논문 2006-43SP-6-7

적응형 임계값을 이용한 움직임 벡터 예측 방법

(Motion Vector Estimation using an Adaptive Threshold)

김 진 욱*, 박 태 근**

(Jin wook Kim and Tae geun Park)

요 약

움직임 예측은 비디오 신호의 압축에 중요한 역할을 한다. 본 논문에서는 효율적으로 움직임 벡터를 찾기 위하여 적응형 임계값과 매크로블록간의 차이값(Sum of Absolute Difference, SAD)의 분포특성을 이용하였다. 일반적으로 SAD분포가 단조로우면 SAD값이 작고 복잡하면 SAD값이 큰 경향이 있다. 따라서 단조로운 분포에서는 탐색 포인트를 줄이고 복잡한 분포에서는 지역 극소점을 피하기 위해 탐색 포인트를 늘려서 탐색하였다. 검색할 매크로 블록을 9개의 영역으로 나누고, 시공간적 유사성을 이용하여 예측한 영역을 제 1 영역이라 하고 나머지 8개의 영역을 모두 제 2 영역이라 정한다. 이 두 개의 영역 중 어느 한 영역(제 1 영역 또는 제 2 영역)만 탐색할지, 아니면 두 영역 모두 탐색할지를 적응형 임계값을 이용하여 적절하게 탐색하였다. 실험 결과 기존의 대표적인 고속 알고리즘들에 비하여 매크로블록 당 탐색 포인트 수가 평균 16.4% 감소하고, MSE는 평균 32.83 감소한 것을 확인할 수 있었다.

Abstract

Motion estimation plays an important role for the compression of video signals. The proposed method utilizes an adaptive threshold and characteristics of a distribution of SAD (sum of absolute difference). Generally, the more complex the SAD distribution is, the larger SAD value tends to be. This proposed algorithm tries to reduce the search points in a simple distribution but increase them in a complex distribution to avoid local minima. A macro block is divided into 9 areas. One of them chosen using spatio-temporal correlation is called the primary area and the others are called the secondary area that will be searched to avoid local minima. The proposed algorithm decides if just one area (the primary area or the secondary area) will be enough to be searched or both areas should be searched, using adaptive threshold. Compared with famous motion estimation algorithms, the simulation result shows that the searching points per macro block and MSE decreases about 16.4% and 32.83 respectively on the average.

Keywords: block matching, motion estimation, video compression

I. 서 론

통신 및 방송 분야에서, 고화질 TV, 주문형 비디오, 화상회의 그리고 CD-ROM 보관기록 등에서와 같이 디지털 비디오 및 오디오 신호를 제한된 대역의 채널을 통하여 전송하거나 저장매체에 저장하기 위해서는 이를 충분히 압축할 수 있는 부호화가 필요하다.

동영상에서는 상당한 양의 시간적인 중복성(temporal redundancy)이 존재하며 이를 적절히 이용하면 동영상의 압축 성능을 높일 수 있다. 움직임 추정 및 보상은 MPEG-1/2/4[1-2]와 ITU-T H.261/263/ 264[3-4] 같은 여러 영상 압축 표준에서 프레임간의 시간적 중복성을 없애기 위한 중요한 방법으로서, 동영상 압축효율에 가장 큰 영향을 주는 부분이다. 하지만 움직임 추정은 부호화기에서 가장 많은 계산 량을 요구하는 부분이기도 하다.

압축효율을 높이기 위하여 시간적인 유사성을 활용하는 움직임 벡터(MV)를 구한다. 프레임을 크기 N의 정방형태 매크로 블록(MB)으로 나누고 MB단위로 현재 프레임과 이전 프레임을 사용하여 최적의 MV를 탐색한다. 일반적으로 MV척도는 MAD(Mean Absolute

* 학생회원, ** 정회원, 가톨릭대학교 정보통신전자공학부
(School of Information, Communications and Electronics Engineering, Catholic University)
※ 본 연구는 한국과학재단 특정기초연구(R01-2005-000-11054 -0) 지원으로 수행되었음
접수일자: 2006년4월3일, 수정완료일: 2006년10월16일

Difference) 또는 SAD(Sum of Absolute Difference)가 사용된다. 그림 1에서와 같이 만일 이전 프레임에서 최대 R화소의 크기까지 MV 탐색이 가능하다면 가장 적당한 MV는 $(2R+1)^2$개의 후보를 가질 수 있다.

전역 탐색 방법(FS)은 $(2R+1)^2$개의 후보를 모두 탐색하는 알고리즘이다. FS 방법은 탐색범위 내의 모든 위치를 탐색하기 때문에 가장 최적의 움직임 벡터를 찾을 수 있으나 많은 계산량이 요구되어 실시간 동영상 압축 시스템에 적용하기에는 문제가 있다.

몇몇 알고리즘들은 계산의 복잡도를 줄이기 위해서 제안되었다. 예를 들면 3단계 탐색(TSS)[5], 개선된 3단계탐색(NTSS)[6], 4단계 탐색(4SS)[7], 다이아몬드 탐색(DS)[8-9], 크로스-다이아몬드 탐색(CDS)[10]과 같은 알고리즘이 있다.

본 논문에서는 효율적으로 지역극소점들(local minima)을 피하기 위하여 탐색범위를 9등분하여 이 9개의 영역 중 한 영역을 시공간적 유사성을 이용하여 제 1 영역으로 선택하고, 나머지 영역들을 제 2 영역이라 정의하였다. 그럼으로써 증가하는 복잡도를 줄이기 위하여 항상 예측한 영역과 예측하지 못한 영역을 탐색하지 않고 적응형 임계값(adaptive threshold)을 이용하여 SAD분포에 따라서 적절하게 탐색하는 방법으로 복잡도를 개선하였다.

본 논문은 다음과 같이 구성된다. 먼저 Ⅱ장에서는 SAD값을 사용하는 기존의 움직임 추정 기법에 대해 설명하였다. 탐색범위 내 모든 위치의 블록을 후보 블록으로 하는 전역 탐색 방법과 후보 블록의 개수를 줄이기 위한 여러 고속 탐색 방법들에 대해 간략히 소개하였다. Ⅲ장에서는 SAD값의 분포를 설명한 후 제안한 알고리즘을 설명하였다. Ⅳ장에서는 모의실험 결과를 분석하였으며 결론은 Ⅴ장에 기술하였다.

Ⅱ. 고속 움직임 예측 알고리즘

움직임 추정은 블록정합알고리즘(BMA)을 이용하여 수행된다. 일반적으로 BMA에서 부호화될 현재 프레임은 MB단위로 나뉘어 지며 각각의 MB당 유사한 MB을 이전 프레임에서 찾게 된다.

이때 유사한 블록의 기준으로 MAD 값과 SAD값을 사용한다. 식(1)은 그림 1에서의 t 번째 프레임 내에서 (u,v)에 위치한 크기 n 인 MB에서의 SAD값에 대한 수식이다.

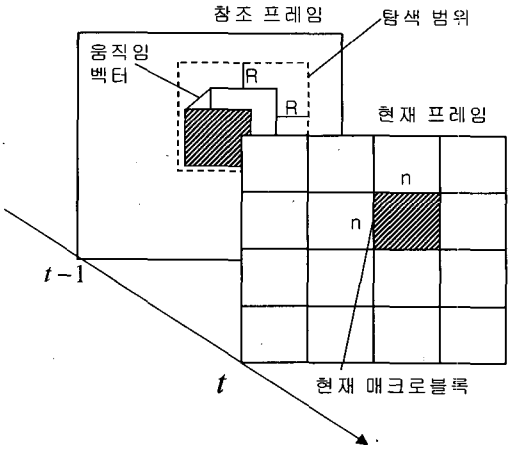


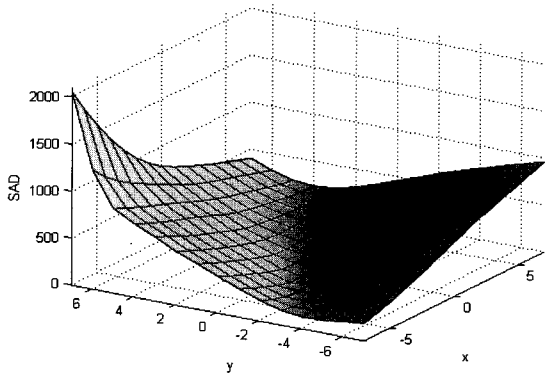
그림 1. 움직임 벡터 추정
Fig. 1. Motion vector estimation.

$$SAD^t(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{n} \left| MB^t(u,v) - MB^{t-1}(u+i,v+j) \right| \quad (1)$$

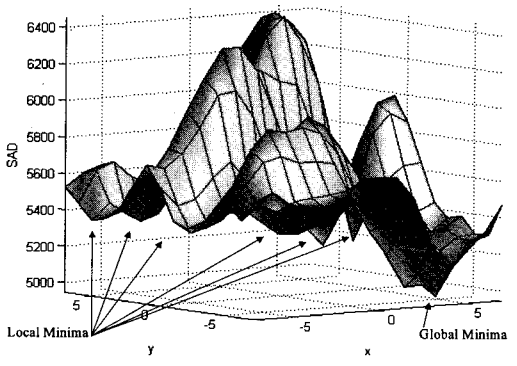
BMA 중 FS방법은 탐색범위 내에서 모든 위치의 후보들을 탐색하기 때문에 가장 최적의 MV를 찾을 수 있으나 계산 량이 많이 요구되어 실시간 동영상 압축 시스템에 적용하기에는 문제가 있다. 따라서 이를 개선하기 위해 많은 고속 알고리즘들이 제안되었다.

이 중 가장 널리 쓰이는 방법 중의 하나인 TSS 방법은 탐색범위(R) ±7에서 탐색 포인트 간 간격이 첫째, 둘째, 셋째 단계에서 각각 4, 2, 1인 경우, 이 알고리즘에서 탐색 포인트 수는 9 + 8 + 8 = 25개 이고, 225개의 포인트를 탐색하는 FS와 비교하였을 때 약 9배의 계산량 감소를 얻을 수 있다. NTSS 방법은 중앙에 치우친 분포를 활용하여 TSS 방법을 개선하였다. 작은 MV에 적응하기 위해 TSS 방법에서 탐색범위의 중앙을 둘러싼 8포인트를 포함하여 탐색하였다. 이 추가된 8개의 포인트를 탐색하는 경우 탐색 포인트 측면에서 기존의 TSS 방법의 25개 보다 적은 18개의 탐색 포인트만으로도 탐색을 마칠 수 있다. 따라서 TSS 방법과 유사한 성능을 얻으며 더 적은 탐색 포인트를 사용하였다.

4SS 방법은 TSS 방법에서 2 단계 탐색형태와 같은 간격 2인 탐색창과 TSS 방법의 3 단계 탐색형태와 같은 간격 1인 탐색창을 사용하여 간격 2인 탐색창에서의 중앙 포인트가 최적 포인트일 때까지 계속 탐색한 후 간격 1인 탐색창으로 탐색하는 방법이다. MSE 측면에서 NTSS 방법과 유사한 성능을 얻었으며 탐색 포인트



(a) foreman seq. 1번째 프레임 (10, 5)
(a) foreman seq. frame no.1 (10, 5)



(b) football seq. 16번째 프레임 (10, 2)
(b) football seq. frame no.16 (10, 2)

그림 2. SAD값의 분포 유형
Fig. 2. Distribution type according to SAD.

수를 줄여서 개선하였다.

DS 방법은 탐색형태를 다이아몬드 탐색형태인 LDP (Large Diamond Pattern)와 SDP(Small Diamond Pattern)를 사용하였다. LDP로 계속 탐색을 하여 최적의 포인트가 LDP의 중앙 포인트에 위치할 경우 SDP로 탐색하고 마치는 방법으로 기존의 알고리즘보다 우수한 성능을 얻을 수 있었지만, DS 방법은 SAD분포가 단조롭다는 가정이 아니라면, 지역 극소점에 빠질 위험이 있다.

CDS 방법은 중앙부분에서 8포인트를 十형태로 탐색 후 적당한 MV를 찾지 못 할 경우 DS 방법으로 탐색하였다. 따라서 MV가 중앙에 있는 경우에 DS 방법보다 더 적은 탐색 포인트를 사용하였다.

이러한 고속 움직임 예측알고리즘에서 탐색 패턴의 모양과 크기는 알고리즘의 성능을 결정짓는 중요한 요소이다. 탐색하는 후보 지점의 위치와 개수를 적절히 조절하여 움직임 예측의 정확도와 탐색 점을 조절한다. 하지만 이 방법으로는 지역극소점들을 피하기 어려우며 지역극소점에 빠짐으로 인하여 총 SAD값이 증가한다. 제안한 알고리즘은 탐색 점을 적게 사용하기 위한 패턴을 사용할 뿐만 아니라 SAD값의 특성을 이용하여 지역극소점들을 피하였으며 실험을 통해 우수한 성능이 증명되었다.

Ⅲ. 제안한 알고리즘

TSS, NTSS, 4SS, DS, CDS 등의 알고리즘들은 탐색 시 각 단계에서 가운데 주위의 몇 개의 점들을 탐색하고 그 점들 중에서 SAD값이 최소인 점을 찾아 다시 그 점을 중심으로 몇 개의 점들을 탐색하는 과정을 반복한다. 이 때 주위의 점들을 탐색했음에도 불구하고 최소의 SAD값이 자기 자신일 경우에는 그 가운데의 점에서 SAD값이 최소의 SAD값이기 때문에 탐색을 중단하는 방법으로 알고리즘의 복잡도를 개선하였다. 하지만 이러한 방법은 지역극소점들에 빠질 위험이 있다. 그림 2는 탐색영역에서의 SAD값의 분포를 나타내었다. 그림 2(b)에서 잘 알 수 있듯이 우리는 지역극소점들을 예측할 수 없다. 위의 기존 알고리즘에서와 같이 한 방향으로만 탐색을 진행하면 지역극소점들을 피할 수 없다. 또한 이러한 기존의 블록정합알고리즘들은 각 단계를 진행할 때 탐색 점의 수가 일정하게 증가하지만 때로는 그렇게 많은 탐색 포인트를 투자하기에 적절하지 않을 때도 있다.

1. SAD(Sum of Absolute Difference)분포 특성

탐색 점의 수와 SAD값의 크기는 대체로 반비례 한다고 볼 수 있다. 즉, 많은 점을 탐색할수록 SAD값은 감소한다. 따라서 BMA알고리즘이란 적은 수의 탐색 점을 이용하여 상대적으로 우수한 SAD값을 유지하는 최적화 알고리즘이라고 이해할 수 있다. 그림 2와 같이 SAD값의 분포는 크게 다음과 같이 두 가지 형태로 이해할 수 있다. 첫 번째, 전체적으로 탐색영역이 작은 SAD값으로 분포된 경우로 그림 2(a)와 같은 경우이다. 이처럼 대부분 작은 SAD값으로 이루어진 영역에서는 지역극소점들이 많이 발생하기보다 단조감소 하는 경향이 크다. 따라서 전체적으로 탐색 점의 수를 줄이더

라도 SAD값은 크게 나빠지지 않는다. 반면에 영상의 물체가 탐색영역 내에서 큰 움직임을 가지면 큰 블록간의 오차를 발생시킨다고 볼 수 있다. 그림 2(b)에서는 그림 2(a)에 비해 전체적으로 큰 SAD값을 갖는다. 이와 같은 분포에서는 지역극소점들이 존재할 가능성이 크다. 만일 지역극소점들에 빠진다면 전역극소점(global minima)과의 오차가 매우 크므로 SAD값이 상당히 나빠진다. 즉, SAD분포의 변동성이 크므로 기존보다 더 좋은 성능을 유지 하려면 이 지역극소점들을 피하기 위해 노력해야 한다.

본 논문에서는 시공간 유사성을 이용하여 예측한 영역을 제 1 영역이라 하고 나머지 8개영역들 중 최소의 SAD값을 얻은 탐색 포인트가 속해 있는 탐색 영역을 제 2 영역이라 정의하고, 제 1 영역과 제 2 영역을 적절하게 탐색함으로써 지역극소점들을 피하고자 하였다.

2. 제안한 알고리즘

본 논문에서는 시공간적 유사성을 이용하여 현재 탐색해야 할 탐색범위에서 MB의 MV위치를 그림 3에서의 MV_1^t, MV_2^t, MV_3^t, MV_0^{t-1} 총 4개의 MV를 다음과 같이 식(2)을 이용하여 예측한다.

$$MV_{mean} = Mean\,(MV_1^t, MV_2^t, MV_3^t, MV_0^{t-1}) \quad (2)$$

제안한 알고리즘의 전체적인 구조는 그림 4에 흐름도로 나타내었다. 탐색영역을 균일하게 5×5의 정방형으로 9등분하여 위에서 예측한 점을 포함하고 있는 영역을 제 1 영역, 나머지 8 개영역들 중 최소의 SAD값을 얻은 탐색 포인트가 속해 있는 탐색 영역을 제 2 영역이라 정의한다. 우선, 제 1 영역에서 중앙의 점 1개만을 탐색하여 SAD값(SAD_0)을 구한다.

SAD_0값이 실험을 통하여 얻은 임계값 L_2보다 작다면 현재 탐색하려는 MB에서의 SAD값의 분포가 전체적으로 단조롭다고 예상할 수 있다. 이 경우 전체적으로 SAD값이 작은 분포이기 때문에 제 1 영역만을 탐색하여 탐색 포인트 수를 줄인다.

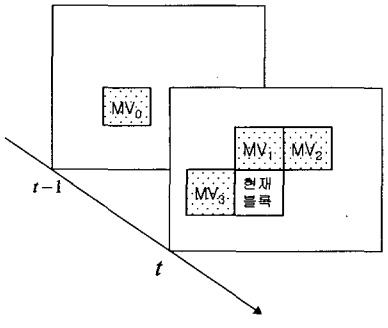


그림 3. 시공간적 유사성을 이용한 MV의 예측
Fig. 3. MV prediction using spatio-temporal correlation.

SAD_0가 L_2보다 큰 경우에는 현재 탐색하려는 MB에서의 SAD값의 분포가 복잡하다고 예상할 수 있다. 이 경우 지역극소점을 피하기 위해서 제 2 영역까지 탐색한다. 하지만 항상 제 1, 2 영역을 탐색 한다면 충분히 좋은 성능은 얻을 수 있지만 많은 계산량이 요구되므로 제 1 영역의 SAD값(SAD_1)과 제 2 영역의 SAD값(SAD_2)의 차이 값을 실험을 통하여 얻은 파라메터인 $Diff_{TH}$와 비교하여 차이가 충분하면 작은 쪽만 탐색하고, 차이가 확실하지 않을 경우에만 두 영역을 모두 탐색한다. 그 다음, 최적의 MV에 대응하는 SAD값을 이용하여 임계값 $Diff_{TH}$를 갱신한다. 적응형 임계값인 $Diff_{TH}$는 제 1 영역과 제 2 영역 중 어느 한 영역만 탐색가능한지를 판단하기 위한 중요한 값이며, 이 값은 MB단위로 존재하기 때문에 각각의 MB에서 최적의 MV를 찾은 후 값을 갱신하여 다음 프레임에서의 같은 위치의 MB을 탐색할 때 사용한다.

가. 모드선택

식(2)와 같은 시공간적 유사성을 이용하여 제 1 영역을 찾는다. 제 1 영역은 탐색영역 15×15(±7)를 정방형 5×5로 나누어 생성된 9개의 영역 중 한 영역이다. 그림 6 a)-i) 에서의 밝은 부분과 같이 9가지 경우가 있다. 제 1 영역에서 그림 5(a)의 중앙의 한 점(ⓒ)을 탐색하며 이 때 얻은 SAD값을 SAD_0이라한다. 이 값을 이용하여 그림 4에서와 같이 현재 MB에 대한 탐색을 멈추거나, 제 1 영역만 탐색하거나, 제 1 영역과 제 2 영역을 모두 자세히 탐색할 것인지를 결정한다. SAD_0값이 L_1보다 작은 경우는 충분히 작은 SAD값을 얻었다고 판단하고 복잡도 개선을 위해서 탐색을 정지하였다. (L_1 = 100, L_2 = 1000).

나. 단순 탐색 모드

SAD_0가 L_2보다 작을 경우인 단순탐색모드는 현재 탐색하려는 MB에서 SAD값의 분포가 단조로워 탐색 포인트를 적게 사용하겠다고 판단한 경우이며 5×5인 제 1 영역을 넓고 균등하게 탐색하기위해 그림 5(a)와 같이 먼저 1/5탐색을 한다. 이전 단계인 모드선택 단계에서 SAD_0값을 얻기 위해 이미 중앙의 한 점(ⓒ)을 탐색했기 때문에 4개의 탐색 점만을 추가로 탐색한다. 이렇게 1/5탐색을 사용하였으므로 4개의 포인트 추가로 기존의

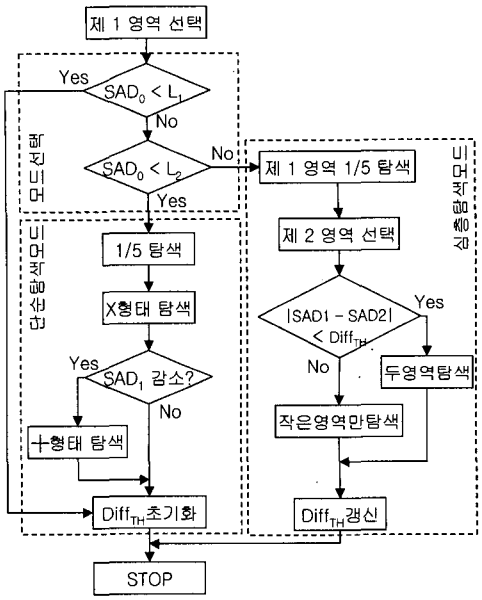
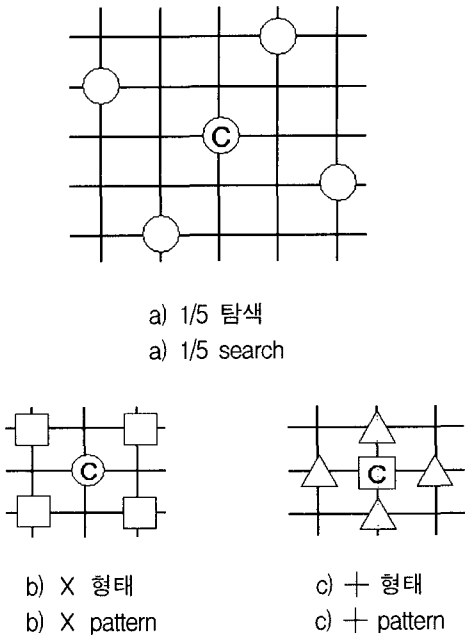


그림 4. 제안한 알고리즘 흐름도
Fig. 4. Flow chart of the proposed algorithm.



a) 1/5 탐색
a) 1/5 search

b) X 형태
b) X pattern

c) + 형태
c) + pattern

그림 5. 탐색 형태
Fig. 5. search pattern.

알고리즘들보다 더 넓은 범위를 탐색할 수 있다.

그 다음, 그림 5(b)와 같은 X형 탐색을 진행한 후에 그림 5(c)와 같은 十형 탐색을 진행하지만 항상 진행하지는 않는다. 만약 1/5 탐색이후 X형 탐색을 진행함으로써 SAD값이 감소하지 않았기 때문에 4개의 탐색 점들만큼 손해를 보았다면 十형 탐색으로 8개의 탐색 점을 투자하여 약간의 SAD값 개선을 기대할 수 있으며 그다지 효율적이지 않다. 따라서 1/5탐색이후 X형 탐색에서 SAD값이 감소해야만 十형 탐색까지 진행한다.

다. 심층 탐색 모드

심층탐색모드는 현재 탐색하려는 MB에서 SAD값의 분포편차가 심하고 지역극소점에 빠질 우려가 있다고 판단한 경우이다. SAD_0가 L_2보다 큰 경우인 심층탐색모드는 제 1 영역을 먼저 1/5탐색을 한다. 그림 6에서 밝은 부분은 제 1 영역, 어두운 부분은 제 2 영역 후보이다. 제 1 영역은 9가지 경우가 가능하며 이에 따라서 제 2 영역의 후보도 정해진다. 지역극소점을 피하기 위해 그림 6과 같이 제 1 영역을 제외한 8개 각 영역의 중앙 한 점씩, 총 8개를 탐색한다. 그리고 이 중 가장 최소의 SAD값을 갖는 영역을 제 2 영역으로 정한다.

제 2 영역에서는 넓게 8개의 점을 탐색하기 때문에 모든 영역을 탐색하지 않은 제 1 영역의 예측의 단점을 보완할 수 있다.

이렇게 제 1 영역 1/5탐색과 제 2 영역선택을 마친 후, 현재 제 1 영역에서의 최소 SAD값을 SAD_1, 제 2 영역에서의 최소 SAD값을 SAD_2라 정의한다. 항상 두 영역을 탐색하는 것은 비효율적이며 이 두 개의 SAD값의 차이를 실험을 통하여 얻은 파라메터인 $Diff_{TH}$와 비교하여 어느 한 영역만을 탐색할 수 있을지 판단한다. 만약 두 값의 차이가 충분히 많이 난다면 SAD값이 작은 영역만 탐색하며, 충분히 많이 나지 않는다면 두 영역 모두 탐색한다. 심층탐색모드의 탐색은 그림 5와 같이 a), b), c)순서로 진행된다.

라. 적응형 임계값 갱신

이렇게 탐색을 마친 후, $Diff_{TH}$값을 각각의 MB마다 적절히 수정한다. MB들은 시간적 상관관계가 매우 높다. 그러므로 현재 MB에서의 SAD값을 이용하여 다음 프레임에서 같은 위치의 MB에서 사용할 임계값을 미리 정할 수 있다.

만약 현재 MB에서 SAD값이 작았다면 다음 프레임의 같은 위치의 MB도 유사하게 SAD값이 작을 확률이 크며 따라서 SAD분포가 단조로울 것이라고 예측할 수 있고 이 경우 $Diff_{TH}$값을 한 단계 감소시킨다. $Diff_{TH}$값이 감소되면 두 영역 모두 탐색할 확률이 줄어들어 탐색 점의 수가 감소될 가능성이 크다.

반대로 SAD값이 크다면 다음 프레임의 동일한 위치의 MB에서 복잡한 SAD분포를 예상할 수 있으므로 $Diff_{TH}$값을 증가시키며, 두 영역 모두 탐색할 가능성이

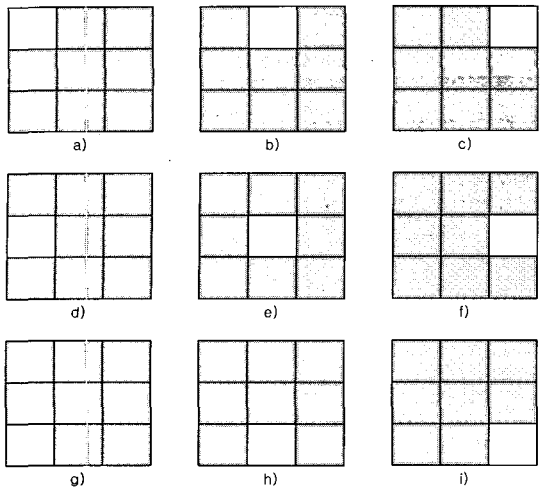


그림 6. 탐색 영역 유형
Fig. 6. a type of search area.

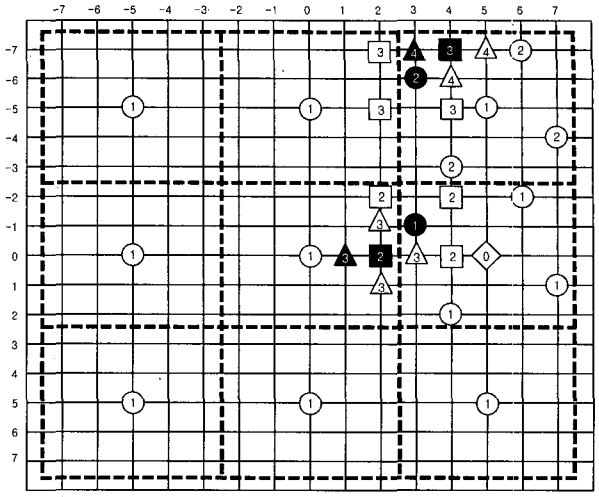


그림 7. 두 영역 탐색의 예
Fig. 7. example of both of two areas.

증가되여 탐색 점의 수는 증가하나 좋은 SAD 값을 얻을 수 있을 것이다. 이를 정리하면 다음과 같다.

if SAD < L_3 then $Diff_{TH}$ = $Diff_{TH}$ - δ
else if L_3 < SAD < L_4 then $Diff_{TH}$ = $Diff_{TH}$
else if L_4 < SAD then $Diff_{TH}$ = $Diff_{TH}$ + δ

(실험을 통하여 얻은 임계값 L_3, L_4)

그러므로 적응형 임계값($Diff_{TH}$)은 MB마다 각각 독립적으로 존재하며, 현재 탐색이 끝난 MB의 MV의 SAD값을 이용해 위와 같이 $Diff_{TH}$를 갱신한다. 여기서 $Diff_{TH}$값은 제 1 영역 혹은 제 2 영역 중 어느 한 영역만 탐색할지와 제 1, 2 영역 모두 탐색할지를 결정하기 위한 중요한 임계값이므로 프레임단위 보다 MB단위로 더욱 세밀하게 다루는 것이 적당하다.(L_3 = 2000, L_4 = 3000)

알고리즘의 이해를 돕기 위하여 두 영역 탐색의 예를 그림 7에 나타내었다. 그림 7은 fooball 영상에서 16번째 프레임의 (19, 10)에 위치한 MB이다. 시공간적 유사성을 이용하여 얻은 MV_{mean}은 (5, 2)이므로 제 1 영역은 그림 6(f)에서의 밝은 부분과 같다. 우선, 제 1 영역의 중앙 1개의 포인트를 탐색하여 SAD_0값을 얻는다. 모드선택 단계에서 SAD_0 = 6892 이므로 심층탐색모드로 진행한다. 먼저, 제 1 영역에서 4개의 점을 추가 탐색하여 MV(3, -1) 이고 SAD값이 6534인 ❶을 얻을 수 있으며 따라서 SAD_1 = 6534이다. 그 다음, 제 2 영역을 얻기 위하여 8구역의 ①들을 탐색하여 MV(5, -5) 이고 SAD값이 5652인 그림 6(f)에서의 ① 포인트가 제 2 영역임을 찾을 수 있으며 따라서 SAD_2 = 5652이다.

두 영역 SAD 값의 차이가 882이므로 $Diff_{TH}$보다 작다(이전 단계에서 $Diff_{TH}$ = 1000 으로 갱신). 따라서 두 영역 중 어느 한 영역만 탐색할 수 없으므로 두 영역

표 1. 평균 MSE와 PSNR의 비교
Table 1. The performance comparison of average PSNR and MSE.

Algorithm	FS		TSS		NTSS		4SS		DS		CDS		proposed	
Test Sequence	average PSNR	average MSE	average PSNR	average MSE	average PSNR	average MSE	average PSNR	average MSE	average PSNR	average MSE	average PSNR	average MSE	average PSNR	average MSE
Foreman	27.270	153.396	26.578	175.314	26.411	178.058	26.043	196.361	25.922	200.013	25.875	200.148	26.923	162.972
Football	23.296	311.388	22.946	337.102	22.956	336.514	22.708	355.829	22.575	366.455	22.571	366.571	23.098	325.694
Bus	26.311	155.280	23.909	277.902	23.967	275.216	22.995	338.990	23.051	339.268	22.985	343.203	24.657	230.684
평 균	25.626	206.688	24.477	263.439	24.444	263.263	23.916	297.060	23.849	301.912	23.810	303.307	24.893	239.783

표 2. 블록 당 평균 탐색 점의 수 비교
Table 2. The performance comparison of average search points per block.

Algorithm / Test Sequence	FS		TSS		NTSS		4SS		DS		CDS		proposed	
	SearchPT	Speed Up	SearchPT	Speed Up	SearchPT	Speed Up	SearchPT	Speed Up	SearchPT	Speed Up	SearchPT	Speed Up	SearchPT	Speed Up
Foreman	225.000	1.000	25.000	9.000	31.475	7.149	24.701	9.109	27.099	8.303	28.819	7.807	19.830	11.346
Football	225.000	1.000	25.000	9.000	23.352	9.635	19.984	11.259	18.006	12.496	16.685	13.485	18.840	11.943
Bus	225.000	1.000	25.000	9.000	28.572	7.875	21.834	10.305	21.127	10.650	21.328	10.550	21.187	10.620
평 균	225.000	1.000	25.000	9.000	27.799	8.220	22.173	10.224	22.077	10.483	22.277	10.614	19.952	11.303

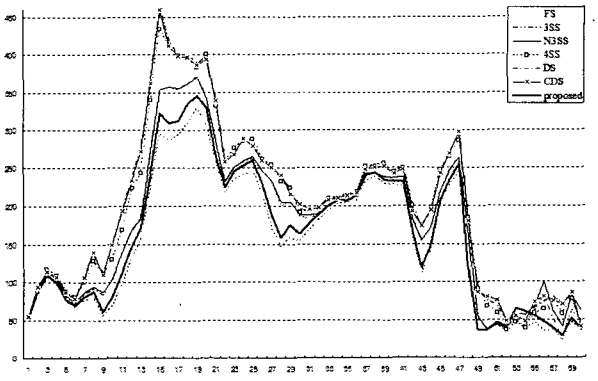
모두 탐색한다. 제 1 영역은 X형과 十형 탐색을 진행하고, 제 2 영역도 남은 1/5, X형, 十형 탐색순서로 진행한다. 최종적으로 제 1 영역의 최적의 MV(▲3)는 (1, 0)이며, SAD_1 = 5981이다. 제 2 영역의 최적의 MV(▲4)는 (3, -7)이며 SAD_2 = 4787 임을 얻게 되며 SAD_2값이 작으므로 최종MV = (3, -7)이 된다. 최종 SAD값을 이용하여 현재의 MB의 $Diff_{TH}$값을 갱신한 후 탐색을 마친다.

Ⅳ. 결과 분석

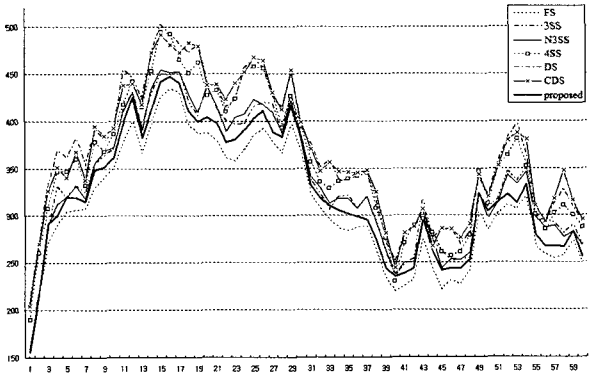
본 논문에서는 다양한 테스트 영상을 사용하여 기존의 예측 방법들과 제안한 알고리즘의 성능을 실험 분석하였다. 또한 각 방법을 통하여 부호화된 영상의 MSE와 움직임 벡터 탐색 과정에서의 탐색 점의 수를 비교하였다. CIF영상에서 60 프레임 단위로 실험하였고 탐색 범위는 15×15(±7)로 수행하였으며 기존의 FS, 3SS, N3SS, 4SS, DS, CDS알고리즘과 비교하였다.

표1은 평균 MSE 값을 표2는 블록 당 평균 탐색 포인트의 수를 각각 나타낸다. N3SS의 경우에 최적의 MV가 탐색영역의 중앙에 존재할 경우 1단계탐색에서 중앙부분 탐색을 많이 하기 때문에 유리하지만 MV가 탐색영역 외곽에 존재할 경우 중앙부분 탐색은 오히려 탐색 포인트 수만 증가하는 단점이 있다. 4SS방법과 DS방법은 최적의 MV의 크기가 클 경우 중앙부터 탐색을 시작하여 3개 혹은 5개의 탐색 포인트를 증가시키고 MV를 2화소씩 이동하며 최적의 포인트를 탐색하는 방법이기 때문에 SAD 값이 단조감소 해야 하며 그렇지 않을 경우 탐색 포인트 수가 증가된다. 예를 들면, (7, -6)이 최적의 MV이고 중앙에서 단조 감소하는 SAD 값의 분포라 가정을 한다면 30개 이상의 탐색 포인트를 탐색 해야만 최적의 MV를 얻게 되는데 이는 TSS방법보다 더 많은 탐색 포인트를 필요로 한다.

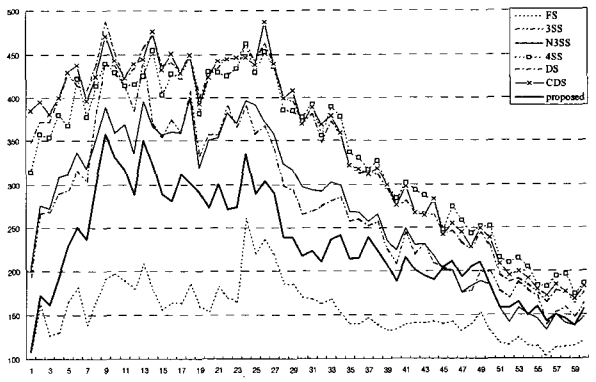
그림 8는 다양한 영상들에 대한 각 프레임당 MSE 값을 나타낸다. 그림 8(a)는 foreman 영상의 실험 결과이다. 양 끝부분에는 상대적으로 모든 알고리즘들이 FS방법에 근사한 성능이 나타나는 것으로 볼 수 있다. 하지만 중앙부분에 FS방법의 MSE값이 높은 부분에서는 지역극소점들에 빠지기 때문에 알고리즘들 간에 차이가 나게 됨을 알 수 있다. 그림 8(b)는 football 영상의 실험 결과이며, FS방법의 MSE값이 전반적으로 높기 때문에 전반적으로 제안한 알고리즘이 대체로 지역극소점들에 빠지지 않는다는 것을 알 수 있다. 그림 8(c)에서는 bus



(a) foreman



(b) football



(c) bus

그림 8. 다양한 영상에 대한 MSE 성능비교
Fig. 8. MSE comparison for various video samples.

영상의 실험 결과이다. 실제로 버스의 속도가 점점 줄어드는 테스트 동영상이므로 프레임수가 증가할수록 전체적으로 MSE값이 작아지는 경향을 관찰할 수 있다. 이 그림에서도 역시 제안한 알고리즘은 상대적으로 FS방법에 근사하는 성능을 얻었다.

제안한 알고리즘에서는 지역극소점들을 피하기위해서 두 영역을 탐색하였기 때문에 좋은 성능을 얻을 수 있었으며, 적응형 임계값을 사용하여 항상 두 영역을 모두 탐색하지 않고, 둘 중 한 영역을 탐색하거나 두 영역을 탐색하여 탐색 점을 적게 사용하였다. 결론적으로

제안한 알고리즘의 MSE값은 다른 알고리즘에 비하여 평균 32.83 감소했으며 움직임 벡터 탐색 포인트 수는 평균 16.4% 감소하였다.

V. 결 론

본 논문에서는 검색 영역을 제 1영역과 제 2영역으로 구분하고 SAD값의 분포특성을 이용하여 SAD값이 작은 경우에는 위험이 적으므로 탐색 포인트 수를 최소화하여 제 1 영역만 탐색하였으며 SAD값이 큰 경우에는 지역극소점에 빠질 위험이 크므로 원영상과 오차를 최대한 줄이기 위해서 제 1 영역과 제 2 영역 모두 탐색하였다. 또한 적응형 임계값을 사용함으로써 영상의 유형별로 적절히 대처하여 성능이 개선되었다. 제안된 알고리즘은 실험을 통하여 기존의 여러 고속 움직임 탐색 알고리즘과 비교한 결과 탐색 포인트 수는 평균 16.4% 감소하였고, MSE값도 평균 32.83 감소하면서 성능 역시 향상한 것을 확인하였다.

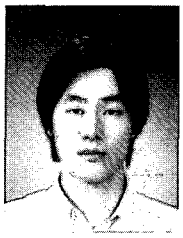
감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] ISO/IEC 11172-2. "Information Technology-Coding of Moving Picture and Associated Audio for digital Storage Media at Up to about 1.5 Mbits/sec, Part 2: Video," Aug. 1993.
[2] ISO/IEC 13818-2, "Imformation Technology -Generic Coding of Moving Picture and Associated Audio, Part 2: Video," Nov. 1995.
[3] ITU Telecom. Standardization Sector of ITU, "Video coding for audiovisual services at p x 64 kbits/s," ITU-T Recommendation H.261, 1990.
[4] ITU Telecom. Standardization Sector of ITU, "Video coding for low bitrate communication," ITU-T Recommendation H.263, March 1996.
[5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motioncompensated interframe coding for video conferencing," in Proc. NTC 81, pp. C9.6.1-9.6.5, New Orleans, LA, Nov./Dec. 1981.
[6] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 4, no. 4, pp. 438 - 443, Aug. 1994.
[7] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, pp. 313 - 317, June 1996.
[8] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 4, pp. 369 - 377, Aug. 1998.
[9] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast blockmatching motion estimation," IEEE Trans. Image Processing, vol. 9, no. 4, pp. 287 - 290, Feb. 2000.
[10] C. H. Cheung and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 12, pp. 1168 - 1177, Dec. 2002.

저 자 소 개



김 진 욱(학생회원)
2005년 가톨릭대학교 정보통신공학과 졸업.
2005년~현재 가톨릭대학교 컴퓨터공학과 석사과정
<주관심분야 : VLSI 설계, 영상처리, Motion estimation>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년~1993년 Coherent Research Inc. VLSI 설계 엔지니어.
1994년~1998년 현대전자 System IC 연구소 책임연구원.
1998년~현재 가톨릭대학교 정보통신전자공학부 부교수.
<주관심분야 : VLSI 설계, CAD, 병렬처리>