Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 1 pp. 402-410, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.1.402

이진트리 구조에 따른 구간별 탐색 후보점을 이용한 비디오 코딩의 움직임 추정

곽성근[1*]
[1]인천대학교 디자인학부

Motion Estimation in Video Coding using Search Candidate Point on Region by Binary-Tree Structure

Sung-Keun Kwak[1*]
[1]**School of Design, University of Incheon**

요 약 본 논문에서는 비디오의 시공간적인 특성과 인접 블록 움직임 벡터의 통계적 특성을 이용하는 새로운 고속 블록 정합 알고리즘을 제안한다. 비디오 시퀀스의 현재 블록의 움직임 벡터와 이전 블록의 움직임 벡터는 시간적 상관성을 갖고 있다. 제안된 알고리즘은 이전 프레임과 현재 프레임의 인접 블록으로부터 예측된 움직임 벡터와 이진트리 구조로 분할된 탐색 구간에 속하는 후보 벡터 중에서 가장 작은 SAD 값을 갖는 점을 정확한 움직임 벡터를 찾기 위한 탐색점 위치로 결정한다.
실험 결과 제안된 방식은 FS를 제외한 기존의 대표적인 고속 탐색 방식들에 비교하여 부호화 성능의 저하 없이 움직임 추정을 위한 탐색점 수 및 연산량이 급격히 감소되었다.

Abstract In this paper, we propose a new fast block matching algorithm for block matching using the temporal and spatially correlation of the video sequence and local statistics of neighboring motion vectors. Since the temporal correlation of the video sequence between the motion vector of current block and the motion vector of previous block. The proposed algorithm determines the location of a better starting point for the search of an exact motion vector using the point of the smallest SAD(sum of absolute difference) value by the predicted motion vectors of neighboring blocks around the same block of the previous frame and the current frame and the predictor candidate point on each division region by binary-tree structure.
Experimental results show that the proposed algorithm has the capability to dramatically reduce the search points and computing cost for motion estimation, comparing to fast FS(full search) motion estimation and other fast motion estimation.

Key Words : Block Matching Algorithm, Motion Estimation

1. 서 론

비디오 전송 수요가 확대되고 인터넷 및 이동 통신 등의 급격한 발전에 따라 비디오 분석 기법에 대한 관심이 고조되고 있다. 2차원 빛의 밝기 함수가 디지털 영상을 만들기 위해 표본화, 양자화가 되면 거대한 양의 데이터가 만들어진다. 이렇게 만들어진 데이터의 양은 너무 크기 때문에 결과적으로 비현실적인 저장, 처리, 통신 조건을 요구하게 된다. 영상 부호화의 목적은 적은 양의 정보로 원영상을 충실히 표현하고 재생하는데 있으며, 비디오의 빠른 전송과 효율적인 저장을 위해서 비디오 내에 존재하는 시간적, 공간적 중복성을 비디오 분석 기법을 통

*Corresponding Author : Sung-Keun Kwak (Incheon University)
Tel: +82-10-5121-3102 email: skkwak@incheon.ac.kr
Received October 11, 2012 Revised (1st November 9, 2012, 2nd November 14, 2012) Accepted January 10, 2013

해 제거함으로써 압축 효율을 증가 시키는 일이 효율적이다. 일반적으로 비디오 압축에 있어 비디오 프레임간의 움직임 벡터를 찾아 프레임간의 차를 부호화하는 움직임 추정(Motion Estimation)을 이용하여 압축 효율을 높이고 있다.

비디오 영상의 현재 블록의 움직임 벡터와 이전 블록의 움직임 벡터는 시간적 상관성을 갖고 있으므로 이를 이용하여 프레임간의 시간적 여분을 감소시켜 압축 효율을 증가시키는 것이 가능하다. 이러한 면을 고려하여 비디오 영상 압축에 있어 움직임 예측을 이용하여 영상 압축 효율을 높이고 있다. 비디오에서 인접 프레임간의 시간 간격은 매우 짧기 때문에 단위 프레임의 시간당 움직임 크기 변화량은 일반적으로 적은 범위로 제한된다고 볼 수 있다. 즉, 연속하는 두 프레임간의 움직임에 많은 시간적 중복성을 가지고 있으므로 참조 프레임의 움직임 정보를 현재 프레임의 동일한 위치 매크로블록의 탐색 시작점으로 사용함으로써 적은 탐색점들을 사용하여 움직임 벡터를 구할 수 있고, 양호한 보상 결과를 얻을 수 있다.

일반적으로 움직임 추정 기법은 영상의 각 단위로 블록 또는 화소 단위로 적용되며, 이중 계산 복잡도 및 하드웨어 구현에 있어서 용이한 블록 단위의 움직임 추정이 널리 사용되고 있다.

블록 단위로 움직임을 추정하는 대표적인 알고리즘이 전역 탐색 블록 정합 알고리즘(FS: Full Search)이다. 그러나 FS는 과정이 간단하고 예측 효율과 추정의 정확도를 고려할 때 전체적으로 좋은 특성을 가지며 하드웨어 구현이 용이하고 또한 탐색 영역의 내부 전체를 탐색하면서 가능한 모든 블록들에 대한 정합을 수행하므로 정합 오차(BDM: Block Distortion Measure)가 가장 적은 움직임 벡터를 찾을 수 있지만 많은 계산량이 필요한 단점이 있다.

이러한 FS의 단점을 극복하기 위해 속도가 개선된 TSS(Three Step Search), FSS(four Step Search), DS(Diamond Search) 및 이와 유사한 다양한 고속 블록 정합 알고리즘(FBMA: Fast Block Matching Algorithm)이 개발 되었다[7][8][9]. 이들 속도 개선 알고리즘은 주로 탐색 영역 내에서 탐색할 위치의 포인터 개수를 감소시켜 계산량의 감소를 유도하는 탐색 패턴에 따라 SAD(Sum of Absolute Difference) 계산횟수를 줄임으로써 계산량을 줄였지만, 그에 따라 화질이 열화되는 단점이 있었다.

또한 이러한 고속 블록 정합 알고리즘의 단점을 개선하기 위해 움직임 벡터간의 상관관계를 이용하여 계산량과 화질의 열화를 획기적으로 줄이게 되면 참조한 블록의 움직임 벡터들의 상관성이 떨어질 경우 압축 성능이 현저히 떨어지는 단점이 있다.

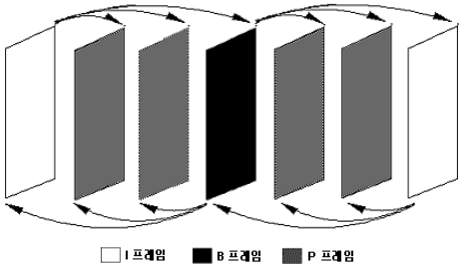
따라서 본 논문에서는 고속 블록 정합 알고리즘 중 가장 간단하면서도 효율적인 TSS를 개선한 이진트리 구조로 구간 분할을 하여 구간별 중심점을 탐색 후보로 하여 넓은 영역에 걸쳐 탐색함으로써 기존의 고속 블록 정합 알고리즘보다 탐색 속도를 향상시키고 국부적 최적에 빠지는 단점을 보완하는 탐색 방법을 제안한다. 시공간적으로 인접한 블록들은 비슷한 속도로 거의 같은 방향으로 움직인다는 점을 고려하여 이전 프레임의 같은 위치에 있는 블록의 인접 블록들의 움직임 벡터와 현재 프레임의 같은 위치에 있는 블록의 인접 블록의 움직임 벡터들을 참조하여 효과적으로 탐색점 수를 줄이고, 또한 기존의 패턴을 적용하지 않음으로써 국부적 최소(Local Minimum)로 인한 시각적인 화질 저하의 문제점과 화면내 움직임이 크고 평행이동 특성이 두드러진 영상에서 성능이 떨어지는 단점을 개선할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서 기존의 예측 탐색 알고리즘에 대해 설명하고, 제3장에서는 제안한 알고리즘을 위한 예측된 움직임 벡터의 설정 방법과 구현에 대한 방법을 논하며, 제4장에서는 실험 결과에 대하여 기존 방법과 같은 기준을 통해서 비교 분석하고, 제5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 기존의 움직임 추정 기법

2.1 움직임 추정 및 보상

비디오 시퀀스란 초당 24~30장의 연속되는 이미지 프레임들로서 하나의 화면을 만들어 나가게 된다. 움직임 평가 및 보정 단계에서는 비디오 시퀀스를 이루고 있는 각 이미지 프레임의 움직임을 분석하여 연산에 사용하기 위한 움직임 벡터를 생성해 내는 작업을 한다.



[Fig. 1] Motion Analysis of Frame Classification

프레임 분류법은 Fig. 1과 같이 연속되는 이미지 프레

임의 동작 분석을 위해서 시작되는 프레임과 끝나는 프레임 또는 장면의 변화가 두드러진 프레임을 I 프레임(Intra Frame)으로 분류하고 I 프레임들의 사이를 몇 단계로 나누어서 예측 프레임인 P 프레임(Forward Predicted Frame)으로 분류하고 I 프레임과 P 프레임 또는 P 프레임과 P 프레임의 사이에 있는 프레임을 B 프레임(Bidirectional Predicted Frame)으로 분류하여 I 프레임과 P 프레임은 프레임 자체를 분석하고 B 프레임은 앞의 두 프레임과 관계를 보간하여 동작을 분석하게 된다[3].

이때 각 프레임의 상황에 따라 매크로블록을 적절히 부호화하여 장면전환을 검출한다. 그리고 움직임 보상이 없는 경우나 해당 프레임 내에서 움직임 보상을 한 결과보다 인트라 모드로 부호화하는 것이 우수한 경우는 프레임 종류에 관계없이 매크로블록을 인트라 모드로 부호화한다[4].

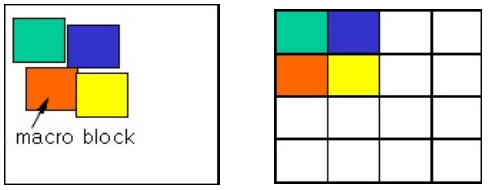
인트라 모드로 부호화된 매크로블록의 개수 θ가 많은 프레임은 장면전환이 발생했을 가능성이 많으므로 해당 프레임을 후보 컷으로 검출한다. 검출된 후보 컷의 θ는 히스토그램 차의 임계값 선택에 중요한 정보가 된다. 식 (1)은 후보 컷을 검출하기 위한 조건이다.

$$MB_{\min} \le \theta \qquad (1)$$

임계값 $MB_{\min}$는 인트라 모드로 부호화된 매크로블록의 최소 개수로 총 매크로블록 개수의 40%를 가정한다[9].

2.2 기존의 움직임 벡터의 확률적 분포

움직임 추정 기법은 영상의 각 단위로 블록 또는 화소 단위로 적용되며, 이 중에서 계산 복잡도 및 하드웨어 구현에 있어서 용이한 블록 단위의 움직임 추정이 널리 사용된다. 블록 단위의 움직임 추정은 동일한 블록 내의 화소들은 동일한 움직임을 갖는다는 것과 블록들은 수평, 수직(상하좌우)으로만 움직인다는 두 가지 전제 조건을 가지므로 영상의 한 프레임을 Fig. 2와 같이 동일한 크기의 블록들로 나누고 이들의 각 블록들에 대하여 참조 프레임(Reference Frame)의 탐색 영역 내에서 정합 오차가 가장 작은 블록이 움직임을 갖게 된다.



(a) Reference Frame (b) Current Frame

[Fig. 2] Motion Estimation for Block Matching

움직임 예측은 물체의 움직임 변화를 추정하는 방법으로 연속적인 영상 신호에서 현재 프레임의 화소들이 이전 프레임에 비해 어느 정도 움직였는지를 나타내는 움직임 벡터를 추정하여 전체 영상의 전송 대신 움직임 벡터를 전송함으로써 전송 데이터 량을 줄이는 방법이다. 이와 같이 움직임 예측은 주위 영상들의 움직임을 예측해서 부호화하는 신호를 줄임으로써 압축을 수행하는 것이다. 움직임 예측을 위해 가장 널리 사용하는 알고리즘은 블록 정합(Block Matching)이다. 이것은 연속된 두 영상 사이에서 각각의 화소를 16×16 단위로 비교하여 각 화소의 차이를 가지고 움직임을 예측하는 것이다. 현재 영상의 매크로블록과 기준 영상의 매크로블록 사이의 식 (2)와 같이 절대 평균 오차(MAD: Mean Absolute Difference)를 비교하여 MAD가 가장 작은 기준 영상의 위치를 검색하여 그 변위를 움직임 벡터로 결정한다.

$$MAD = \left(\frac{1}{M \times N}\right) \sum_{k=1}^{M} \sum_{l=1}^{N} |I_t(k,l) - I_{t-1}(k,l)| \qquad (2)$$

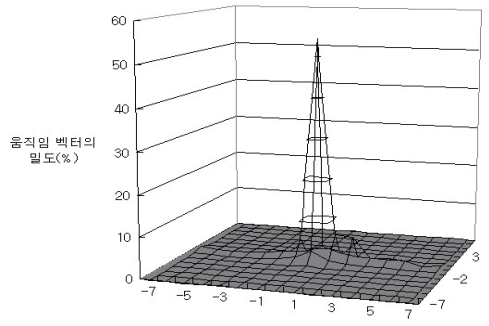
여기서 N은 영상의 가로와 세로의 각각의 크기이며, $I_t(k,l)$은 원영상의 화면을 나타내고, $I_{t-1}(k+i,l+j)$은 움직임 추정 화면을 나타낸다.

[Table 1] Average Distribution Table of Motion Vector

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
6	0.00%	0.01%	0.00%	0.00%	0.00%	0.01%	0.01%	0.03%	0.01%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%
5	0.00%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.03%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%	0.01%
4	0.01%	0.01%	0.01%	0.01%	0.01%	0.02%	0.02%	0.03%	0.01%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%
3	0.01%	0.02%	0.02%	0.02%	0.02%	0.02%	0.04%	0.11%	0.03%	0.03%	0.02%	0.01%	0.01%	0.01%	0.01%
2	0.02%	0.02%	0.02%	0.02%	0.04%	0.07%	0.16%	0.43%	0.09%	0.07%	0.04%	0.03%	0.02%	0.01%	0.01%
1	0.03%	0.03%	0.04%	0.03%	0.09%	0.23%	1.16%	2.63%	0.88%	0.27%	0.15%	0.12%	0.09%	0.07%	0.03%
0	0.33%	0.21%	0.35%	0.28%	0.58%	1.11%	4.18%	60.11%	5.47%	5.41%	1.01%	1.30%	1.08%	0.19%	0.11%
-1	0.07%	0.03%	0.07%	0.03%	0.11%	0.20%	0.54%	1.92%	0.55%	0.16%	0.09%	0.04%	0.03%	0.02%	0.02%
-2	0.03%	0.04%	0.04%	0.03%	0.04%	0.08%	0.10%	0.38%	0.07%	0.06%	0.04%	0.02%	0.02%	0.02%	0.02%
-3	0.00%	0.01%	0.01%	0.01%	0.02%	0.03%	0.12%	0.16%	0.03%	0.02%	0.03%	0.01%	0.01%	0.01%	0.01%
-4	0.00%	0.00%	0.01%	0.01%	0.01%	0.02%	0.07%	0.14%	0.01%	0.01%	0.01%	0.02%	0.01%	0.01%	0.00%
-5	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.02%	0.07%	0.01%	0.01%	0.00%	0.01%	0.00%	0.00%	0.00%
-6	0.00%	0.01%	0.00%	0.01%	0.01%	0.01%	0.02%	0.06%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
-7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.07%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%

실제 6개의 실험 비디오 시퀀스로부터 구한 평균 움직임 벡터의 분포를 살펴보면, Table 1과 같이 대부분 중심 부근에 집중되어 있다. 특히 수평과 수직선상으로 많은 움직임이 있음을 알 수 있다. Table 1에서 움직임 벡터가 p=±2인 중앙 5×5 화소 내에서 찾아질 확률이 약 81.80%가 된다.

Fig. 3은 FS 알고리즘을 이용한 실험 영상의 움직임 벡터 분포의 예를 보인 것으로 실제 6개의 비디오 시퀀스로부터 구한 움직임 벡터의 평균 분포로 대부분 중심 부근에 집중되어 있으며, 특히 수평과 수직선상으로 많은 움직임이 있음을 알 수 있다.

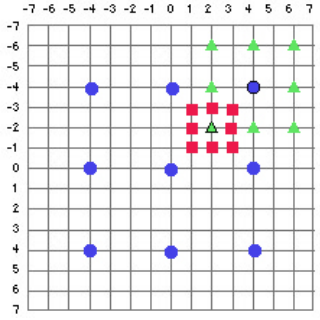


[Fig. 3] Average Distribution Chart of Motion Vector of All Experimental Sequence

2.3 기존 TSS 기법의 문제점

3단계 탐색(TSS: Three-Step Search)은 넓은 영역에 걸쳐 몇 개의 탐색점을 조사한 후, 점차 범위를 좁혀 나가는 방식으로 가장 간단하면서도 효율적인 방법이지만 모든 영상에 대해서 고정된 패턴을 적용하기 때문에 움직임이 거의 없는 정적 블록인 경우는 문제가 없지만 움직임이 많은 동적 블록의 경우 첫 번째 탐색이 잘못될 때는 국부적 최적에 빠질 수 있는 단점을 가지고 있다[9].

3단계 탐색 기법은 Fig. 4와 같이 먼저 4화소 간격의 9×9의 9개의 탐색점을 가진 사각형 블록의 형태로 분포된 탐색점들로 구성하여 최소 정합 오차를 갖는 탐색점을 중심으로 탐색점간의 간격을 이전 단계보다는 절반으로 줄인 2화소 간격으로 8개의 탐색점에 대해 최소 정합 오차를 갖는 탐색점을 구한다. 그리고 마지막 단계에서는 전 단계에서 구한 최소 정합 오차 값을 갖는 탐색점을 중심으로 1화소 간격으로 8개의 탐색점에 대해 최소 정합 기준 오차를 갖는 탐색점을 구하여 최종적으로 해당 블록에 대한 움직임 벡터로 결정한다[5]. TSS 기법의 탐색 경로는 Fig. 4와 같다.



[Fig. 4] Search Path of TSS

3. 제안한 움직임 추정 알고리즘

제안하는 알고리즘은 TSS 기법에 기초하고 있다. TSS는 넓은 영역에 걸쳐 몇 개의 탐색점을 조사한 후, 점차 범위를 좁혀 나가는 방식으로 가장 간단하면서 효율적인 탐색 알고리즘이다.

기존의 탐색 알고리즘이 예측된 움직임 벡터로 탐색원점을 이동시켜 탐색을 수행함으로써 블록 내에 공존하는 최소 정합 오차를 가지는 탐색점을 제외시켜 잘못된 추정을 하는 문제점을 개선하기 위해 보다 넓은 영역으로 분할된 탐색 구간에 속하는 탐색 후보점에서 가장 작은 정합 오차를 갖는 점을 탐색원점으로 하여 탐색을 수행하는 알고리즘을 제안한다.

일반적으로 비디오 시퀀스는 인접 영상과 현재 부호화 영상 사이에 높은 상관성을 보인다. 이러한 특성으로 인해 움직임 벡터의 예측이나 움직임 추정 등이 이루어지는데, 기존의 움직임 추정 방식은 실제 최적의 움직임 벡터가 위치하는 곳이 원점에서 멀리 떨어진 곳이 아니면 불필요한 위치를 탐색하여 연산량을 증가시키는 결과를 나타낸다. 이와 같이 불필요한 연산을 감소시키기 위해 이진트리 구조를 이용하여 인접 블록의 움직임 벡터에 따라 적정한 영역 내에서만 움직임 추정을 수행함으로써 탐색 지점 감소를 통한 연산량 절감이 가능하다.

실험 비디오 시퀀스들에 대해 탐색할 매크로블록에서 움직임 벡터가 Table 1과 같이 p=±2의 중앙 5×5 화소 내에서 찾아질 확률이 높은 점을 이용한다. 이 범위내의 영역을 움직임 존재 가능 위치로 설정하여 이를 중심으로 움직임 벡터가 존재할 탐색 구간을 중심에서 Fig. 5와 같이 이진트리 구조로 단계적으로 구간을 설정한다. 이렇게 얻어진 각 구간 내에 정확한 움직임 벡터가 있다고 가정한다. 따라서 움직임 벡터를 빠르게 찾아내기 위해서는 주어진 각각의 구간에서 첫 번째 예측 후보점을 설정하여 탐색 속도와 성능을 향상시킴으로써 서로 다른 초기점 조건을 이용하여 최소 정합 오차 후보에 따라 후보 벡터를 존재 가능한 방향으로 탐색을 수행함으로써 가장 적합한 움직임 벡터를 찾으므로 화질 및 속도 개선을 하는 방법을 연구한다.

3.1 이진트리 구조를 적용한 탐색 구간 분할

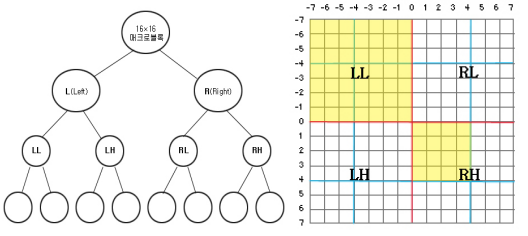
움직임 추정으로 보상이 가능한 부분은 움직임이 없는 배경과 같은 부분이나 움직임이 적은 저주파 부분이고 움직임이 많은 고주파 부분은 움직임 추정으로 보상되기 어렵다.

탐색 구간 설정을 위해서는 매크로블록의 움직임 방향

과 크기를 고려하여야 하는데 이것은 이전 블록의 움직임 벡터를 이용하여 얻을 수 있다. 탐색창의 분할에 있어서 Fig. 5와 같이 탐색 영역(w=±7)에서 왼쪽(L)과 오른쪽(R)으로 각각 2등분한 후, 각각의 공간에 대해 세로 방향으로 H(High)와 L(Low)로 나누어 LL(Left-Low), RL(Right-Low), LH(Left-High), RH(Right-High)로 4등분한다. 여기서 움직임 벡터를 빠르게 찾아내기 위해서 주어진 각각의 구간별 4개의 중심점과 움직임 벡터의 중심 지향적인 분포 특성을 고려하여 탐색 영역의 중심점 (0, 0)을 포함한 탐색 후보점 중에서 최소 정합 오차 값을 갖는 탐색점을 찾은 후에 최소 정합 오차 값을 중심으로 다시 이진트리 구조로 4등분 분할하여 각 구간의 원점을 탐색 후보점으로 하여 이를 반복한다.

제안한 알고리즘에서는 움직임 벡터가 탐색 영역의 중심에 분포된다는 사실을 이용하여 5개의 초기 탐색 후보점에 대한 정합 기준 오차값을 계산하여, 정적 블록과 동적 블록의 빠른 식별을 위해 중간-정지(Half-Stop) 기술을 이용하여 이들 블록의 움직임을 추정한다.

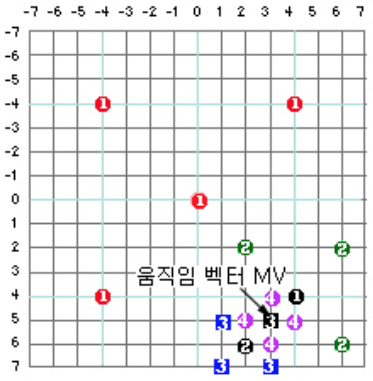
기존의 블록 정합 알고리즘들은 항상 탐색 영역의 원점에서 탐색을 시작하는데 비해, 제안한 알고리즘은 이전 프레임과 현재 프레임간의 인접 블록에서 변화한 움직임의 크기를 현재의 매크로블록에 적용한 좌표에서부터 탐색을 실시함으로써 움직임 벡터가 존재할 잠재적 탐색 영역을 예측할 수 있어 속도와 화질의 뚜렷한 개선이 가능하다. 즉, 탐색 시작점을 더욱 정확하게 예측할 수 있게 되면 속도면에서 뚜렷한 효과를 얻을 수 있을 뿐 아니라, 임계값과 조기 종료 조건을 줄일 수 있어 높은 화질을 이끌어 낼 수 있기 때문이다.



(a) Tree Structure (b) Division Region for Search

[Fig. 5] Division Region of Binary Tree Structure

이전 프레임 블록의 움직임 벡터의 이동 거리를 고려한 이 방식은 연속된 프레임간 움직임 벡터의 변화가 커질 경우에 탐색 후보점이 탐색창의 경계 범위를 벗어날 수 있으므로 이러한 문제를 전처리할 필요가 있다.



[Fig. 6] A Sample of Search Path and Point

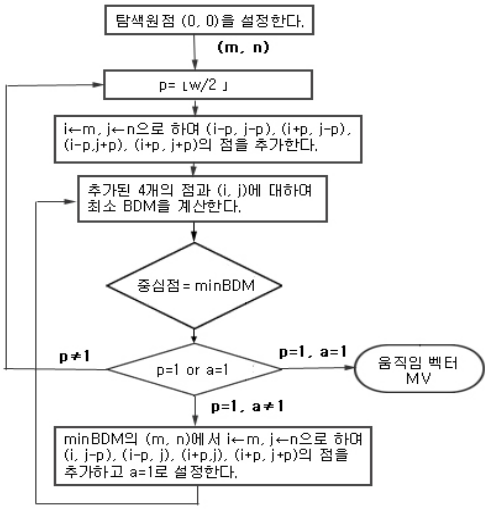
3.2 제안한 탐색 알고리즘에 의한 움직임 추정

블록 정합을 수행할 때, 블록간의 상관도나 정합 기준값을 계산하고 그 중에서 탐색 오차가 작은 움직임 벡터 후보를 찾는 것이 우선이다. 통계적으로 움직임이 탐색 중심에 많이 분포하기 때문에 탐색 영역의 중심으로부터 가까운 거리에 있는 탐색점을 우선적으로 고려하기 쉽다.

상관도나 정합 기준 값이 비슷한 경우에 탐색 영역 중심으로부터 거리가 가까운 탐색점을 최종 움직임 벡터로 결정하지만 우선순위는 정합 기준 값이 되어야 한다. 다시 말하면, 정합 기준 값이 비슷하다고 예상되는 탐색점 위치들 중에서 움직임 벡터의 후보를 결정하는 것이 타당하다. 상관도의 관점으로 볼 때, 정사각형 블록을 사용하는 블록 정합의 경우에 탐색점들의 탐색 영역 중심으로부터 거리와 상관도의 관계는 일치하지 않으며, 실제의 경우에도 거리적으로 동일한 위치에 있을 때 상관도가 같다고 확신할 수 없다.

따라서 제안된 기법은 TSS와 같이 넓은 영역에 걸쳐 움직임 존재 가능 위치를 고려하여 적응적으로 탐색함으로써, 효율적으로 더 정확한 움직임을 추정할 수 있다는 장점이 있다.

매크로블록 16×16에서 화소 간격이 최대 움직임 변위 w에 해당하는 영역만큼을 각각 이진트리 구조로 분할하여 그 영역의 중심점인 4개의 탐색점들로 구성하여 탐색 단계 크기가 1이 될 때까지 반복하여 탐색 영역을 이진트리로 분할하여 수행한다.



[Fig. 7] Flowchart of Motion Vector Estimation of Proposal Algorithm

Fig. 7은 제안한 움직임 벡터 추정 알고리즘의의 순서를 나타낸 흐름도로 이진트리 구조로 분할한 탐색 영역의 중심점들의 최소 정합 오차를 비교하여 더 작은 값을 갖는 탐색점을 중심으로 이진트리 구조로 세분하여 세분된 각 영역의 중심으로 추가된 4개를 포함한 5개의 탐색 후보점에 대하여 다음과 같은 알고리즘을 수행하여 움직임 벡터를 추정한다.

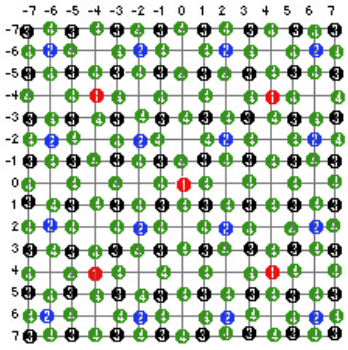
1단계: 이진트리 구조로 분할된 각 영역의 중심점인 화소 간격이 p= ⌊w/2⌋ 인 (i-p, j-p), (i+p, j-p), (i-p, j+p), (i+p, j+p)의 값을 갖는 각 구간의 중심점인 4개의 좌표점와 탐색 원점 (0, 0)을 포함한 5개의 탐색 후보점의 위치를 구한다.

2단계: 5개의 탐색점에 대하여 최소 정합 오차를 계산한다.

3단계: p= ⌊w/2⌋ 를 설정한다. p=1이면 4단계로 가고, 아니면 이전 단계의 최소 정합 오차점을 중심점으로 하여 (i-p, j-p), (i+p, j-p), (i-p, j+p), (i+p, j+p)의 4개의 탐색점들과 최소 정합 오차를 계산한다. 이때 중심점이 최소 정합 오차이면 4단계로 가고, 아니면 최소 정합 오차점을 중심점으로 하여 3단계를 반복한다.

4단계: 이전 단계의 최소 정합 오차점을 중심점으로 하여 (i-p, j), (i+p, j), (i, j-p), (i, j+p)의 4개의 점들을 포함하여 5개의 탐색점에 대하여 최소 정합 오차를 계산한다.

5단계: 이전 단계에서 구한 최소 정합 오차점이 움직임 벡터의 최종 해가 된다.



[Fig. 8] Search Sequence of Proposal Algorithm

4. 실험 결과

제안된 기법의 성능을 평가하기 위하여 Fig. 9와 같이 움직임이 아주 적은 Mother & Daughter와 움직임이 있는 Foreman과 Coastguard, 움직임이 큰 Stefan과 Flower Garden, Football의 6개의 실험 비디오 시퀀스에 대해 각각 80프레임씩을 대상으로 실험하였다.

실험은 MPEG-4 VM 인코더를 이용하여 이루어졌으며, 실험에 사용한 영상은 QCIF (176 QCIF 144), CIF(352×288)의 2종류이며, 움직임 추정에 사용된 매크로블록의 크기는 QCIF 영상에 대해 16×16 화소를, CIF 영상에 대해 32×32 화소를 사용하였다. 탐색 영역의 변위는 각각 ±7, ±15로 설정하고, Bitrate 1024k에 대해 실험하였다.

비교 탐색 기법으로는 FS, DS, TSS, 그리고 제안한 탐색 기법을 사용하였다.

(a) “Mother & Daughter" Video Sequence

(b) “Foreman" Video Sequence

(c) “Coastguard" Video Sequence

(d) “Stefan" Video Sequence

(e) “Flower Garden" Video Sequence

(f) “Football" Video Sequence

[Fig. 9] Experimental Video Sequence

블록 정합의 정도를 평가하기 위해 대표적인 정합 기준인 평가 함수(Cost Function)로 영상 화질의 품질을 평가하기 위한 식 (3)과 같은 평균 제곱 오차(MSE: Mean Squared Error), 식 (4)의 평균 절대값 오차(MAD: Mean Absolute Difference)와 정합 오차 측정 함수로는 식 (5)의 절대값 오차의 합(SAD: Sum of Absolute Difference)을 이용하였다. 또한 제안하는 기법의 성능 향상을 측정하기 위해 블록 당 탐색점의 개수를 기존 방법들과 비교하였다.

$$MSE(i,j)=(\frac{1}{N^2})\sum_{k=1}^{N}\sum_{l=1}^{N}[I_t(k,l)-I_{t-1}(k+i,l+j)]^2 \quad (3)$$

$$MAD(i,j)=(\frac{1}{N^2})\sum_{k=1}^{N}\sum_{l=1}^{N}[I_t(k,l)-I_{t-1}(k+i,l+j)] \quad (4)$$

$$SAD(i,j)=\sum_{k=1}^{N}\sum_{l=1}^{N}[I_t(k,l)-I_{t-1}(k+i,l+j)] \quad (5)$$

그리고 화질의 평가를 위한 PSNR은 식 (6)과 같다.

$$PSNR=10\log_{10}(\frac{255^2}{MSE}) \quad (6)$$

Table 2와 Table 3은 각각의 실험 비디오 시퀀스의 처음 80 프레임을 부호화한 결과이다. Table 2는 기존의 탐색 기법과 제안한 알고리즘의 각 블록에 대한 평균 PSNR 값을 비교한 결과이며, Table 3은 각 블록에 대한 평균 탐색횟수에 대한 비교 결과를 나타낸다.

[Table 2] Comparison of PSNR of Search Algorithm

Sequence	FS	DS	TSS	**Proposed**
Mother&Daughter	40.204	40.126	40.084	40.137
Foreman	33.456	33.175	32.789	33.275
Coastguard	29.642	29.409	29.322	29.539
Stefan	25.693	24.655	25.278	25.340
FlowGarden	23.994	23.249	23.249	23.867
Football	22.413	21.874	22.090	22.186
Average	29.23	28.75	28.80	29.06

[Table 3] Comparison of Search Point Number of Search Algorithm

Sequence	FS	DS	TSS	**Proposed**
Mother&Daughter	184.56	21.51	23.17	15.01
Foreman	204.28	15.90	23.29	14.66
Coastguard	204.28	23.27	23.27	16.26
Stefan	204.28	23.30	25.72	16.57
FlowGarden	202.05	23.24	24.35	16.30
Football	202.05	23.09	24.12	15.30
Average	200.25	21.72	23.99	15.68

이러한 결과를 본다면 제안한 방식이 FS에 비해 평균 탐색점 수가 약 78% 정도 감소하였고, 탐색 속도면에서는 약 13배 정도의 성능 향상을 나타내었으며, 기존의 탐색 기법에 비해 탐색점 수가 감소하여 탐색 속도가 향상되었다. 또한 화질면에서 FS를 제외한 기존의 방식보다 우수함을 볼 수 있다. 즉, 평균 PSNR 값의 6개의 실험 영상에 대한 평균이 제안한 방식이 29.06[dB]으로 나타나 29.23[dB]인 FS 다음으로 우수하였다. 그리고 실험에 사용한 영상별로 볼 때, 움직임이 아주 작은 Mother & Daughter 시퀀스는 영상의 프레임간의 시간적 상관성이 많이 존재하는 제안된 기법이 탐색점 수가 가장 작아 속도면에서 가장 우수하고, FS에 비해 탐색점 수를 82.7% 감소시키면서도 PSNR 값이 거의 근접하였으며, 또한 다른 모든 기법에 비해 움직임 추정에 필요한 탐색점 수를 약 71.3% 정도 감소시키면서도 PSNR 값이 가장 우수하였다.

그보다 움직임이 있는 Foreman 시퀀스에서는 FS가 제안된 기법보다 0.181[dB] 정도 우수하나 탐색점 수를 13.9배 이상 사용되었고, 움직임이 큰 Stefan, Flower Garden 시퀀스에서는 FS가 각각 0.356[dB], 0.127[dB]정

도가 좋으나 역시 12배 이상의 많은 탐색점 수를 사용하였다.

이상의 실험 결과에 의하면, PSNR 측면에서 FS와 유사한 값으로 이를 제외한 모든 기법에 비해 제안된 방식이 가장 우수하였으며, 속도면에서는 제안된 방식이 가장 우수하게 나타났다.

5. 결 론

본 논문에서는 기존의 고속 블록 정합 알고리즘이 예측된 움직임 벡터로 탐색 원점을 이동시켜 탐색을 수행함으로써 블록 내에 공존하는 최소 정합 오차를 가지는 탐색점이 제외되어 화질 저하를 초래하는 문제점을 개선하기 위해 이진트리 구조로 분할된 탐색 영역의 중심에 가깝게 정의한 탐색 구간별 후보 벡터 중에서 가장 작은 SAD 값을 갖는 점을 탐색 원점으로 하여 탐색을 수행하는 알고리즘을 제안하였다. 실험의 결과를 보면 제안된 기법이 화질면에서는 전역 탐색에 근접한 우수한 성능을 보였으며, 기존의 고속 블록 정합 기법보다 적은 탐색점 수를 사용하면서도 화질면에서는 나은 결과를 보였다. 이 경우 FS를 제외한 다른 기법과 비교하였을 때 움직임 예측면에서는 평균적으로 0.26~0.31[dB] 정도의 성능 향상을 보였다.

움직임 추정시 본 논문에서 제안한 탐색 기법을 사용하여 움직임 추정을 한다면 보다 빠르게 움직임 벡터를 찾을 수 있을 것이며, 탐색 패턴의 도출 없이, 이진트리 구조로 분할되는 탐색 영역의 중심점만으로 탐색이 가능함으로써 다른 고속 블록 정합 방법들보다 탐색점 수를 감소시키고, 우수한 보상 결과를 얻을 수 있을 것으로 기대된다. 또한 일정한 시간 간격을 두고 연속적으로 추출되는 움직임 벡터의 크기를 일정하게 조절할 수 있는 시간적인 동기화에 대한 연구도 보강될 필요가 있다.

움직임 벡터의 예측 과정에서 발생할 수 있는 예외적인 요소를 모두 고려하여 가장 적응적인 현재 프레임의 매크로블록 내에서 움직임 벡터를 구간별로 탐색하므로 본 연구에서 제안한 탐색 기법을 사용하여 움직임 예측을 한다면 보다 빠르게 움직임 벡터를 찾을 수 있을 것이며, 또한 비디오 압축의 성능을 향상시키고 우수한 보상 결과를 얻을 수 있을 것으로 기대된다.

References

[1] Shih-Hao Wang, Shih-Hsin Tai, Tihao Chiang, "A Low-Power and Bandwidth-Efficient Motion Estimation IP Core Design Using Binary Search", Circuits and Systems for Video Technology, IEEE Transactions on, pp.760-765, May 2009.
DOI: http://dx.doi.org/10.1109/TCSVT.2009.2017416

[2] Goela, N., Wilson, K., Feng Niu, "An SVM Framework for Genre-Independent Scene Change Detection", Multimedia and Expo, 2007 IEEE International Conference on, pp.532-535, Jul, 2007.
DOI: http://dx.doi.org/10.1109/ICME.2007.4284704

[3] W. A. C. Fernando, "Sudden Scene Change Detection in Compressed Video using Interpolated Macroblocks in B-frames", Multimedia Tools and Applications, Vol. 28, No.3, pp.301-320, May, 2006.
DOI: http://dx.doi.org/10.1007/s11042-006-7716-7

[4] X Yi, N Ling, "Fast Pixel-Based Video Scene Change Detection", Circuits and Systems, IEEE International Symposium on, Vol.4, pp.3443-3446, May, 2005.
DOI: http://dx.doi.org/10.1109/ISCAS.2005.1465369

[5] Saez, E., Benavides, J. I., Guil, N., "Reliable real time scene change detection in MPEG compressed video", Multimedia and Expo, 2004. IEEE International Conference on, Vol. 1, pp.567-570, June, 2004.
DOI: http://dx.doi.org/10.1109/ICME.2004.1394255

[6] C. H. Cheung, L. M. Po, "A Novel Cross-Diamond Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Transactions on Circuits & System for Video Tech., Vol. 12, No. 12, pp.1168-1177, Dec., 2002. DOI: http://dx.doi.org/10.1109/TCSVT.2002.806815

[7] A. M. Tourapis, O. C. Au, M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation", IEEE Transactions on Circuits & System for Video Tech., Vol. 12, No. 10, pp.934-947, Oct., 2002. DOI: http://dx.doi.org/10.1109/TCSVT.2002.804894

[8] L. M. Po, W.C. Ma, "A Novel Four-Step Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Transactions on Circuits & System for Video Tech., Vol. 6, No. 3, pp.313-317, June 1996.
DOI: http://dx.doi.org/10.1109/76.499840

[9] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated Interframe Coding for Video Conferencing", in Proc. National Telecommunications Conf., New Orleans, LA, pp.G5.3.1-G5.3.5, Nov. 1981.

곽 성 근(Sung-Keun Kwak) **[정회원]**

- 1980년 2월 : 연세대학교대학원 전자공학과 (공학석사)
- 2004년 8월 : 아주대학교대학원 컴퓨터공학과 (공학박사)
- 1980년 3월 ~ 2010년 2월 : 인천전문대학 컴퓨터애니메이션과 교수
- 2010년 3월 ~ 현재 : 인천대학교 디자인학부 교수

<관심분야>
컴퓨터그래픽스, 동영상압축, 컴퓨터애니메이션