고속 움직임 추정을 위한 개선된 블록 정합 방식

정회원 문 용 호*

An improved block matching algorithm for fast motion estimation

Yong-Ho Moon* *Regular Member*

요 약

PDE기법에 기반한 움직임 추정에서는 SAD가 누적 초기에 큰 값을 가지는 것이 이상적이다. 이를 위해 본 논문에서는 새로운 누적 순서 및 탐색 순서를 결합한 효율적인 블록 정합 방식을 제안한다. 제안하는 누적 순서는 현재 블록에서 고주파 성분이 큰 영역에 대하여 SAD 계산이 먼저 수행되도록 한다. 또한, 제안하는 탐색 순서는 현재 블록과 각 후보 블록간의 DC차와 SAD간의 연관성에 기반하여, 낮은 DC차를 지니는 후보 블록을 먼저 정합되도록 하였다. 모의 실험 결과는 제안 방식에 의하여 약 6%의 성능 향상이 이루어짐을 보여준다.

ABSTRACT

According to the PDE scheme, it is desirable to make the value of SAD to be rapidly increased in motion estimation. In this paper, we propose a new block matching algorithm based on the combination of a new scan and search order. In the proposed scan order, a current block is divided by the frequency that is estimated by summing the difference between the mean and the values of the pixels. And then the calculation of SAD is performed from the highest frequency area. In the proposed search order based on DC difference between the current and the candidate block, candidate blocks with low DC are early checked in motion estimation. Simulation results show that the proposed method achieves about 6% computational gain compared with the conventional methods.

I. 서 론

일반적으로 동영상 데이터에는 공간적(Spatial), 시간적(Temporal), 그리고 통계적 (Statistical) 중복성(Redundancy)이 존재한다[1]. 동영상 압축의 기본 개념은 영상내의 이러한 중복성을 제거하는데 그 기반을 두고 있다. 이 가운데에서 시간적 중복성 제거는 움직임 추정 및 보상을 통해 수행된다. 현재 하드웨어 구현이 용이한 블록 정합 알고리즘이 움직임 추정에 널리 이용되고 있으며, 유사성의 판단 기준으로서 SAD(Sum of Absolute Difference)가 사용된다. 일반적으로 블록 정합에 있어서 SAD 계산은 블록의 왼쪽에서 오른쪽으로, 위에서 아래 방향으로 정해진 순서에 따라 각 화소의 오차를 누적함으로써 수행되어진다.

블록 정합 알고리즘은 전역 탐색 알고리즘과 고속 탐색 알고리즘으로 크게 나뉘어 진다. 전역 탐색 알고리즘[2-4]은 모든 후보 블록에 대해 SAD를 계산함으로써 유사성을 판단한다. 이 방식은 최적의 움직임 벡터를 구할 수 있으나, 동영상 부호화기 전체 계산량의 60~70%를 차지하여 구현상 어려움을 지니고 있다. 이에 반하여 고속 탐색 알고리듬[5-6]은 제한된 수의 후보 블록에 대한 유사성만을 검사함으로써 계산량을 현저히 감소시키는 방식이다. 그러나 이것은 몇 개의 후보 블록만을 선정하여 탐색 영역을 좁혀 나가는 광역-협역 탐색

* 부산외국어대학교 정보과학대학 디지털정보공학부(yhmoon5@taejo.pufs.ac.kr)
논문번호 : 030026-0115, 접수일자 : 2003년 1월 15일

(Coarse-to-Fine Search)방식에 기반하고 있다. 따라서 이 방식은 국부 최소값(Local Minimum)에 빠지는 단점을 지니고 있다.

한편, 오늘날 블록 정합에서 요구되는 SAD 계산량을 감소시키기 위하여 PDE(Partial Distortion Elimination)[7]기법이 널리 사용되고 있다. PDE기법은 SAD 계산 과정 중에 그 값이 주어진 최소 SAD보다 커지게 되면 즉시 SAD 계산을 중지함으로써 불필요한 계산을 제거하는 방식이다. 이것은 SAD 계산 초기부터 큰 값으로 누적이 이루어질 수 록 효과가 더욱 더 증대된다. 따라서 순차적인 순서대로 SAD를 계산하는 기존 방식은 PDE 기법의 성능을 향상시키는 데 적합하지 않다.

PDE기법의 효과를 향상시키기 위하여 누적 순서(Scan Order)를 변경하는 방법이 연구되어 왔다. 누적 순서는 SAD 계산에 있어서 각 화소의 오차를 누적하는 순서를 말한다. 따라서 일반적으로 최종 SAD값은 누적 순서에 의하여 변하지 않는다. 그러나 SAD가 누적되는 추이는 누적 순서에 따라서 달라진다. 그러므로 계산 초기부터 SAD가 큰 값으로 누적될 수 있는 계산 순서를 개발한다면 보다 효과적으로 SAD 계산량을 감소시킬 수 있을 것이다. 현재 발표된 대표적인 누적 순서로는 베이어 디더 행렬 누적 순서(Bayer Dither Matrix Scan Order)[8]와 복잡도를 이용한 누적 순서(Scan Order by Localizing Complexity)[9]가 있다. 베이어 디더 행렬누적 순서는 베이어 디더 행렬을 이용한 순서로서, SAD 누적 추이는 선형적인 성향을 나타낸다. 복잡도를 이용한 누적 순서는 블록내의 복잡도를 조사하여 복잡도가 큰 영역에서 낮은 영역으로 계산을 진행하는 방식으로 성능이 가장 뛰어난 것으로 알려져 있다.

본 논문에서는 새로운 블록 정합 방식을 제안한다. 제안 방식은 새로운 누적 순서와 탐색 순서로 구성되어 진다. 제안하는 누적 순서는 현재 블록의 평균을 계산한 후, 평균과 현재 블록내 각 화소간의 차이를 구한다. 그리고 현재 블록내의 각 서브 블록들에 대하여 그 차이 값들의 합을 구한 후 이를 정렬한다. 이렇게 결정된 순서를 누적 순서로 하여 각각의 후보 블록들에서의 SAD 계산에 적용한다. 한편, 본 논문에서는 현재 블록과 후보 블록들간의 평균값 차이를 실험을 통하여 조사하였다. 그 결과, 평균값 차이가 작은 후보 블록일수록 최종 SAD값 역시 대체적으로 작다는 경향을 발견하였다. 이것은 평균값 차이가 최적의 후보 블록에 대한 정보를 제공한다는 것을 의미한다. 즉, 평균값이 작은 후보 블록이 최적의 후보 블록으로 선정될 가능성이 높다는 것이다. 이를 토대로 본 논문에서는 작은 평균값 차이를 지니는 후보 블록들이 먼저 탐색되어 SAD 계산이 수행되어지도록 하는 새로운 탐색 순서를 제안한다. 본 논문에서 제안하는 두 방식들의 결합은 실질적인 SAD 계산을 감소시켜서 블록 정합 속도를 보다 더 증가시킬 것이다.

Ⅱ. 기존의 누적 및 탐색 순서

블록 정합 알고리즘은 현재 블록과 가장 유사한 블록을 이전 프레임에서 찾는 과정이다. 그림 1은 이러한 과정을 보여준다.

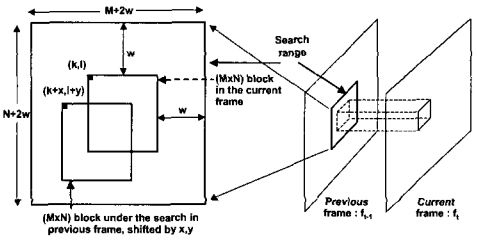


그림 1. 블록 정합 알고리즘
Fig 1. Block matching algorithm

일반적으로 유사성의 척도로서는 식 (1)과 같이 정의되는 SAD가 사용되었다.

$$SAD=\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|f_t(i,j)-f_{t-1}(i+x,j+y)|. \quad (1)$$

오늘날 블록 정합에 있어서 SAD 계산량 감소를 위해 PDE기법이 적용되어지고 있다. PDE기법은 SAD 계산 중 일정 간격마다 그때까지의 부분 SAD값을 주어진 최소 SAD와 비교한다. 그리하여 부분 SAD값이 최소 SAD를 초과하는 경우에 해당 후보 블록에서의 SAD 계산을 중지한다. 식 (2)는 부분 SAD값을 나타내고 있다.

$$P\_SAD(k)=\sum_{i=0}^{k}\sum_{j=0}^{N-1}|f_t(i,j)-f_{t-1}(i+x,j+y)|, \quad \text{for } k=1,2,\cdots,N. \quad (2)$$

식 (2)와 같이 계산되는 M-1개의 부분 SAD값들은 순차적으로 최소 SAD와 비교됨으로써 SAD 계산의 지속 여부가 판정된다. 이러한 PDE 기법은

계산 초기부터 부분 SAD값이 큰 값을 지니면서 누적되어질수록 그 효과가 증대된다. 즉, SAD의 누적 추이가 보다 급격해질수록 PDE기법의 효과는 증가한다.

2.1 누적 순서(Scan Order)

PDE 기법의 적용이 보다 효과적이기 위해서는 부분 SAD의 증가를 보다 빠르게 할 수 있는 적절한 SAD 계산 순서가 필요하다. 지금까지 제안되어 온 대표적인 방식으로 베이어 디더 행렬 누적 순서와 복잡도를 이용한 누적 순서가 있다.

2.1.1 베이어 디더 행렬 누적 순서[8]

일반적으로 베이어 디더 행렬은 정방형태로 구성되며, 행렬내의 연속되어지는 값들이 가능한 멀리 배치되어 있다. 그림 2는 이를 보여준다. 베이어 디더 행렬을 이용한 누적 순서는 기존의 베이어 디더 행렬에서 제시된 순서에 따라서 SAD를 계산하는

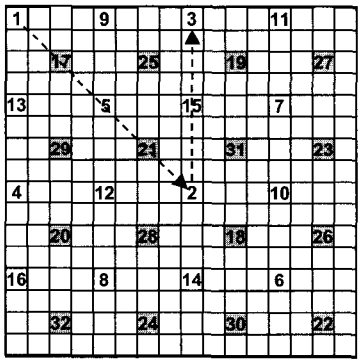


그림 2. 베이어 디더 행렬 누적 순서
Fig 2. Bayer dither matrix scan order

방식이다. 즉, 순차적으로 계산하는 것이 아니라 그림 2에 제시된 순서대로 SAD를 계산하면서 일정 간격마다 부분 SAD와 최소 SAD를 비교하는 것이다. 이 방식은 SAD 누적 추이가 선형적으로 나타나는 특징을 지니고 있다. 그러나 이 방식은 메모리 엑세스 횟수가 많아서 실제 구현에 적합하지 않다.

2.1.2 복잡도를 이용한 순서[9]

이 방식은 현재 블록내에서 복잡도가 높은 서브 블록에 대해 우선적으로 SAD를 계산하는 방식이다. 복잡도는 기울기(gradient) 계산에 의하여 결정된다. 수행 과정은 다음과 같다.

1) 현재 블록내의 각 화소에 대하여 기울기를 계산한다.
2) 현재 블록을 4×4 의 서브 블록으로 나눈다.
3) 서브 블록내의 각 화소에 대한 기울기 값을 모두 합하여 복잡도를 계산한다.
4) 복잡도를 바탕으로 서브 블록을 내림차순으로 정렬한다.
5) 그림 3과 같이 정렬된 순서에 따라서 각 서브 블록에 대한 부분 SAD를 계산함으로써 PDE 기법을 적용한다.

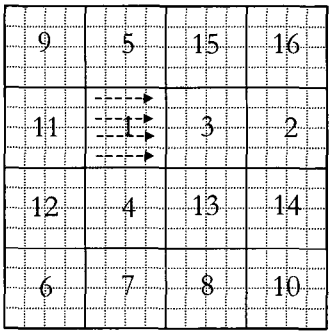


그림 3. 복잡도를 이용한 누적 순서
Fig 3. Scan order using complexity

이 방식은 베이어 디더 행렬을 이용한 누적 순서에 비하여 SAD 증가가 빠르다. 또한 각 서브 블록내에서의 계산은 순차적으로 이루어지므로 실제 구현시 요구되는 메모리 엑세스 횟수가 많지 않다.

2.2 탐색 순서(Search Order)

블록 정합에서 요구되는 계산량은 대부분 SAD 계산에서 발생한다. 따라서 효과적인 SAD 계산을 수행하는 PDE 기법은 블록 정합에서의 계산량을 감소시킨다. 한편, PDE 기법 이외에 효과적인 SAD 계산을 위하여 고려되어야 할 요소로 탐색 순서가 있다. 탐색 순서는 탐색 창 내의 각 후보 블록들에 대하여 정합 과정을 수행하는 순서이다. 만약 주어진 탐색 순서하에서 최적의 후보 블록이 조기에 검출된다면 이후에 검사되는 후보 블럭들에서의 SAD 계산량은 감소할 것이다. 왜냐하면 PDE 방식에 의하여 초기에 SAD 계산이 중지될 확률이 높아지기 때문이다. 따라서 효과적인 탐색 순서는 실질적인 SAD 계산량 감소에 중요한 영향을 미친

다. 현재까지 사용되는 방식으로는 순차적 탐색과 나선 탐색이 있다.

2.2.1 순차적인 탐색(sequential search)

이 방식은 탐색 창 왼쪽에서 오른쪽으로, 위에서 아래 방향으로 진행하면서 현재 블록과 각 후보 블록간의 유사성을 조사한다. 이 방식은 단순하고 구현이 용이하다.

2.2.2 나선 탐색(spiral search)[10]

이 방식은 초기 움직임 벡터를 중심으로 나선방향으로 진행한다. 따라서 이것은 초기 위치 근처에 최적의 후보 블록이 존재한다는 전제하에서 이루어지는 방식이다. 계산량 측면에서 순차적인 탐색 방법에 비해 우수한 것으로 알려져 있다.

Ⅲ. 제안 알고리즘

본 논문에서는 고속 움직임 추정을 위한 새로운 블록 정합 방식을 제안한다. 제안 방식은 PDE기법이 효율적으로 수행되기 위한 새로운 누적 순서 및 탐색 순서로 구성되어 진다.

3.1. 제안하는 누적 순서

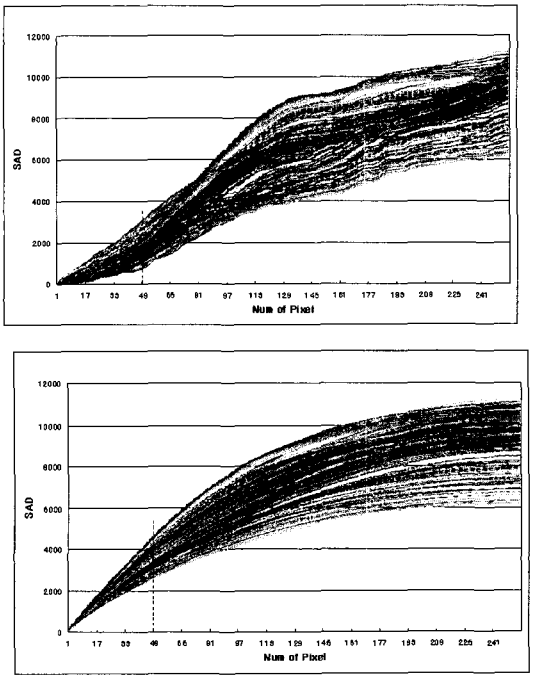
블록 정합은 모든 후보 블록에 대하여 SAD를 계산하고, 그 중 최소 SAD를 가지는 후보 블록을 찾는 과정이다. 이 때의 계산량은 화소 수에 비례하게 된다. 일반적으로 PDE기법의 성능은 부분 SAD와 최소 SAD와의 비교횟수를 나타내는 검사열 수로서 평가되어 진다[9]. 표 1은 PDE기법을 사용한 순차적 누적 순서 및 이상적 누적 순서에서 PDE기법을 적용했을 때의 검사열 수를 나타낸 것이다.

표 1. Full Search에서 누적 순서에 따른 검사열 수
Table 1. Checked rows according to scan orders in *Full Search*

초당 입력 프레임 수	시퀀스	순차적 누적	이상적 누적
30	Flower	6.3642	2.8264
	Football	7.5487	3.8408
10	Flower	8.9981	4.7369
	Football	8.7575	5.0555

표 1에서 이상적 누적 순서는 현재 블록과 후보 블록간에 대응되는 화소에 있어서 오차가 큰 순서대로 SAD를 계산한 방식이다. 즉, 실제적인 화소 차이에 대한 정보가 주어졌다는 가정하에서 SAD를 계산한 것이다. 표 1을 살펴보면, 누적 순서에 따라 검사열의 수에 큰 차이를 있음을 알 수 있다. 또한, 표 1에서 이상적 누적 순서에서의 검사열 수가 순차적 누적 순서에서 보다 매우 작음을 볼 수 있다. 이것은 이상적 누적 순서가 순차적 누적 순서에 비해 매우 효과적임을 입증하고 있다. 그림 4는 순차적 누적 순서와 이상적 누적 순서에 대한 후보 블록의 SAD 추이를 나타낸다. 여기서 각 선은 각각의 후보 블록들에 대한 SAD 누적 추이이다.

그림 4의 (a)와 (b)에서 각 후보 블록에 대한 최종 SAD는 동일하다. 그러나 최종 SAD까지의 추이는 큰 차이점을 보인다. 이상적 누적 순서에서 SAD추이는 순차적 누적 순서에 비해, 누적 초기부터 크게 증가한다. 예를 들어 49번째 화소까지 누적된 경우, 순차적인 누적 순서에서는 SAD가 760 이상인데 반해 이상적 누적 순서에서는 2700 이상의 값을 가진다.



(b) 이상적 누적 순서

그림 4. 누적 순서에 따른 SAD 누적 추이
Fig 4. SAD accumulation transition according to scan orders

그림 4를 종합해 보면 SAD가 빠르게 증가할수록 검사열 수는 감소함을 알 수 있다. 이것은 PDE 방식에서 SAD 계산이 초기에 중지되는 경우가 많아지기 때문이다.

그러나 이상적인 누적 순서는 계산 순서를 결정하기 위하여 많은 부가 계산량을 필요로 한다. 표 1과 그림 4는 이상적인 누적 순서가 주어졌다는 가정하에서 얻어진 실험 결과들이다. 즉, 이것은 PDE 기법에 의하여 얻어질 수 있는 성능의 한계를 보여주고 있다. 그런데 실제 이상적인 누적 순서를 적용하기 위해서는 누적 순서 정보를 얻기 위한 계산이 수행되어져야 한다. 이것은 많은 부가 계산량을 야기한다. 따라서 이상적인 누적 순서는 직접 적용될 수 없다.

본 논문에서는 많은 부가 계산량이 요구되지 않으면서 이상적인 누적 순서의 성능에 근접할 수 있는 누적 순서를 제안하고 자 한다. 효율적인 PDE 기법을 위해서는 SAD가 계산 초기부터 큰 값으로 누적되는 것이 바람직하다. 이것은 현재 블록과 후보 블록에서 대응하는 두 화소 차이가 큰 값부터 누적되어야 함을 의미한다. 일반적으로 현재 블록내에서 큰 고주파 성분을 지니는 화소와 이에 대응하는 후보 블록내의 화소간의 밝기차는 대체로 크다는 사실을 실험적으로 알 수 있다. 이것은 현재 블록의 평균과 블록내 각 화소간의 밝기 차이를 이용하여 누적 순서를 근사적으로 정할 수 있다는 것을 암시한다. 이러한 사실에 기반하여 본 논문은 PDE 기법의 효율을 향상시키기 위하여 현재 블록의 평균과 차이가 큰 화소를 우선적으로 계산하여 그 값을 누적시키는 방식을 제안한다. 제안 방식의 구체적인 수행 과정은 다음과 같다.

1) 현재 블록에 대한 평균을 구하고 현재 블록내 각 화소의 밝기와 평균간의 차를 계산한다. 식 (3)과 식 (4)는 이를 나타낸다.

$$m=\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}f_t(i,j), \tag{3}$$

$$dist(i,j)=|f_t(i,j)-m|. \tag{4}$$

2) 현재 블록을 서브 블록으로 분할 후, 서브 블록내의 각 화소에 대하여 $dist(i,j)$를 모두 더한다.

3) 2)에서 얻어진 값을 내림차순으로 정렬하여 서브 블록에 대응시킨다.

4) 후보 블록들에 대한 블록 정합시 정렬된 순서에 따라 SAD를 계산한다. 즉, 정렬 순서에 대응하는 서브 블록에 대하여 부분 SAD를 순차적인 순서에 따라 계산한다.

위의 과정중에서 2)와 3)은 [9]에서와 같이 메모리 엑세스 횟수를 감소시키기 위하여 도입되었다. 그리고 4)는 식(5)의 수식으로 표현되어 질 수 있다.

$$P\_SAD(k)=\sum_{q=0}^{k}\sum_{i=S_x}^{S_x+M_{mask}}\sum_{j=S_y}^{S_y+N_{mask}}|f_t(i,j)-f_{t-1}(x+i,y+j)|, \tag{5}$$

where S_x, S_y : 서브 블록의 위치,
M_{mask}, N_{mask} : 서브 블록의 폭과 높이,
q : 내림차순으로 정렬된 서브 블록,

제안 누적 순서는 현재 블록 정보만을 이용하여 누적 순서를 정한다. 따라서 현재 블록이 바뀔 때마다 새로운 누적 순서를 구해야 한다. 그러나 이것은 이상적인 순서에 비하여 작은 부가 계산량을 요구한다. 또한 서브 블록내에서의 계산이 순차적으로 수행되기 때문에 구현이 복잡하지 않다.

3.2 제안하는 탐색 순서

지금까지 제안된 고속 전역 탐색 방식들은 SEA[2] 방식을 토대로 발전되어 왔다. SEA는 현재 블록과 각 후보 블록간의 평균값들의 차이 즉, DC 차를 이용하여 불필요한 SAD 계산을 생략함으로써 계산량을 감소시켰다. 이에 본 논문에서는 현재 블록과 후보 블록에서의 DC차와 최종 SAD간의 상관성을 실험적으로 조사하였다. 그림 5는 현재 블록

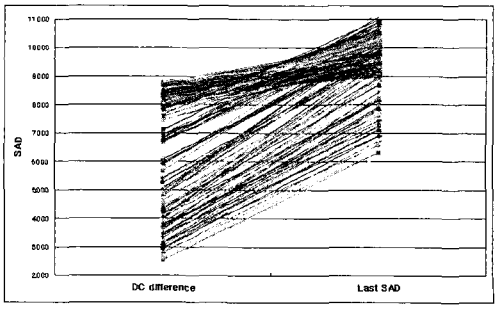


그림 5. DC차와 최종 SAD
Fig 5. DC difference and the last SAD

과 각 후보 블록간의 DC차와 그 때의 최종 SAD 값간의 관계를 나타낸 것이다. 그림 5에서 직선은 주어진 현재 블록과 하나의 후보 블록에 대하여 DC차와 최종 SAD값을 연결한 것이다.

그림 5를 통해 DC차와 최종 SAD간의 상관 관계가 있음을 쉽게 알 수 있다. 특히 최소 SAD부근의 SAD값을 지니는 후보 블록들에 있어서, 대응하는 DC차는 매우 낮음을 확인할 수 있다. 이것은 작은 DC차를 지니는 후보 블록이 최적의 후보 블록이 될 확률이 높다는 것을 의미한다. 따라서 DC차 정보를 활용하여 최적의 후보 블록으로 선정될 확률이 높은 후보 블록들에 대하여 우선적으로 블록 정합을 수행할 경우 PDE기법의 특성상 실질적인 SAD 계산량 감소가 이루어질 수 있을 것이다. 따라서 DC차 정보는 효과적인 탐색 순서를 결정하는 데 중요하게 이용될 수 있을 것이다. 이러한 사실에 근거하여 본 논문에서는 DC차를 이용하여 후보 블록에 대한 탐색 순서를 결정함으로써 SAD 계산량을 추가적으로 감소시키고 자 한다.

현재 가장 성능이 뛰어난 탐색 순서로서 나선 탐색 방식이 있다. 이 방식은 초기 후보 블록 주위에 최소 SAD를 지니는 최적 후보 블록이 존재할 가능성이 높다는 사실에 기반한다. 그리하여 이 방식에서는 초기 후보 블록을 중심으로 나선 방향으로 후보 블록들이 검색되어 진다. 본 논문에서는 나선 탐색의 실질적인 효과를 알아보기 위하여 초기 후보 블록과 최적의 후보 블록간의 거리를 실험적으로 조사하였다. 그 결과, 거리가 2이상인 경우가 거의 10%정도를 차지하였다. 이 결과는 거리가 2이상인 경우에 나선 탐색 순서로 블록 정합을 수행하는 것이 비효율적일 수 있다는 것을 의미한다. 즉, 움직임이 빠른 동영상의 경우에 기존 나선 탐색 방식은 계산량의 이득을 가져 올 수 없다. 따라서 거리가 2이상인 경우에는 효과적인 새로운 탐색 순서에 따라 블록 정합을 수행하는 것이 타당할 것이다.

이러한 배경하에서, 본 논문은 새로운 탐색 방법을 다음과 같이 제안한다. 우선 초기 후보 블록 주위의 일정 영역에 대하여 기존 나선 탐색 순서를 따른다. 그러나 초기 후보 블록에서 떨어진 영역에 대해서는 현재 블록과 후보 블록간의 DC 차를 조사하여 낮은 DC차를 가지는 후보 블록들에 대하여 우선적으로 블록 정합 과정을 적용한다. 이것은 앞에서 언급한 DC 차이가 작은 후보 블록이 낮은 SAD를 가질 가능성이 크다는 사실을 이용한 것이다. 이와 같은 제안 방식은 기존 나선 탐색 방식의 단점을 보완함으로써 계산량을 감소시키는 효과를 가져올 것이다. 그림 6은 제안하는 탐색 방법을 나타내고 있다. 여기서 x는 초기 후보 블록이다.

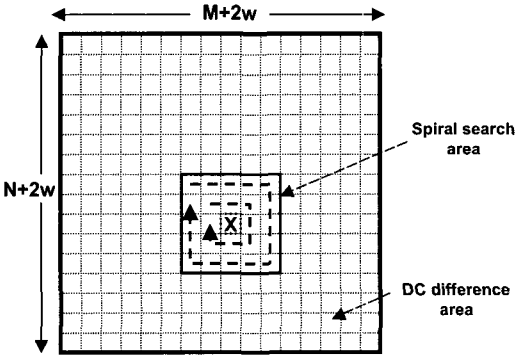


그림 6. 나선 탐색 영역 및 DC 차이 이용 영역
Fig 6. Spiral search area and DC difference area

Ⅳ. 모의 실험

본 논문에서는 해상도가 352x240인 SIF형식의 flower, football, mobile, tabletennis, tennis의 5가지 시퀀스에 대하여 모의 실험을 수행하였다. 이 때 사용된 부호화기는 MPEG-4이고, 프레임 수는 120프레임/30Hz와 40프레임/10Hz이다. 탐색 범위와 매크로 블록의 크기는 16x16으로 하였다. 그리고 PDE기법을 적용하기 위하여 매 16화소마다 1회씩 부분 SAD를 주어진 최소 SAD와 비교하였다. 또한 제안한 누적 순서에서는 버블 정렬 방식을 이용하여 서브블럭들을 정렬하였다.

표 2는 순차적 탐색하에서 각 누적 순서에 따른 검사열 수를 비교한 것이다.

표 2. 누적 순서에 따른 검사열 수
Table 2. Checked rows according to various scan orders

(a) 120frame/30Hz

누적 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	6.3642	7.5487	5.9658	8.9479	6.8074	7.1268
복잡도(4x4)	3.6086	4.7112	3.2233	6.5480	3.9862	4.4155
복잡도(4x2)	3.3529	4.4801	2.9254	6.2605	3.7558	4.1500
제안(4x4)	3.5892	4.6861	3.1999	6.4786	3.9196	4.3747
제안(4x2)	3.2929	4.3991	2.8731	6.1351	3.6346	4.0670
Ideal FS	2.8264	3.8408	2.3334	4.8678	3.2886	3.4314

(b) 40frame/10Hz

누적 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	8.9981	8.7575	6.1346	9.7956	8.9505	8.5273
복잡도(4x4)	6.4201	5.9398	3.4631	7.3725	6.3899	5.9171
복잡도(4x2)	6.1649	5.7113	3.1739	7.1057	6.1554	5.6622
제안(4x4)	6.3747	5.8842	3.4502	7.2785	6.2934	5.8562
제안(4x2)	6.0564	5.5964	3.1340	6.9520	5.9955	5.5468
Ideal FS	4.7369	5.0555	2.5143	5.7560	5.0908	4.6307

여기서, Full Search와 Ideal FS는 순차적 누적 순서와 이상적 누적 순서를 각각 사용한 것을 나타낸다. 일반적으로 검사열 수는 PDE기법에 의한 검사 횟수로서 부가 계산량을 반영하지 않은 순수한 성능을 나타낸다. 따라서 검사열 수가 작을수록 보다 효과적임을 알 수 있다. 표 2에서 제안한 누적 순서가 복잡도를 이용한 누적 순서에 비해 검사열 수가 6~8% 작고, Ideal FS에 보다 근접한 성능을 나타냄을 알 수 있다. 또한 제안한 누적 순서는 모든 시퀀스에 대해 항상 우수한 성능을 보여주는 특징을 지니고 있다.

표 3. 누적 순서에 따른 연산량
Table 3. Computational complexity according to various scan orders

(a) 120frame/30Hz

누적 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	79558	93287	74563	111968	85119	88899.0
복잡도(4x4)	48558	62400	43728	85414	53311	58682.2
복잡도(4x2)	46969	61408	41500	84172	52143	57238.4
제안(4x4)	47687	61458	42806	83916	51847	57542.8
제안(4x2)	45557	59728	40187	81922	49948	55468.4

(b) 40frame/10Hz

누적 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	112599	109579	76679	122601	112002	106692.0
복잡도(4x4)	83801	77803	46736	95754	83442	77507.2
복잡도(4x2)	84318	76148	46606	96028	83646	77749.2
제안(4x4)	82604	76477	45948	93947	81602	76115.6
제안(4x2)	80906	75043	43525	92375	80144	74398.6

표 3은 각 누적 순서에 따른 전체 계산량을 비교한 것이다. 여기서 계산량은 블록 정합에 소요되는 계산량과 부가 계산량을 모두 합한 전체 연산 횟수를 나타낸다. 즉, 평균값 계산, 정렬, 탐색, 그리고 SAD 계산등에서 사용되는 모든 연산 횟수를 반영한 것이다. 표 3에서 제안한 누적 순서가 실제 계산량에서 우수함을 알 수 있다. 제안한 누적 순서는 복잡도를 이용한 누적 순서에 비해 4~5%의 계산량 감소 효과를 보여준다. 또한 표 2에서와 마찬가지로 제안 방식은 모든 시퀀스에 대해 항상 우수한 성능을 나타내고 있다. 따라서 표 2와 표 3으로부터 제안한 누적 순서가 PDE기법에 효과적임을 쉽게 알 수 있다.

표 4는 제안한 누적 및 탐색 순서의 결합에 따른 검사열 수를 비교한 것이다. 표 4에서 누적 순서 뿐 아니라 탐색 순서가 전체 성능에 미치는 효과를 알아 볼 수 있다.

표 4. 누적 순서와 탐색 순서 결합에 따른 검사열 수
Table 4. Checked rows according to combined scans order and search orders

(a) 120frame/30Hz

누적 순서	탐색 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	순차적	6.3642	7.5487	5.9658	8.9479	6.8074	7.1268
복잡도(4x4)	나선	3.3520	4.3767	3.1113	6.2457	3.7477	4.1667
복잡도(4x2)	나선	3.0966	4.1457	2.8142	5.9547	3.5155	3.9053
제안(4x4)	제안	3.3358	4.2680	3.0886	6.1555	3.6645	4.1025
제안(4x2)	제안	3.0410	3.9835	2.7638	5.8132	3.3840	3.7971

(b) 40frame/10Hz

누적 순서	탐색 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	순차적	8.9981	8.7575	6.1346	9.7956	8.9505	8.5273
복잡도(4x4)	나선	5.9047	5.7112	3.2629	6.9314	6.1158	5.5852
복잡도(4x2)	나선	5.6477	5.4843	2.9729	6.6591	5.8796	5.3287
제안(4x4)	제안	5.8135	5.3891	3.2457	6.7526	5.8590	5.4120
제안(4x2)	제안	5.5366	5.1471	2.9700	6.4456	5.6190	5.1437

표 4에서 누적 순서 뿐 아니라 탐색 순서가 전체 성능에 미치는 효과를 알아 볼 수 있다. 표 4로부터 제안한 누적 순서와 탐색 순서를 결합한 방식이 Full Search에 비해 40~46%, 복잡도를 이용한 누적 순서와 나선 탐색을 결합한 방식에 비해 8~9% 정도의 성능이 개선됨을 알 수 있다. 그리고 표 2와 비교해 볼 때, 제안 한 탐색 방식의 도입은 검사열 수에 대한 성능의 향상을 가져옴을 알 수 있다.

표 5는 제안한 누적 및 탐색 순서의 결합에 따른

총 계산량을 비교한 것이다. 제안 방식은 부가 계산량을 포함한 전체 계산량 측면에서도 다른 방식에 비하여 가장 우수하였다. 제안 방식의 전체 계산량은 Full Search에 비해 34~40%, 비교대상에 비해 4~6% 감소하였다. 제안한 누적 순서만을 사용한 표 3과 비교할 경우, 제안한 탐색 방식은 보다 향상된 성능을 가져왔다. 그리고 표 4와 표 5의 모든 시퀀스에 대해서도 제안 방식이 항상 우수한 성능을 보여준다.

표 5. 누적 순서와 탐색 순서 결합에 따른 연산량
Table 5. Computational complexity according to combined scan orders and search orders

(a) 120frame/30Hz

누적 순서	탐색 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	순차적	79558	93287	74563	111968	85119	88899.0
복잡도(4x4)	나선	45538	58395	42496	81807	50521	55751.4
복잡도(4x2)	나선	43887	57318	40249	80440	49269	54232.6
제안(4x4)	제안	44935	56717	41786	80272	49218	54585.6
제안(4x2)	제안	42759	54912	39164	78212	47309	52471.2

(b) 40frame/10Hz

누적 순서	탐색 순서	Flower	Football	Mobile	Table Tennis	Tennis	Aver.
Full Search	순차적	112599	109579	76679	122601	112002	106692.0
복잡도(4x4)	나선	77566	75134	44412	90428	80217	73551.4
복잡도(4x2)	나선	76548	74452	42296	89476	79517	72457.8
제안(4x4)	제안	76231	70845	43761	87972	76788	71119.4
제안(4x2)	제안	74919	69871	41806	86515	75960	69814.2

V. 결 론

본 논문에서는 PDE 기법의 효과를 증대시키기 위하여 새로운 누적 순서 방식과 탐색 순서 방식을 각각 제시하였다. 그리고 이들 두 방식들을 결합함으로써 고속 전역 탐색이 가능한 효율적인 블록 정합 방식을 제안하였다. 제안한 누적 방식은 현재 블록 내에서 큰 고주파 성분을 지니는 영역이 먼저 계산되도록 함으로써 SAD가 보다 빠르게 누적되도록 하였다. 또한, 제안하는 탐색 방식은 현재 블록과 후보 블록간의 DC차와 SAD간의 관계를 바탕으로 작은 DC차를 지니는 후보 블록들이 먼저 탐색하도록 하였다.

모의 실험은 제안한 방식이 기존 방식에 비하여 우수한 성능을 지니고 있음을 보여준다. 제안한 누적 방식은 검사열 수와 전체 계산량 측면에서 약 4%정도의 이득을 가져왔다. 그리고 제안한 탐색 방식과 누적 순서 방식을 결합할 경우, 약 6%의 성능 향상을 확인할 수 있었다. 한편 제안 방식은 블록의 평균 정보를 알고 있는 SEA기반의 움직임 추정에서 더욱 효과적인 성능을 나타낼 것으로 예상된다.

참 고 문 헌

[1] Yun Q. Shi and Huifang Sun, *IMAGE and VIDEO COMPRESSION for MULTIMEDIA ENGINEERING*, Boca Raton London New York Washington, D.C:CRC Press, 2000.

[2] 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.

[3] X.Q.Gao, C.J.Duanmu and C.R.Zou, "A multilevel successive elimination algorithm for block matching motion estimation", *IEEE Trans. on Image Processing*, vol. 9, no. 3, pp. 501-505, Mar. 2000.

[4] Michael Brünig and Wolfgang Niehsen, "Fast Full-Search Block Matching", *IEEE Trans. on Circuits and System for Video Technology*, vol. 11, no. 2, pp. 241-247, Feb. 2001.

[5] H. G. Musmann, P. Pirsch and H.-J. Grallert, "Advances in picture coding", *Proc. IEEE*, vol. 73, no. 4, pp. 523-548, Apr. 1995.

[6] S. Zhu and K.Ma, "A new diamond search algorithm for fast block matching motion estimation", *IEEE Trans. on Image Processing*, vol. 9, pp. 287-290, Feb. 2000.

[7] ITU-T, "Video coding for low bitrate communication", Draft Recommendation H.263, Dec.1995.

[8] Y. H. Jeong and J. H. Kim, "The FASCO lock

matching Algorithm based on Motion Vector Prediction using Spatio-temporal Correlations", *Korean Institute of Communication Sciences*, vol. 26, no. 11A, pp. 1925-1937, Jan. 2002.

[9] J. N. Kim, S. C. Byun, Y. H. Kim, and B. H. Ahn, "Fast Full Search Motion Estiamtion Algorithm Using Early Detection of Impossible Candidate Vectors", *IEEE Trans. on Signal Processing*, vol. 50, no. 9, pp. 2355-2365, Sep. 2002.

[10] Ayala-Ramirez V., Devy M. and Parra C., "Active tracking based on Hausdorff matching", *Proc. Pattern Recognition 15th International Conf.*, vol. 4, pp. 706-709, Sep. 2000.

문 용 호(Yong-Ho Moon) 정회원

1992년 2월 : 부산대학교 공과대학 전자공학과(학사)
1994년 2월 : 부산대학교 일반대학원 전자공학과(석사)
1998년 8월 : 부산대학교 일반대학원 전자공학과(박사)
1998년 9월~2001년 8월 : 삼성전자 DM연구소 Digital Media Lab. 책임연구원
2001년 9월~2003년 2월 : 부산대학교 산업자동화 및 정보통신분야 인력양성 사업단(기금교수)
2003년 3월~현재 : 부산외국어대학교 정보과학대학 디지털정보공학부 (전임강사)

<주관심분야> 정지영상 및 동영상 압축, 영상 처리, 신호처리, VLSI, 신경회로망