인 비용 계산 순서

후보 위치에서 정합 에러를 계산할 경우 정합 에러가 큰 영역부터 비용을 계산 하면 좀 더 불필요한 정합 에러 계산을 피할 수 있다. 따라서 영상의 복잡도를 계산하고 복잡도가 큰 영역부터 비용을 계산하기 위해 현재 부호화할 매크로블록을 16개의 4×4 크기의 서브 블록으로 나누어서 블록의 복잡도를 계산한 후 부호화할 매크로블록의 복잡도의 크기 순서대로 후보 위치에서의 정합 에러를 계산한다.

$$H=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}[X]\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}/2 \quad (1)$$

식 (1)과 같이 하다마드 변환(Hardamard transform)은 덧셈과 뺄셈만으로 구성되므로 계산 과정이 간단하고 영상을 주파수 성분으로 잘 분해하는 성능을 가지므로 효율적이다. 식 (1)에서 $X$는 4×4 크기의 입력 영상을, $H$는 입력 영상 $X$에 해당하는 하다마드 계수들을 의미한다. 즉 각각의 서브 블록들을 하다마드 변환하면 그림 6과 같이 나타낼 수 있고 변환을 통해 얻은 변환 계수들은 해당하는 서브 블록영상의 복잡도와 상당한 관계가 있기 때문에 하다마드 변환 계수를 이용하여 서브 블록 영상의 복잡도를 판별할 수 있다.

식 (1)에 의해 구해진 AC 계수들의 절대 합(sum of absolute AC coefficient)을 *SumAC* 라고 정의 하였다. 또한 그림 6에서 볼 수 있듯이 각각의 서브 블록의 DC 계수 16개를 구할 수 있으며 전체 DC 계수의 평균을 *AverageDC* 라고 정의한다. 또한 각각의 서브 블록의 DC 계수를 *SubDC* 라고 정의 할 때

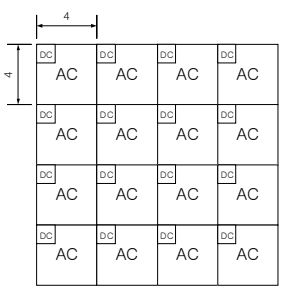



그림 6. 하다마드 변환이 적용된 서브 블록들

최종 i번째 서브블록의 복잡도는 식 (2)와 같다.

$$\text{Complexity of Sub-block(i)} = \text{SumAC(i)} + |\text{AverageDC} - \text{SubDC(i)}| \quad \text{where } i = 0,1,...,15 \quad (2)$$

식 (2)에 의해 서브 블록의 복잡도를 각각 계산한 후 복잡도가 큰 순서대로 *PartialSAD* 를 계산한다. 제안된 방법을 이용하여 그림 7(b)와 같은 방법, 즉 적응적인 순서로 비용을 계산하므로 적은 비용으로 움직임 예측을 할 수 있다.

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

(a)

| | | | |
|---|---|---|---|
| 1 | 9 | 11 | 8 |
| 14 | 2 | 6 | 12 |
| 0 | 7 | 13 | 4 |
| 5 | 10 | 3 | 15 |

(b)

그림 7. 서브 블록내에서의 정합 스캔 순서
(a)기존의 순차적 정합 스캔 순서
(b)복잡도를 이용한 정합 스캔 순서

## Ⅳ. 실험 결과 및 분석

알고리즘의 성능을 확인하기 위해 CIF 크기의 30Hz 표준 영상인 "foreman"(300 프레임), "mobile"(300 프레임), "football Ⅰ"(90 프레임), "football Ⅱ"(150 프레임), "coastguard"(300 프레임)을 가지고 실험을 하였다. "football I" 과 "football II", 그리고 "foreman" 영상은 다른 영상에 비해 움직임이 많으며 "coastguard" 영상은 가로 방향으로 움직임이 특히 많다. 그리고 "mobile" 영상은 다른 영상들에 비해 상대적으로 움직임이 적다. 실험 결과는 PSNR과 테스트 영상 전체 프레임에서 계산되는 전체 행의 횟수와 MSEA와 비교한 계산량의 감소비율로 나타내었다. 제안하는 방법은 기존의 33×33 크기의 탐색 영역을 121개의 3×3 블록으로 나누어서 고속 전영역 탐색 알고리즘을 수행하였으며 블록 크기는 16×16으로 하였다.

### 4.1 적응적인 탐색 순서

먼저 앞에서 제안한 적응적인 탐색 순서 알고리즘의 성능 평가를 하였다. 모든 조건은 동일하게 하고 탐색 순서만 바꾸어 기존의 알고리즘들(나선형 전영역 탐색 알고리즘(spiral FS), 나선형 MSEA)과 비교한다. 우선 영상의 전체 프레임에 대하여 16가지 Line에 해당하는 수치값을 구하는데, 이 수치값

은 행 단위 PDE 알고리즘을 사용하는 과정에서 각 행에서의 *PartialSAD* 와 기존의 min *SAD* 를 비교하였을 때 *PartialSAD* 값이 큰 경우의 횟수를 나타낸다. 예를 들어, Line 5의 경우, 블록 내에서 5번째 행까지의 *PartialSAD* 가 min *SAD* 보다 큰 경우의 횟수를 나타낸다. 또한, *PartialSAD* 가 min *SAD* 보다 크게 되면 더 이상 비용 계산을 할 필요가 없어진다. 즉, 불가능한 후보 블록이라고 판단하게 되며 작은 행(Line1, Line2,…)에서 제거될수록 불필요한 계산량을 더 줄일 수 있다. 추가적으로, 16번째 Line의 값은 블록 전체의 SAD 계산 횟수와 동일하다.

3가지 알고리즘들은 탐색 영역의 중심(0,0)에서 초기 min *SAD* 값을 갖고 움직임 예측을 한다. 단, 다른 점은 나선형 전영역 탐색 방법과 MSEA는 탐색 영역내에서 기존의 나선형으로 전영역을 탐색하지만 제안하는 적응적인 탐색 순서 방법은 탐색 영역을 세분화하여 제안하는 순서대로 탐색을 한다. 즉, 모든 조건이 같은 상황에서 탐색 영역을 세분화하고 제안하는 순서대로 탐색을 함으로써 계산량을 줄일 수 있다.

표 1의 수치값들은 식 (3)을 이용하여 1~16까지의 Line 크기를 가중치로 하여 각 Line에 해당하는 수치값을 곱하여 누적한 값을 나타낸다.

식 (3)에서 $i$ 는 블록 내에서 $i$ 번째 행, 즉 Line

$$\text{전체 행 계산 횟수} = \sum_{i=1}^{16} i \times Line\ i \qquad (3)$$

표 1. 각 알고리즘의 전체 행 계산 횟수 비교 및 제안된 알고리즘이 얻는 계산량의 감소비(%)

| | 나선형 전영역 탐색방법 | MSEA | 적응적인 탐색순서 | 계산량의 감소비율 |
|---|---|---|---|---|
| foreman | 599714633 | 45021568 | 31143469 | 30.825 |
| mobile | 501842547 | 8800578 | 4236667 | 51.859 |
| football Ⅰ | 310307693 | 36113844 | 24204394 | 32.978 |
| football Ⅱ | 325303153 | 12001549 | 6976097 | 41.873 |
| coastgurad | 551432409 | 12365211 | 5667729 | 54.164 |

의 크기를 의미한다. 식 (3)을 통해 테스트 영상의 모든 프레임에서 계산 되어지는 전체 행의 횟수를 얻을 수 있다. 이 값이 작을수록 계산량이 줄어드는 것을 의미하며 본 논문에서는 이 값을 계산량의 척도로 사용한다. 표 1에서 보듯이 모든 조건은 같고 탐색순서만 변경한 경우, 모든 영상에서 계산량이 현저히 감소하는 것을 볼 수 있다.

“foreman”, “football Ⅰ”과 같은 움직임이 많은 영상은 움직임 벡터가 탐색 영역의 중심에서 멀리 존재할 가능성이 높기 때문에 기존의 나선형 탐색 방법을 사용하면 불필요한 후보 블록의 SAD 계산을 하게 되어 계산량이 증가한다. 반면에 제안된 적응적인 탐색 순서 알고리즘은 분할된 영역별로 탐색을 하기 때문에 보다 넓은 영역을 좀 더 빨리 탐색함으로써 이러한 문제점을 해결 한다.

표 1에서 모든 조건이 같고 탐색 순서만 바꾼 경우 모든 영상에서 기존의 나선형 전영역 탐색 방법과 MSEA에 비해 계산량이 감소하는 것을 볼 수 있다. 표 1에서 계산량의 감소비는 기존의 MSEA와 비교하였을 때 제안하는 알고리즘이 갖는 계산량의 감소비를 의미하며 영상에 따라 31%~54%까지 계산량의 감소를 얻었다. 표 1을 통해 기존의 MSEA에서 탐색 순서를 변경함으로써 계산되는 전체 행의 횟수가 줄어드는 것을 알 수 있다.

### 4.2 중간값 예측기 적용

기존의 MSEA와 중간값 예측기를 적용한 MSEA의 성능을 비교 하였다.

표 2에서 볼 수 있듯이 중간값 예측기를 사용함으로써 최적의 초기 min *SAD* 값을 구하는 것이 가능하며 이 값을 이용하여 많은 수의 불가능한 후보 블록들을 일찍 제거할 수 있다. 그 이유는 PDE 알고리즘을 사용할 때 비교하는 초기 min *SAD* 값이 원래의 min *SAD* 값과 비슷하기 때문이다. 그러므로 모든 영상에서 중간값 예측기를 적용할 경우 전체 행 계산량이 감소하는 것을 볼 수 있으며 영상에 따라 25~61% 까지 계산량이 감소하였다.

표 2. 기존의 MSEA와 중간값 예측기를 적용한 MSEA의 전체 행 계산 횟수의 비교 및 감소비(%)

| | MSEA | 중간값 예측기 적용 | 계산량의 감소비 |
|---|---|---|---|
| foreman | 45021568 | 29145540 | 35.263 |
| mobile | 8800578 | 6029362 | 31.489 |
| football Ⅰ | 36113844 | 20066313 | 44.436 |
| football Ⅱ | 12001549 | 9024996 | 24.801 |
| coastgurad | 12365211 | 4825932 | 60.972 |

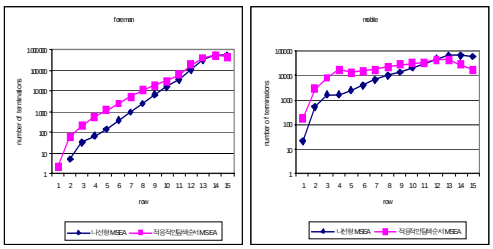
### 4.3 영상의 복잡도 이용

기존의 MSEA와 MSEA에 영상의 복잡도를 이용하는 알고리즘을 적용하여 실험 하였다.

표 3을 보면 영상의 복잡도를 이용한 MSEA가 모든 영상에서 기존의 MSEA 보다 적은 계산량을 얻는 것을 알 수 있다. 영상의 복잡도를 이용하여 복잡한 서브 블록 순서대로 탐색을 함으로써 불가

표 3. 기존의 MSEA와 영상의 복잡도를 이용한 MSEA의 전체 서브 블록 계산량의 비교 및 감소비(%)

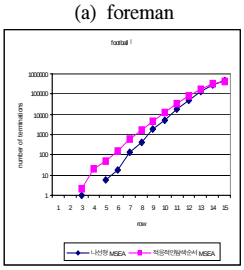
| | MSEA | 영상의 복잡도 이용 | 계산량의 감소비 |
|---|---|---|---|
| foreman | 45021568 | 30141270 | 37.167 |
| mobile | 8800578 | 3419448 | 61.145 |
| football Ⅰ | 36113844 | 23625441 | 34.581 |
| football Ⅱ | 12001549 | 6547171 | 45.447 |
| coastgurad | 12365211 | 5342462 | 56.794 |



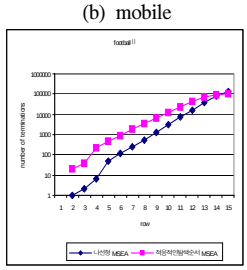

(a) foreman



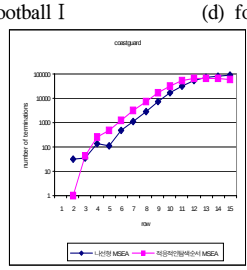

(b) mobile




(c) football Ⅰ




(d) football Ⅱ




(e) coastguard

그림 8. 영상의 각 행에서 제거되는 행의 수

능한 후보 블록들을 일찍 제거 할 수 있기 때문이다. 표 3에서 전체 서브 블록의 계산량이 영상에 따라 35~61% 감소 하는 것을 볼 수 있다.

그림 8(a)~(e)는 행의 수에 따라 제거되는 후보 블록의 수를 로그 스케일(Log scale)로 나타낸다. 그림 8(a)~(e)에서 보듯이 영상의 복잡도를 적용한 알고리즘이 기존의 MSEA 보다 불가능한 후보 블록들을 빨리 제거하는 것을 알 수 있다. 즉, 크기가 작은 행에서 다수의 후보 블록들이 일찍 제거되며, 크기가 큰 행에서는 다른 알고리즘들에 비해 적은 수의 후보블록들이 제거된다. 이렇게 됨으로써 전체 계산량을 줄일 수 있다.

지금까지 실험한 3가지 방법을 정리하면, 적응적인 탐색 순서 방법은 움직임 벡터 예측기를 사용하지 않았으며 초기 $\min SAD$ 값을 탐색 영역의 중앙점에서의 SAD값으로 한다. 즉 기존의 MSEA와 모든 조건을 같게 하고 탐색 영역을 여러개의 분할된 영역으로 나누어서 탐색 순서만을 다르게 한 것이다. 중간값 예측기 적용 방법은 기존의 MSEA에 중간값 예측기를 적용함으로써 최적의 초기 $\min SAD$ 값을 구할 수 있다. 영상의 복잡도를 이용하는 방법은 기존의 MSEA에 영상의 복잡도를 이용하여 복잡한 서브 블록부터 SAD 계산을 하는 알고리즘이며 영상의 복잡도를 이용함으로써 불필요한 계산량을 일찍 줄일 수 있었다.

### 4.4 최종적으로 실험 결과 비교

표 4에서는 나선형 전영역 탐색 방법을 기준으로 하여 나선형 MSEA와 제안하는 알고리즘이 수행하는 전체 행 계산량을 비교하였다. 표 4에서 제안하는 알고리즘은 적응적인 탐색 순서에 중간값 예측기와 영상의 복잡도를 기존의 MSEA에 적용한 알고리즘을 의미한다. 제안하는 알고리즘이 전체 계산량을 가장 많이 감소시키는 것을 알 수 있으며 기존의 나선형 전영역 탐색 방법에 비해 모든 영상에서 92% 이상의 계산량를 감소 시킨다. 표 4에서 보듯이 움직임이 많은 "football Ⅰ" 영상과 "foreman" 영상의 경우에 제안하는 알고리즘이 다른 영상에 비해 더 좋은 성능을 보이는 것을 알 수 있다. 이것은 제안하는 알고리즘이 움직임이 많은 영상에 더 적합하다는 것을 의미한다. 또한, "mobile" 영상과 "coastguard" 영상의 경우 제안하는 알고리즘이 전체 행 계산량을 99% 이상 감소 시키는 것을 볼 수 있다.

표 4. 나선형 전영역 탐색 방법과 전체 행 계산량 비교(%)

| | 나선형 전영역 탐색방법 | 나선형 MSEA | 제안하는 알고리즘 |
|---|---|---|---|
| foreman | 100.00 | 7.51 | 5.03 |
| mobile | 100.00 | 1.75 | 0.68 |
| football Ⅰ | 100.00 | 11.64 | 7.61 |
| football Ⅱ | 100.00 | 3.69 | 2.01 |
| coastgurad | 100.00 | 2.24 | 0.97 |

## Ⅴ. 결 론

본 논문에서는 탐색 영역을 여러 개의 $N \times N$ 분할 영역으로 나누어서 고속 전영역 탐색을 수행하였다. 이렇게 함으로써 기존 알고리즘의 국소적인 특성을 개선하는 것이 가능해졌다. 여러 개의 블록

단위로 탐색 하는 것이 화소 단위로 탐색하는 것보다 큰 간격으로 탐색을 하기 때문에 움직임이 많은 영상에 적용을 하면 화질 열하 없이 계산량을 확연히 줄일 수 있었으며 이것은 실험을 통해 움직임이 많은 영상에 대해 더 많은 양의 계산량이 줄어드는 것을 통해 검증되었다. 중간값 예측기를 이용하여 최적의 초기 최소 비용을 구하였으며 기존의 행단위 PDE 알고리즘 대신 영상의 복잡도를 고려하여 복잡한 영역순으로 PDE 알고리즘을 수행함으로써 불가능한 후보 블록들을 일찍 제거할 수 있었고 이로 인하여 불필요한 계산량을 최대한 줄일 수 있었다. 또한, 탐색 영역 내에서 분할된 영역들의 탐색 순서와 각각의 분할 영역 내에서의 탐색 순서를 제안함으로써 최소 정합 에러를 구하는데 필요한 계산량을 기존의 나선형 전영역 탐색 방법과 비교하였을 때 제안하는 알고리즘은 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.



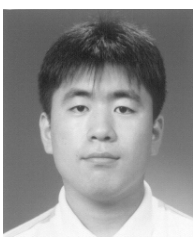
박 상 준 (Sang-Jun Park) 준회원

2006년 2월 한양대학교 전자컴퓨터공학부 졸업

2006년 3월~현재 한양대학교 전자통신컴퓨터공학과 석사 과정

<관심분야> 영상 처리, 화질 개선, 영상 압축, SVC

진 순 종 (Soonjong Jin) 준회원



2004년 2월 한양대학교 전자컴퓨터공학부 졸업

2006년 3월 한양대학교 전자통신컴퓨터공학과 석사

2006년 3월~현재 한양대학교 전자통신전파공학과 박사과정

<관심분야> Image Compression, Image Processing, H.264, Image Enhancement, Transcoding, MVC, SVC,

정 제 창 (Jechang Jeong) 정회원

1980년 2월 서울대학교 전자공학과 졸업

1982년 2월 KAIST 전기전자공학과 석사

1990년 미국 미시간대학 전기공학과 공학박사

1980~1986년 KBS 기술연구소 연구원(디지털 TV 및 뉴미디어 연구)

1990~1991년 미국 미시간대학 전기공학과 연구교수 (영상 및 신호처리 연구)

1991~1995년 삼성전자 멀티미디어 연구소 (MPEG, HDTV, 멀티미디어 연구)

1995~현재 한양대학교 전자통신컴퓨터공학과 교수 (영상통신 및 신호처리 연구실)

1998년 11월 27일 과학기술자상 수상

1998년 12월 31일 정보통신부장관상 표창

<관심분야> 영상처리 및 영상압축