움직임 벡터 예측 후보들과 적응적인 탐색 패턴을 이용하는 블록 정합 알고리즘

곽 성 근[†] · 위 영 철[††] · 김 하 진[††]

요 약

본 논문에서는 영상의 시공간적인 특성과 움직임 벡터의 중심 지향적 분포 특성을 이용하는 예측 탐색 알고리즘을 제안한다. 제안된 알고리즘은 이전 프레임 블록으로부터 예측된 움직임 벡터, 분할된 탐색 구간에 속하는 후보 벡터와 현재 프레임 블록의 이웃 블록으로 예측된 움직임 벡터 중에서 가장 작은 SAD 값을 갖는 점을 정확한 움직임 벡터를 찾기 위한 초기 탐색점 위치로 결정한다. 그리고 초기 탐색점 위치로 이동하여 움직임 크기에 따라 적응적인 탐색 패턴으로 탐색을 수행한다. 실험 결과 제안된 방식은 FS를 제외한 기존의 대표적인 고속 탐색 방식들에 비해 PSNR 값에 있어서 평균적으로 0.05~0.34dB 개선되고 영상에 따라 최고 0.75dB 정도 우수한 결과를 나타내었다.

A Block Matching Algorithm using Motion Vector Predictor Candidates and Adaptive Search Pattern

Sung-Keun Kwak[†] · Youngcheul Wee[††] · Ha-Jine Kim[††]

ABSTRACT

In this paper, we propose the prediction search algorithm for block matching using the temporal/spatial correlation of the video sequence and the center-biased property of motion vectors. 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 vector from the same block of the previous frame and the predictor candidate point on each search region and the predicted motion vector from the neighbour blocks of the current frame. And the searching process after moving the starting point is processed a adaptive search pattern according to the magnitude of motion vector. Simulation results show that PSNR(Peak-to-Signal Noise Ratio) values are improved up to the 0.75dB as depend on the video sequences and improved about 0.05~0.34dB on an average except the FS(Full Search) algorithm.

키워드 : 예측 후보점(Predictor Candidate Point), 블록 정합 알고리즘(Block Matching Algorithm)

1. 서 론

참조 프레임에서 현재 블록의 움직임 추정(ME : Motion Estimation)을 위한 블록 정합 알고리즘(BMA : Block-Matching Algorithm)은 알고리즘의 단순성과 고속 처리를 위한 하드웨어적인 접근과 용이성으로 인해 MPEG 계열이나 H.261, H.263, H.264 등의 동영상 부호화 표준으로 광범위하게 채택되어 사용되고 있다.

가장 단순한 블록 정합 알고리즘은 영상을 모양과 크기가 동일한 사각형 블록으로 분할한 후, 블록 단위의 특징을 정의하고 정의된 특징에 의해 구성되는 정합 척도를 일정한 탐색 영역 내의 블록들에 적용하여 가장 높은 정합 척도를 가지는 블록을 찾는 FS(Full Search)이다. 그러나 전역 탐색 블록 정합 알고리즘은 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 오차가 가장 적은 움직임 벡터를 찾을 수 있지만 많은 계산량이 필요한 단점이 있다.

이러한 전역 탐색법의 단점을 극복하기 위해 속도가 개선된 TSS(Three Step Search)[1], FSS(Four Step Search)[3], BBGDS(Block-Based Gradient Descent Search)[4], DS(Diamond Search)[6], HEXBS(Hexagon-based Search)[7], CHS(Cross and Hexagonal Search)[11] 등의 다양한 고속 블록 정합 알고리즘(FBMA : Fast Block Matching Algorithm)이 개발 되었다. 이들 속도 개선 알고리즘은 주로 탐색 영역 내에서 탐색할 위치의 포인터 개수를 감소시켜 계산량의 감소를 유도하는 탐색 패턴을 사용한다. 탐색

† 정 회 원 : 인천전문대학 컴퓨터정보과 교수
†† 정 회 원 : 아주대학교 컴퓨터공학과 교수
논문접수 : 2004년 2월 12일, 심사완료 : 2004년 3월 31일

패턴이란 블록 정합을 위해 각 탐색 단계에서 정합 오차를 검사하는 탐색점들을 의미하며, 이 탐색점들 중에서 최소 정합 오차를 가지는 위치를 중심으로 다음 단계의 움직임 탐색이 수행된다. 따라서 고속 블록 정합 움직임 탐색 방법에서 사용되는 탐색 패턴은 그 모양과 크기에 따라 탐색의 속도와 성능을 좌우하는 중요한 요소가 될 수 있다. 그러나 이러한 고속 블록 정합 알고리즘은 계산량을 줄이기 위해 탐색 영역에 포함되는 특정한 패턴들의 몇몇 점들만 조사하여 움직임 벡터를 찾기 때문에 국부적인 탐색을 하게 되거나, 해당 블록에 대한 움직임 정보를 가지고 있지 않기 때문에 항상 탐색 영역의 원점에서부터 움직임을 추정하는 과정에서 일부 탐색점들을 블록 정합 대상에서 제외시키거나 부정확한 방향으로 움직임을 추정함으로써 복원된 영상의 화질을 저하시키는 단점이 있다.

이러한 기존의 고속 블록 정합 알고리즘의 단점을 개선하기 위해 이전 프레임의 같은 위치의 블록 또는 인접한 이전 블록들의 움직임 벡터들의 정보를 이용하는 예측 탐색 알고리즘(PSA : Prediction Search Algorithm)이 제안되었다[5, 8, 9, 10]. 이 방식은 이전 블록들의 움직임 정보를 이용함으로써 보다 정확한 움직임 벡터를 찾으려는 것이다. 그러나 이 방식은 참조한 블록의 움직임 벡터들의 상관성이 떨어질 경우 압축 성능이 현저히 떨어지는 단점이 있다.

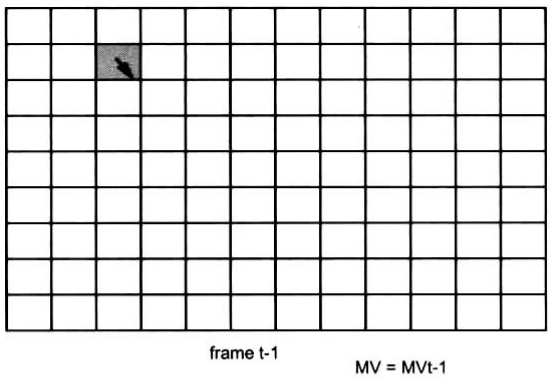
일반적으로 움직임이 있는 동영상의 경우 짧은 시간에 영상의 움직임이 있더라도 그 시간에는 많은 움직임이 있을 수 없음으로 인해서 움직임 벡터는 중심 지향적인 분포 특성을 갖게 된다. 또한 현재 매크로 블록의 움직임 벡터는 이전 프레임의 같은 위치에 있는 블록들과 상관관계가 높으며, 비슷한 운동을 한다.

따라서 본 논문에서는 이전 프레임 블록과 현재 프레임 블록의 움직임 정보를 이용하여 현재 영상의 움직임 후보 벡터를 각각 구한다. 또한 영상의 국부적인 변화에 따른 움직임을 효과적으로 반영하기 위해 예측된 움직임 벡터가 속하는 탐색 구간에 따라 중심 지향적인 분포 특성에 알맞은 후보 벡터를 또 하나 설정한다. 그리고 3개의 후보 벡터에 대한 정합 오차(BDM : Block Distortion Measure)를 비교하여 더 작은 값을 갖는 후보 벡터로부터 탐색을 수행하여 움직임 벡터의 상관성을 높임으로써 효과적으로 탐색점수를 줄이고, 또한 예측되는 움직임 벡터의 크기에 따라 적응적으로 탐색 패턴을 달리 함으로써 속도 및 화질을 개선하는 예측 탐색 기법을 제안한다.

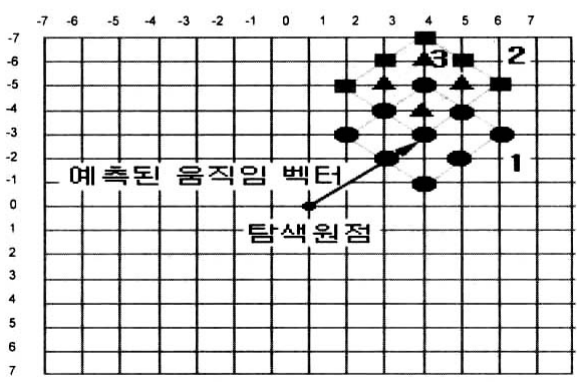
본 논문의 구성은 다음과 같다. 제2장에서 기존의 예측 탐색 알고리즘에 대해 설명하고, 제3장에서는 제안한 알고리즘에 대해 기술하였다. 그리고 제4장에서는 실험 결과에 대하여 기존 방법과 같은 기준을 통해서 비교 분석하고, 제5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 기존의 예측 탐색 알고리즘

동영상에서 인접한 프레임간의 시간 간격은 매우 짧기 때문에 단위 프레임의 시간당 움직임 크기 변화량은 일반적으로 적은 범위로 제한되며, 공간적으로 인접한 블록들도 비슷한 속도로 거의 같은 방향으로 움직인다고 볼 수 있다. 즉, 현재의 매크로 블록의 움직임은 주위 블록 또는 이전 프레임의 같은 위치에 있는 블록들과 상관관계가 높으며 비슷한 운동을 하므로 이러한 움직임 정보를 현재 프레임의 동일한 위치 매크로 블록의 탐색 초기점으로 사용함으로써 적은 탐색점을 이용하여 움직임 벡터를 구할 수 있고, 양호한 보상된 결과를 얻을 수 있다. 최근에는 이와 같이 이전에 탐색된 움직임 벡터들의 정보를 이용하는 예측 탐색 알고리즘(PSA : Prediction Search Algorithm)이 제안되고 있으며, 이러한 예측 알고리즘은 현재 블록과 같은 위치에 있는 이전 시점의 블록과의 시간적 상관관계를 이용하는 TPMV(Temporal Predicted Motion Vector)와 현재 프레임에서 추출된 주변 블록들의 움직임 벡터와 같은 공간적 상관관계를 이용하는 SPMV(Spatial Predicted Motion Vector), 그리고 이 두 방법을 모두 이용하는 TSPMV(Temporal/Spatial Predicted Motion Vector)로 크게 세 가지로 분류할 수 있다.



(a) 이전프레임의 움직임 벡터



(b) 탐색 경로의 예

(그림 1) TPS의 예측 방법과 탐색 경로

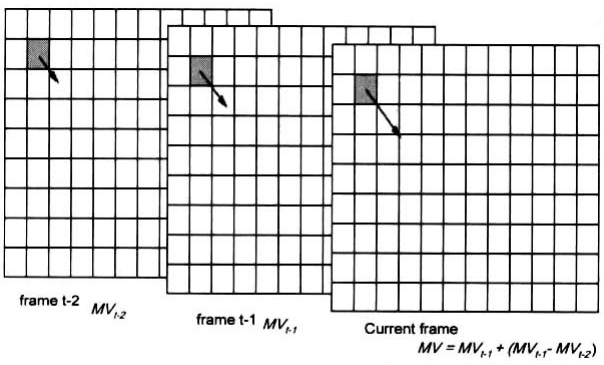
이전 프레임과의 시간적 상관관계를 적용하는 방법은 영상의 움직임은 급격히 변화하기보다는 완만하게 변화한다는 데 점[2]에 근거하여 (그림 1)(a)와 같이 이전 프레임의 같은 위치에 있는 블록의 움직임 벡터를 참조하는 TPS(Temporal Predicted Search)가 제안되었다. 이 방법은 탐색 영역에서 탐색 원점의 위치를 참조 프레임의 예측된 움직임 벡터(Predicted Motion Vector)만큼 이동시켜 새로운 탐색 원점을 설정한 후 블록 정합을 수행한다.

$$MV_{Predicted} = MV_{t-1} \quad (1)$$

(그림 1)(b)는 예측된 움직임 벡터 $MV_{Predicted}$로 탐색 원점을 이동시켜 다이아몬드 패턴으로 블록 정합을 하는 TPS의 탐색 경로의 예를 보인 것이다.

그리고 A. Tourapis[8]는 지역 탐색법(Zonal Search)에서 연속된 프레임간 움직임의 변화가 서로 상관성이 있다는데 착안하여 (그림 2)와 같이 이전 두 개의 t-1 시점 프레임과 t-2 시점 프레임의 같은 위치에 있는 블록의 움직임 벡터의 변화를 참조하여 움직임 벡터 $MV_{Predicted}$를 식 (2)와 같이 예측 적용하였다. 이전 프레임 블록의 움직임 벡터의 이동 거리를 조사하여 그 이동 거리를 적응적으로 고려한 이 방식은 연속된 프레임간 움직임 벡터의 변화가 커질 경우에 $MV_{Predicted}$가 탐색 영역의 경계 범위를 벗어날 수 있으므로 이러한 문제를 전처리할 필요가 있다.

$$MV_{Predicted} = MV_{t-1} + (MV_{t-1} - MV_{t-2}) \quad (2)$$

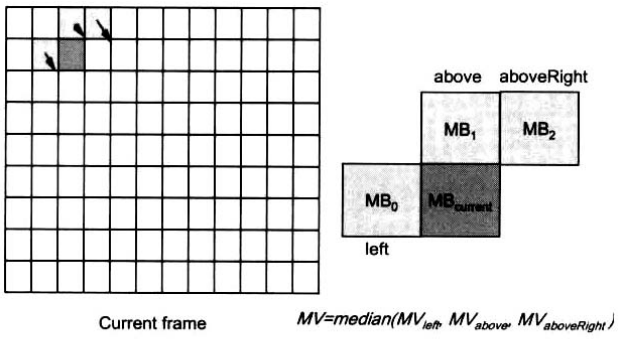


(그림 2) 참조하는 이전 프레임의 같은 위치의 매크로 블록

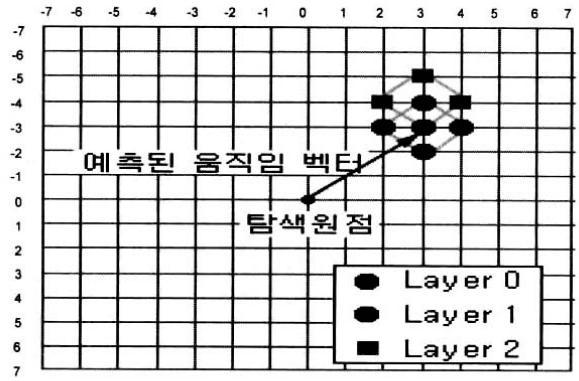
현재 프레임의 블록들간의 공간적인 상관관계를 반영하는 방법은 공간적으로 인접한 매크로 블록들은 비슷한 속도로 거의 같은 방향으로 움직인다는 점을 고려하여 M. Gallant[5]는 NNS(Nearest Neighbour Search)에서 (그림 3)(a)와 같이 수평, 수직 그리고 대각 방향에 있는 매크로 블록 MB_0, MB_1, MB_2을 지원 영역(ROS : Region Of Support)으로 현재 블록의 예측 움직임 벡터를 식 (3)과 같이 이들 인접 매크로 블록 각각의 움직임 벡터 MV_0, MV_1, MV_2들의 중간 값으로 제안하였다.

$$MV_{Predicted} = median(MV_0, MV_1, MV_2) \quad (3)$$

(그림 3)(b)는 예측된 움직임 벡터 $MV_{Predicted}$로 탐색 원점을 이동시켜 작은 다이아몬드 패턴으로 블록 정합을 하는 NNS의 탐색 경로의 예를 보인 것이다.



(a) 참조하는 인접 매크로 블록



(b) 탐색 경로의 예

(그림 3) NNS의 지원 영역과 탐색 경로

Y. Nie[9]는 ARPS(Adaptive Rood Pattern Search)에서 (그림 5)의 Type D와 같은 지원 영역을 제안하였다. 이것은 현재 블록의 바로 왼쪽 이웃에 위치하는 매크로 블록 MB_0만을 참조하여 현재 블록의 예측 움직임 벡터로 결정한다.

$$MV_{Predicted} = MV_0 \quad (4)$$

기존의 탐색 알고리즘에서는 이렇게 정의된 움직임 벡터를 결정하여 블록 정합을 위한 탐색 패턴으로 TSS 또는 DS 방법을 적절히 적용하고 있다. 이러한 예측 탐색 기법은 시공간적인 상관관계를 적용하여 속도뿐만 아니라 PSNR 및 시각적인 측면에서 좋은 결과를 보인다고 알려져 있다[9, 10]. 그러나 예측된 움직임 벡터를 탐색 원점으로 이용하여 빠르게 탐색을 수행함으로써 속도면에서는 좋은 결과를 보일 수 있으나, 움직임 벡터의 상관성이 떨어질 때는 잘못된 움직임을 추정하여 영상의 화질이 크게 저하될 수 있다.

3. 제안 탐색 알고리즘

여기서는 기존의 예측 탐색 알고리즘이 예측된 움직임 벡터로 탐색 원점을 이동시켜 탐색을 수행함으로써 블록 내에 공존하는 최소 정합 오차를 가지는 탐색점을 제외시켜 잘못된 추정을 하는 문제점을 개선하기 위해 시, 공간적인 움직임 정보에 의해 예측된 움직임 벡터들과 분할된 탐색 구간에 속하는 후보 벡터 중에서 가장 작은 정합 오차를 갖는 점을 탐색 원점으로 하여 적응적인 탐색 패턴에 따라 탐색을 수행하는 알고리즘을 제안한다.

이때 블록 정합 오차를 결정하는 평가 함수로 SAD(Sum of Absolute Difference)을 사용한다.

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

식 (5)에서 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.

3.1 움직임 벡터 예측 후보의 설정

본 논문에서 제안하는 예측 탐색 알고리즘에서는 보다 정확한 초기 탐색 위치를 얻기 위해서 <표 1>과 같은 3개의 움직임 벡터 예측 후보(Motion Vector Predictor Candidate)를 생성하여 이들의 SAD을 비교하여 가장 작은 값을 갖는 점으로 탐색 원점을 이동시켜 블록 정합을 수행한다.

〈표 1〉 움직임 벡터 예측 후보의 생성 방법

	예측 후보 1	예측 후보 2	예측 후보 3
상관관계	시간적	공간적	시/공간적
프레임 시점	t-1	t	t-1
예측 방법	현재 프레임 블록과 같은 위치에 있는 이전 프레임 블록의 움직임 벡터에 따른 구간별 후보점	현재 프레임 블록과 인접한 이전 수직 블록(left)과 수평 블록(above)의 움직임 벡터의 평균값	현재 프레임 블록과 같은 위치에 있는 이전 프레임 블록과 상, 하, 좌, 우로 인접한 움직임 벡터의 평균값

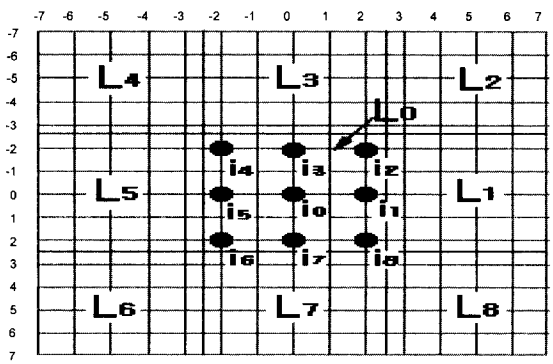
첫 번째 움직임 벡터 예측 후보(First Predictor)는 움직임이 있는 동영상의 경우 짧은 시간에 영상의 움직임이 있더라도 많은 움직임이 있을 수 없음으로 인해서 움직임 벡터는 탐색창의 중심으로 반경 2화소 이내에 분포할 확률이 약 56.72%~98.70%[6]이라는 중심 지향적 분포 특성(Center-Biased Property)을 갖게 된다는 점을 고려한다. 움직임 벡터의 설정은 식 (6)과 같이 현재 프레임 블록과 같은 위치에 있는 이전 프레임 블록의 움직임 벡터를 예측 움직임 벡터로 정한다.

$$MV_T=MV_{t-1} \quad (6)$$

이때 식 (6)에 의해 구해진 예측 움직임 벡터가 시간적 상관성이 떨어질 경우에 블록 정합 대상에서 최소 정합 오차를 갖는 탐색점이 제외될 수 있는 단점을 보완하기 위해 탐색 영역의 중심인 (0,0)에 가까운 위치에 설정한다. 따라서 정확한 움직임 후보 벡터의 설정을 위해 (그림 4)와 같이 탐색 영역을 9개의 탐색 구간으로 구분한다. 각 구간의 후보 벡터의 위치는 현재 블록과 같은 위치의 이전 프레임의 블록의 움직임 벡터가 속하는 영역으로 한다. 이때 구간별 후보점들의 위치는 해당 구간내의 중심점으로 설정할 수 있으나, 이전 움직임 벡터의 거리 오차와 움직임 벡터의 중심 지향적인 분포 특성을 고려하여 탐색 영역의 중심점 (0,0)에서 예측된 움직임 벡터가 속한 구간 L_i로 향하는 점들 중에서 L_0내의 끝점으로 함으로써 움직임 벡터를 찾는 확률을 높인다.

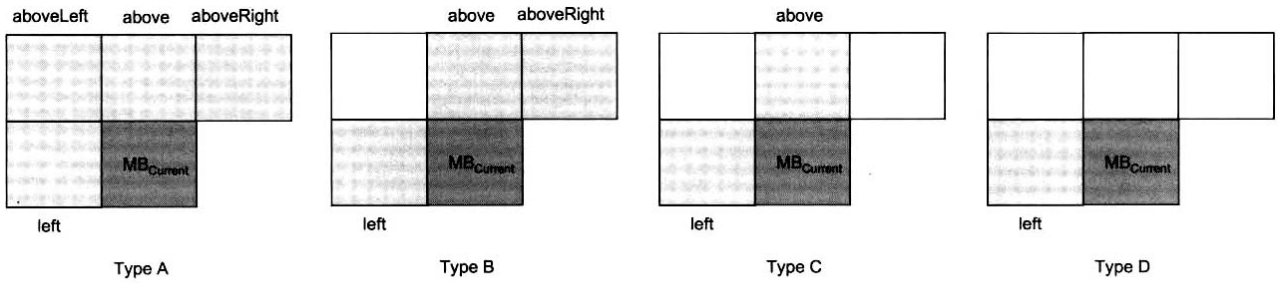
(그림 4)는 예측된 움직임 벡터가 속하는 각 구간별 후보점 위치를 나타낸다. 이때 분할된 탐색 구간의 크기를 s라 할 때, 중심점에서 각 탐색 구간과의 거리는 $d=\lfloor(s/2)$로 이 경우 $s=5$, $d=2$이므로 탐색창의 가운데 구간 L_0의 5×5에서 2화소 간격의 정사각형 형태로 분포하는 9개의 점을 탐색 구간 각각의 후보 벡터로 설정한다. 즉, 블록 정합을 위한 탐색 영역의 크기를 $(2w+1)\times(2w+1)$로 설정하고, 이전 프레임의 블록의 움직임 벡터가 MV_{t-1}이고 이 움직임 벡터가 속하는 탐색 구간이 L_i라 하면 구간별 탐색 후보점은 탐색 구간 L_i의 대표점인 I_i가 된다. 이때 각 구간의 후보점 위치를 (i,j)라 할 때, 움직임 벡터를 추정하기 위한 후보 벡터의 위치는 식 (7)과 같다.

$$MV_{pl}\leftarrow I_i(i,j)\leftarrow L_i(MV_{t-1}),\ -w\le i\le w,\ -w\le j\le w \quad (7)$$



(그림 4) 구간별 후보점의 설정을 위한 탐색창의 구간 분할과 구간별 후보점

두 번째 움직임 벡터 예측 후보(Second Predictor)는 공간적으로 인접한 매크로 블록들은 비슷한 속도로 거의 같은 방향으로 움직인다는 점을 고려한다. 공간적인 관점에

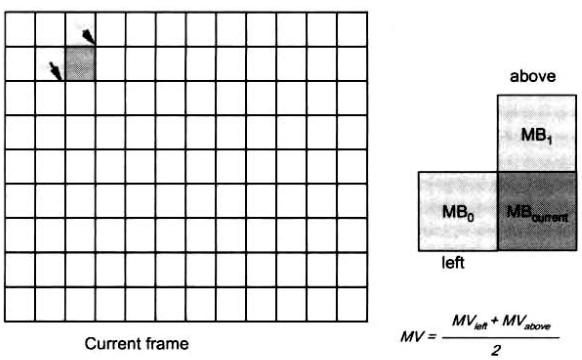


(그림 5) 지원 영역(ROS)의 4가지 유형

서, 비디오 프레임의 범위 내에서 모든 블록들은 래스터 스캔 과정에서 움직임 벡터들의 인접 블록들은 현재 블록에서 바로 이웃되는 left, above, above-left와 above-right가 참조할 대상이 된다. 이 외의 가까운 위치의 블록들은 현재 블록과 상관성이 적거나 예측 신뢰성이 떨어진다. 그 이상의 블록들의 사용은 더 많은 복잡한 계산이 필요하므로 공간적인 지원 영역(ROS : Region Of Support)은 (그림 5)와 같이 4개의 신뢰성 있는 유형으로 이웃 블록들을 정의할 수 있다.

(그림 5)에서 Type A는 4개의 이웃 블록 모두를 사용하는 것이며, Type B는 움직임 벡터들의 차등 부호화를 위한 H.263의 국제 표준으로 채용된 예측되는 지원 영역이다. Type C는 2개의 직접적으로 인접하는 블록으로 구성되며, Type D는 현재 블록의 바로 왼쪽 이웃에 위치하는 1개의 블록만을 참조 한다. 본 논문에서는 카메라의 움직임이 거의 대부분 수평, 수직 방향임을 고려하여 Type C와 같이 현재 프레임의 이전 수직 블록(left)과 수평 블록(above)을 참조한다. 즉, (그림 6)과 같이 수평, 수직에 있는 매크로 블록 MB_0, MB_1을 참조한다. 이때 현재 블록의 예측 움직임 벡터는 식 (8)과 같이 이들 인접 매크로 블록 각각의 움직임 벡터 MV_0, MV_1의 평균값으로 한다.

$$MV_{f2} = \frac{MV_{above} + MV_{left}}{2} \tag{8}$$



(그림 6) 참조하는 인접 매크로 블록

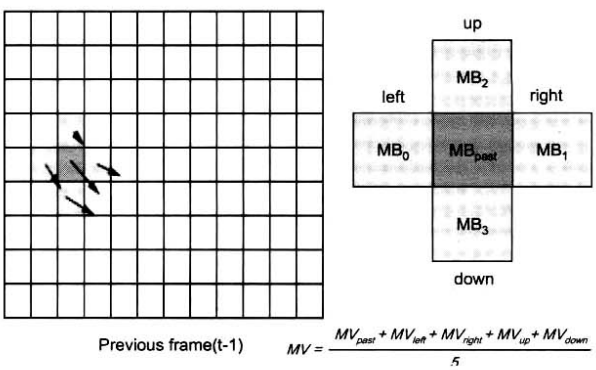
(그림 6)에서 현재 블록의 인접 매크로 블록(Adjacent Macroblock)인 MB_0, MB_1에서 예측 움직임 벡터 MV_P을 결정할 수 있다. 인접 블록 MB_i (MB_i for $i \in \{0,1\}$)의 움직임 벡터를 $V_i = [\Delta c_i, \Delta r_i]$라고 할 때, 현재 매크로 블록 MB_c의 예측 움직임 벡터는 $MV_p = [\Delta c_p, \Delta r_p]$가 된다. 예를 들어, 블록 MB_0와 MB_1의 예측 벡터인 Δc_0과 Δc_1 값이 Δc_p와 유사하다고 가정한다면 식 (9)와 같다.

$$\Delta c_p = \frac{(\Delta c_0 + \Delta c_1)}{2} \tag{9}$$

그리고 Δr_p도 식 (9)와 같은 방법으로 구한다. 따라서 현재 블록의 좌표가 (m, n)이면 탐색의 초기 중앙점의 좌표는 $(m + \Delta c_p, n + \Delta r_p)$이다.

세 번째 움직임 벡터 예측 후보(Third Predictor)는 영상의 움직임은 급격히 변화하기 보다는 완만하게 변화한다는 연속성 제약 사항(Motion Continuity Constraint)[2]에 근거하여 이전 프레임의 같은 위치에 있는 블록의 움직임 벡터를 참조하여 t-1 시점에서 추출한 움직임 벡터를 이용한다. 즉, (그림 7)과 같이 이전 프레임(t-1)의 같은 위치에 있는 블록 MB_{past}와 좌, 우, 상, 하의 인접 매크로 블록 MB_0, MB_1, MB_2, MB_3들을 포함한 5개의 움직임 벡터의 평균으로 구한다.

$$MV_{f3} = \frac{MV_{past} + MV_{left} + MV_{right} + MV_{up} + MV_{down}}{5} \tag{10}$$



(그림 7) 이전 프레임의 인접 매크로 블록

그리고 생성된 3개의 후보 벡터의 SAD을 비교하여 가장

작은 값을 가지는 벡터를 최종적으로 탐색 원점으로 결정한다.

$$MV_{predicted} = minSAD(SADMV_{p1}, SADMV_{p2}, SADMV_{p3}) \quad (11)$$

3.2 탐색을 위한 적응적인 탐색 패턴

기존의 탐색 알고리즘은 움직임의 크기와 관계없이 고정된 탐색 패턴을 사용함으로써 고정된 수의 탐색점들을 검색하여야 하는 단점이 있다. 탐색 패턴의 설정은 움직임 방향 자체가 아니라 움직임 벡터를 찾기 위해 매 단계의 탐색을 수행할 때 어떤 탐색 패턴을 사용하는 것이 최소 개수의 효과적인 탐색 후보점을 검사하여 보다 정확도가 높은 움직임 벡터를 찾는데 있다. 따라서 본 논문에서는 식 (11)에 의해 구해진 최소 정합 오차를 갖는 움직임 벡터의 크기에 따라 중간-정지(Half-Stop) 탐색 패턴으로 적응적으로 조절한다.

이때 패턴을 결정하는 기본적인 개념은 예측된 움직임 벡터의 수직 움직임 벡터 요소와 수평 움직임 벡터 요소를 참조하여 식 (13)의 임계값 Γ로서 정한다.

$$\Gamma = |MV_{predicted}| = \left|\sqrt{MV^2_{predicted}(x) + MV^2_{predicted}(y)}\right| \quad (12)$$

여기서 $MV_{predicted}(x)$와 $MV_{predicted}(y)$는 각각의 예측된 움직임 벡터의 수평, 수직 요소들이다. 매개 변수 Γ는 제곱과 제곱근을 포함한다. 이때 예측된 움직임 벡터의 두개의 요소에서 절대 값이 큰 값을 가지는 것을 패턴 적용 범위로 결정한다. 본 논문에서는 움직임이 통계적으로 탐색 영역의 중심으로부터 반경 2화소 내에서 찾아질 확률이 크므로[6] 적응적인 패턴을 고려하는 경계 임계값을 $\Gamma=2$로 하였다.

$$\Gamma = Max\{|MV_{predicted}(x)|, |MV_{predicted}(y)|\} \quad (13)$$

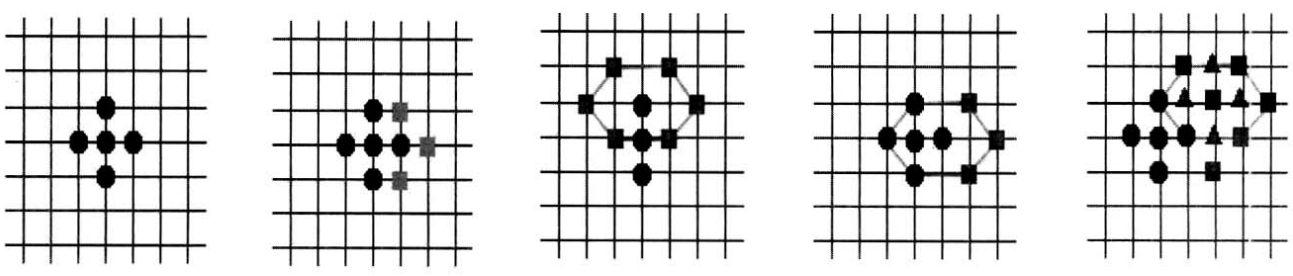
식 (13)에서 Γ가 작으면 움직임 벡터의 크기가 작으며 영상의 움직임이 작고 탐색창의 중심에 가까운 거리에 존재한다. 이럴 경우 후보점을 중심으로 예측 후보들의 크기에 따라 예측의 정밀도를 향상시키며, 예측 후보들이 선택된 예측 후보 내에서 고신뢰적이라면 작은 탐색점 수를 가진 탐색 패턴을 적용하여 알고리즘의 전체적인 속도를 개선한다. 그리고 Γ가 크면 움직임이 크며 중심점으로부터 일정한 거리만큼 떨어져 있는 위치에 움직임 벡터가 존재한다고 판단되므로 국부 영역 탐색에 효율적인 방법을 이용하여 빠르게 움직임 벡터를 탐색한다.

따라서 $\Gamma=0$인 경우는 예측 움직임 벡터가 MV=(0,0)으로 탐색 영역의 중심에 존재할 확률이 매우 크므로 정합 기준 값을 검사하는 탐색수가 가장 작은 루드 패턴(Rood Pattern)을 사용하며, $0<\Gamma\leq 2$일 때는 넓은 영역에 걸쳐 움직임 벡터가 분포되어 있으므로 수평 방향으로 빠르게 탐색해 나갈 수 있는 수정된 육각 패턴(Improve Hexagon Pattern)을 사용하고, $\Gamma>2$는 분포 확률이 매우 적으므로 역시 탐색점 수가 작은 루드 패턴을 사용한다.

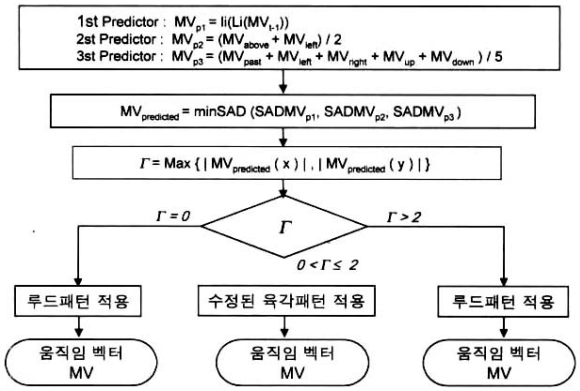
3.3 제안된 탐색 알고리즘에 의한 움직임 추정

현재 블록과 같은 위치에 있는 이전 시점의 프레임 블록의 움직임 벡터가 속하는 탐색 구간에 따른 해당 후보점(1st Predictor) MV_{p1}과 현재 프레임 블록과 같은 위치에 있는 이전 시점의 프레임 블록과 주변 매크로 블록들의 평균으로 예측되는 후보 움직임 벡터(2nd Predictor) MV_{p2}와 이전 프레임의 같은 위치에 있는 블록과 상, 하, 좌, 우의 인접 매크로 블록들을 포함한 5개의 움직임 벡터의 평균으로 예측되는 후보 움직임 벡터(3rd Predictor) MV_{p3}을 구하여 이 3개의 후보점 중에서 최소 SAD인 최적합점을 찾아 최소 SAD인 최적합점의 움직임 벡터의 크기 Γ를 구한다. 최적합점으로 탐색 원점을 이동하여 이동된 탐색 원점을 중심으로 $\Gamma=0$ 또는 $\Gamma>2$이면 루드 패턴을 적용하고, $0<\Gamma\leq 2$일 때는 수정된 육각 패턴을 적용한다.

(그림 9)는 제안한 알고리즘의 순서를 간단히 나타낸 흐름도로 제안한 알고리즘에서는 각 움직임 예측 벡터의 SAD를 비교하여 가장 작은 값을 갖는 점을 구한 후, 이 점을 중심으로 임계값 Γ에 따라 적응적인 탐색 패턴에 의해 움직임 벡터를 추정하는 과정을 나타낸다.



(그림 8) 탐색 패턴의 정의 (a)~(b) 루드 패턴 (c)~(e) 납작한 육각 패턴



(그림 9) 제안한 움직임 벡터 추정 방식의 흐름도

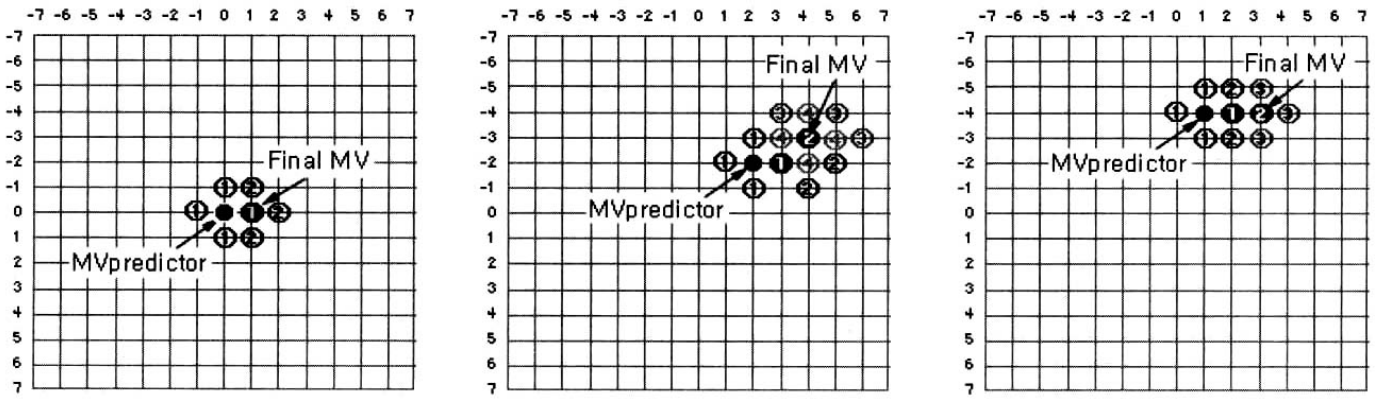
각 패턴의 탐색 과정은 탐색 원점을 중심으로 (그림 8)(a)와 같이 5개의 탐색점에 대하여 최소 정합 오차를 계산한다. 이때 중심점이 최소 정합 오차면 탐색을 중단하고 이 점을 최종적으로 움직임 벡터로 결정한다. 만일 중심점이 최소 정합 오차가 아니라면 루드 패턴인 경우는 최소 정합 오차를 가지는 점을 중심점으로 (그림 8)(b)와 같이 탐색을 반복하고, 수정된 육각 패턴인 경우는 (그림 8)(c), (그림 8)(d)와 같이 최소 정합 오차인 점을 중심으로 큰 납작한 육각 패턴을 구성하여 최소 정합 오차를 계산한다. 새로 계산된 최소 정합 오차 점이 중심점에 위치할 때까지 탐색 과정을 반복하여 최소 정합 오차를 구하며, 최종적으로 최소 정합 오차 점이 중심점일 경우는 (그림 8)(e)와 같이 반경 1화소에 해당하는 이웃 4점을 포함하는 루드 패턴을 구성하여 최소 정합 오차를 계산하여 이 단계에서 구해진 최소 정합 오차 점이 움직임 벡터로 결정된다. 단, 탐색 과정에서 탐색 영역(w = ± 7)을 벗어나는 모든 점들은 무시한다. 그리고 매 단계마다 최소 정합 오차는 재정의된다.

(그림 10)은 제안 알고리즘의 탐색 수행의 예로서 3개의 후보 벡터 중에서 최소 SAD인 최적합점으로 탐색 원점을 이동하여 $\Gamma = 0$ 또는 $\Gamma > 2$이면 루드 패턴을 적용하고, $0 < \Gamma \leq 2$일 때는 수정된 육각 패턴을 적용하여 블록 정합을 수행하는 과정을 각각 보인 것이다. 제안된 방식은 탐색 영역이 벗어나는 모든 점을 무시하고 중간에 탐색을 멈추는 방법을 제시하였는데 중간에 멈출 수 있는 조건은 현재 탐색 단계에서 최소 정합 오차를 갖는 점의 위치가 이전 탐색 단계의 최소 정합 오차를 갖는 점의 위치와 같다면 중간에 탐색을 멈추고 최종적인 움직임 벡터로 결정한다.

4. 실험 결과

제안된 기법의 성능을 평가하기 위하여 9개 실험 영상에 대해 각각 80프레임씩을 대상으로 실험하였다. (그림 11)은 각 실험 영상의 첫 번째 프레임이다. 비교 탐색 기법으로는 FS와 DS, 예측 탐색 알고리즘인 TPS, NNS, ARPS, 그리고 제안한 탐색 기법을 사용하였다. 그리고 움직임 추정에 사용된 매크로 블록의 크기는 16×16 화소며, 탐색 영역의 변위는 ±7을 적용하여 Pentium Ⅳ 1.6GHz와 256MB 메모리가 장착된 컴퓨터상에서 실험을 수행하였다.

(그림 11) 각 실험 영상의 첫 번째 프레임



(a) $\Gamma = 0$일때 루드 패턴 적용 (b) $0 < \Gamma \leq 2$일때 육각 패턴 적용 (c) $\Gamma > 2$일때 루드 패턴 적용

(그림 10) 제안 알고리즘의 탐색 패턴의 적용과 탐색 경로의 예

블록 정합의 정도를 평가하기 위해 대표적인 정합 기준인 평가 함수(Cost Function)로 영상 화질의 품질을 평가하기 위한 식 (14)와 같은 평균 제곱 오차(MSE : Mean Squared Error), 식 (15)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.

$$MSE(i,j)=\left(\frac{1}{N^2}\right)\sum_{k=1}^{N}\sum_{l=1}^{N}[I_t(k,l)-I_{t-1}(k+i,l+j)]^2 \quad (14)$$

$$MAD(i,j)=\left(\frac{1}{N^2}\right)\sum_{k=1}^{N}\sum_{l=1}^{N}|I_t(k,l)-I_{t-1}(k+i,l+j)| \quad 15)$$

그리고 화질의 평가를 위한 PSNR은 식 (16)과 같다.

$$PSNR=10\ log_{10}\frac{255^2}{MSE} \quad (16)$$

실험 영상에 대한 실험 결과는 <표 2>와 <표 3>에 각각 나타내었다. <표 2>는 기존의 탐색 기법과 제안한 알고리즘의 각 블록에 대한 평균 PSNR 값과 FS의 탐색 회수를 1로 기준하여 각 기법의 상대적인 속도 향상에 대하여 비교한 결과이며, <표 3>은 각 블록에 대한 MAD의 비교

〈표 2〉 각 실험 영상에 대한 성능 비교 평가 함수의 결과 비교 값

Sequence	Format		Motion Estimation Algorithm					
			FS	DS	TPS	NNS	ARPS	Proposed
News	CIF	PSNR	38.23	38.15	38.00	38.11	38.10	38.14
		Speed Up	1	16	13	39	38	39
Miss America	CIF	PSNR	39.10	38.76	38.79	38.99	38.66	39.01
		Speed Up	1	12	14	28	29	28
Foreman	CIF	PSNR	33.46	33.15	32.98	32.67	32.41	33.16
		Speed Up	1	13	15	26	25	27
Susie	SIF	PSNR	34.72	34.42	34.40	34.02	33.82	34.45
		Speed Up	1	13	15	26	27	27
Flower Garden	SIF	PSNR	23.99	23.81	23.49	23.73	23.18	23.89
		Speed Up	1	12	16	25	23	30
Mobile	SIF	PSNR	22.94	22.89	22.92	22.98	22.90	22.93
		Speed Up	1	16	16	29	32	36
Akiyo	QCIF	PSNR	43.85	43.85	43.85	43.85	43.85	43.85
		Speed Up	1	16	16	40	38	40
Coastguard	QCIF	PSNR	31.01	30.93	30.90	30.92	30.86	30.99
		Speed Up	1	14	16	28	27	32
Stefan	QCIF	PSNR	25.71	25.62	25.23	25.40	25.25	25.62
		Speed Up	1	13	16	27	28	29
평 균		PSNR	32.56	32.40	32.28	32.30	32.11	32.45
		Speed Up	1	13.9	15.2	29.8	29.7	32.0

〈표 3〉 각 실험 영상에 대한 제안된 기법과 기존 기법의 각 블록에 대한 평균 절대값 오차(MAD) 비교

Sequence	Format	Motion Estimation Algorithm					
		FS	DS	TPS	NNS	ARPS	Proposed
News	CIF	1.042	1.047	1.062	1.047	1.051	1.046
Miss America	CIF	1.930	2.018	2.011	1.966	2.024	1.965
Foreman	CIF	2.817	2.904	2.956	2.958	2.997	2.882
Susie	SIF	2.665	2.744	2.741	2.787	2.811	2.728
Flower Garden	SIF	8.520	8.768	8.950	8.704	8.919	8.608
Mobile	SIF	9.546	9.604	9.572	9.507	9.573	9.554
Akiyo	QCIF	0.488	0.488	0.488	0.488	0.488	0.488
Coastguard	QCIF	4.180	4.266	4.220	4.242	4.276	4.191
Stefan	QCIF	6.794	6.896	6.891	6.869	7.061	6.852

결과를 나타낸다. <표 2>와 <표 3>의 CIF(Common Intermediate Format)는 352×288, SIF(Source Input Format)는 352×240, 그리고 QCIF(Quarter-CIF)는 176×144로 프레임의 크기를 나타낸다.

<표 2>에서 제안한 방식이 FS에 비해 평균 탐색점 수가 약 95% 정도 감소하였고, 탐색 속도면에서는 약 27~40배 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 9개의 실험 영상에 대한 평균이 제안한 방식이 32.45[dB]로 나타나 33.56[dB]인 FS 다음으로 우수하였다.

그리고 실험에 사용한 영상별로 볼 때, 입술과 머리 등을 주기적으로 움직이는 것으로 움직임이 거의 없는 Akiyo, News와 Miss America 영상은 FS와 비교하였을 때 탐색점 수를 95% 감소시키면서도 PSNR 값이 Akiyo는 0, 나머지는 0.09[dB] 차이로 거의 화질의 저하가 없었으며, FS를 제외한 다른 기법에 비해 PSNR 값이 가장 우수하였다. 얼굴을 흔들고 크게 말을 함으로써 움직임이 조금 있는 Foreman과 Susie 영상의 PSNR 값에서는 제안된 기법이 FS보다 약 0.3[dB] 정도 떨어지나 탐색점 수에서 약 27배의 많은 탐색점 수를 사용하여 움직임을 추정한 것에 비해 거의 근접한 값을 얻을 수 있었다. 빠르게 수평 방향으로 움직이는 보트를 카메라가 따라가는 Coastguard 영상에서는 FS가 제안된 기법보다 0.3[dB] 정도 우수하나 탐색점 수를 27배 이상 사용되었고, 움직임이 큰 Stefan과 Flower Garden 영상에서는 FS가 각각 0.09[dB], 0.1[dB]정도가 좋으나 약 30배 정도의 많은 탐색점 수를 사용하였다.

이상의 실험 결과에 의하면, PSNR 측면에서 움직임이 크거나 작을 때는 FS에 비해 0.01~0.1[dB] 정도 떨어지고 움직임이 조금 있는 영상은 FS에 비해 0.3[dB] 정도가 떨어졌으나, FS를 제외한 모든 기법에 비해 제안된 방식이 가장 우수한 것으로 나타났다. FS에 비해 0.3[dB] 떨어지는 비교적 움직임이 큰 영상에서는 제안된 방식이 다른 영상에 비해 더 적은 수의 탐색점 검색만으로 정합 블록을 찾으므로 속도면에서 가장 우수하게 나타났다. 즉, 프레임별 블록당 평균 탐색수(Speed Up)에 있어서 움직임이 적은 영상인 경우 루드 패턴을 이용하는 NNS와 제안된 기법이 높은 탐색 속도 향상을 나타내었으며, 비교적 움직임이 있는 영상에서는 제안된 기법이 평균 탐색점 수가 6.9~16.6% 정도 감소되어 탐색 속도가 향상되었음을 볼 수 있었다. 이것은 루드 패턴이 탐색 영역의 중심에 분포하는 움직임 벡터를 빨리 찾을 수 있음을 보여 주었으며, 수정된 육각 패턴이 탐색 영역에 넓게 분포하는 수평축 방향으로 존재하는 움직임 벡터의 탐색에 효율적임을 알 수 있었다.

전체적으로 제안된 기법이 영상의 화질을 평가하기 위한 비교 평가 함수로 사용된 MAD에서는 비교된 예측 탐색 알고리즘에 비해 0.13~0.39 정도 우수하고, 평균 PSNR 값에서는 0.15~0.34[dB] 개선되어 화질면에서도 FS에 근접하는 안정된 예측 정확도를 얻을 수 있었다. 그리고 MPEG 표준인 다이아몬드 패턴을 사용한 고속 블록 정합 알고리즘인 DS에 비해 제안된 방식이 PSNR 값에서는 0.05[dB], 속도면에서는 2.3배 향상 되었다.

5. 결 론

본 논문에서는 기존의 예측 탐색 알고리즘이 예측된 움직임 벡터로 탐색 원점을 이동시켜 탐색을 수행함으로써 블록 내에 공존하는 최소 정합 오차를 가지는 탐색점이 제외되어 화질 저하를 초래하는 문제점을 개선하기 위해 이전 움직임 정보에 의해 예측된 움직임 벡터와 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 예측 탐색 기법과 거의 같은 탐색점 수를 사용하면서도 화질면에서는 나은 결과를 보였다. 이 경우 FS를 제외한 다른 기법과 비교하였을 때 움직임 예측면에서는 평균적으로 0.05~0.34[dB] 정도의 성능 향상을 보였다.

움직임 추정시 본 논문에서 제안한 예측 탐색 기법을 사용하여 움직임 추정을 한다면 보다 빠르게 움직임 벡터를 찾을 수 있을 것이며, 고속 블록 정합 방법에 적합한 탐색 패턴을 도출하고, 탐색 패턴을 적응적으로 변화시키면 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.

참 고 문 헌

[1] 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.

[2] Vincent S., S. Hwang, "Tracking Feature Points in Time-Varying Image using an Opportunistic Selection Approach," Pattern Recognition, Vol.22, pp.247-256, 1989.

[3] L. M. Po, W. C. Ma, "A Novel Four-Step Search Al-

gorithm for Fast Block-Matching Motion Estimation," IEEE Transactions on Circuits & System for Video Tech., Vol.6, No.3, pp.313-317, June, 1996.

[4] L. K. Kuo, E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," IEEE Transactions on Circuits & System for Video Tech., Vol.6, No.4, pp.419-422, Aug., 1996.

[5] M. Gallant, G. Cote, F. Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding", IEEE Transactions on Image Processing, Vol.8, No.12, pp.1816-1823, Dec., 1999.

[6] S. Zhu, K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Transactions on Image Processing, Vol.9, No.2, pp.287-290, Feb., 2000.

[7] C. Zhu, X. Lin, L. P. Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation," IEEE Transactions on Circuits & System for Video Tech., Vol.12, No.5, pp.349-355, May, 2002.

[8] 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.

[9] Y. Nie, K. Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation," IEEE Transactions on Image Processing, Vol.11, No.12, pp.1442-1449, Dec., 2002.

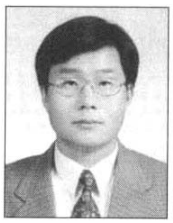
[10] F. Moschetti, M. Kunt, E. Debes, "A Statistical Block-Matching Motion Estimation," IEEE Transactions on Circuits & System for Video Tech., Vol.13, No.4, pp.417-431, Apr., 2003.

[11] H. W. Nam, I. Y. Park, Y. C. Wee, H. J. Kimn, "A New Cross and Hexagonal Search Algorithm for Fast Block Matching Motion Estimation," KIPS, Vol.10-B, No.7, pp.811-814, Dec., 2003.

곽 성 근

e-mail : skkwak@icc.ac.kr
1973년 건국대학교 전자공학과(학사)
1980년 연세대학교 대학원 전자공학과 (공학석사)
1998년 아주대학교 대학원 컴퓨터공학과 (박사과정수료)
1980년~현재 시립인천전문대학 컴퓨터정보과 교수
관심분야 : 컴퓨터그래픽스, 이미지프로세싱 및 애니메이션

위 영 철



e-mail : ycwee@ajou.ac.kr
1980년 연세대학교 수학과(학사)
1982년 SUNY at Albany Computer Science Department(학사)
1984년 SUNY at Albany Computer Science Department(공학석사)
1989년 SUNY at Albany Computer Science Department (공학박사)
1990년~1995년 삼성종합기술원 수석연구원
1995년~1998년 현대전자 부장
1998년~현재 아주대학교 정보통신대학 조교수
관심분야 : 컴퓨터그래픽스, 이미지프로세싱 및 알고리즘

김 하 진

e-mail : hjkimn@ajou.ac.kr
1962년 서울대학교 수학과(학사)
1978년 프랑스 Grenoble 1대학교 대학원 응용수학과(석사)
1980년 프랑스 Saint-Etienne 대학교 대학원 응용수학과(박사)
1991년~1992년 한국정보과학회 회장
1992년~현재 국제정보올림피아드 추진위원장
1992년~현재 JTC1/IEC SC24(그래픽스 표준화) 국내위원회 위원장
1974년~현재 아주대학교 정보통신대학 교수
관심분야 : 컴퓨터그래픽스, 이미지프로세싱 및 응용수학