움직임 해석을 통한 고속 움직임 예측 알고리즘

*전영현, 윤종호, 조화현, 최명렬
한양대학교 전자전기제어공학
e-mail : *impword, sfw1179, chh, choimy@asic.hanyang.ac.kr*

A Fast Motion Estimation Algorithm with Motion Analysis

*Young-Hyun Jun, Jong-Ho Yun, Hwa-Hyun Cho, Myung-Ryul Choi
Dept.of EEIC Hanyang University

Abstract

We present an efficient block-based motion estimation algorithm with motion analysis. The motion analysis determines a size of search pattern and a maximum repeated count of search pattern. In case of large movement in large image, we reduce search points and the local minimum which caused by low performance. The proposed algorithm employs with searching step of 2. The first step determines an initial search point with neighbor block vector and a size of initial search pattern. The second step determines a size of search pattern and a maximum repeated count with motion analysis.

We improve motion prediction accuracy while reducing required computational complexity compared to other fast block-based motion estimation algorithms.

I. 서론

영상 부호화는 제한된 전송 용량에서 실시간으로 디지털 비디오를 저장하고 전송하기 위한 가장 핵심적인 기술이다. 최근 영상 부호화 표준에서는 블록 정합 움직임 예측 기법을 사용하여 부호화를 한다. 영상 부호화 표준은 공간영역에서의 중복 요소를 제거하기 위한 DCT 변환과 시간영역에서의 중복 요소를 제거하기 위한 블록 정합 움직임 예측 기법을 사용한다. 움직임 예측기법은 시간영역에서 중복 요소를 제거하기 위해 움직임 벡터를 예측하고, 부호화기에서 가장 많은 계산량을 요구하는 부분이다.

일반적으로 고속 움직임 예측 알고리즘으로는 NTSS (New Three Step Search)[1], 4SS(Four Step Search)[2], 1DFS(One Demensional Full Search)[3], DS(Diamond Search)[4], TSS(Three Step Search)[5] 등이 있다. 고속 움직임 예측 알고리즘을 사용하면 FS(Full Search)을 사용하였을 때 보다는 탐색점 개수가 감소하여 움직임 예측 정확도는 감소하지만 탐색 속도를 향상 시킬 수 있다.

본 논문에서는 인접 블록의 움직임 벡터를 이용한 탐색 패턴과 가변적인 패턴 크기로 불필요한 탐색점 개수와 성능저하를 줄이는 움직임 예측 알고리즘을 제안한다. 2장에서는 고속 움직임 예측에 대표적인 알고리즘인 NTSS와 4SS에 대하여 고찰하고, 3장에서는 제안된 알고리즘에 대하여 설명한다. 4장에서는 모의실험을 통한 알고리즘 성능을 비교하며, 5장에서는 결론을 맺는다.

II. 관련 연구

2.1 Full Search (FS)

블록 정합 알고리즘에서 FS는 탐색 영역내에 있는 모든 탐색점에 대하여 기준 블록과 비교한다. 모든 탐색 영역에서 기준 블록과 비교로 성능은 제일 좋으나, 화면과 탐색영역의 크기가 증가함에 따라 탐색점 개수가 증가하여 연산량이 많아지는 단점이 있다.

2.2 New Three Step Search (NTSS)

움직임이 작은 영상에서 첫 번째 단계에서의 균일한 탐색 크기를 갖는 TSS[5]는 적절하지 않다. 작은 움직임을 효율적으로 찾기 위해 NTSS 알고리즘에서는 첫 번째 단계에서 탐색영역 중심에 8개의 추가 탐색점을 가진다. 그림 1은 탐색영역이 최대 ±7에 대한 NTSS의 탐색 경로의 예를 나타내며, 각 과정은 아래와 같다.

첫 번째 단계 : 첫 번째 탐색 크기로 d/2인 9개의 탐색점과 탐색영역 중심으로 최소 탐색 크기(d=1)인 8개의 탐색점에 대하여 MBD(Minimum Block Distortion) 를 구한다. 탐색점 16개중 최소 MBD인 탐색점을 다음 단계의 중심점으로 한다. 단. d는 최대로 가질 수 있는 탐색 영역이다.

두 번째 단계 : 탐색 크기를 반으로 줄인 후 8개의 탐색점 중 MBD가 최소인 탐색점을 구한다.

세 번째 단계 : 탐색 크기를 반으로 줄인 후 8개의 탐색점 중 MBD가 최소인 탐색점을 움직임 벡터로 한다.

움직임이 작은 영상에 대해서는 세 단계를 모두 처리하지 않는 두가지 경우가 있다.

1) 첫 번째 단계에서 최소 MBD가 중심점이라면 탐색을 중지하고 움직임 벡터를 구한다.
2) 첫 번째 단계에서 최소 MBD가 중심점에서 인접한 8개의 탐색점이라면, 두 번째 단계에서 최소(d=1)인 탐색점에서만 탐색을 하여 최소 MBD인 탐색점에서 움직임 벡터를 구한다.

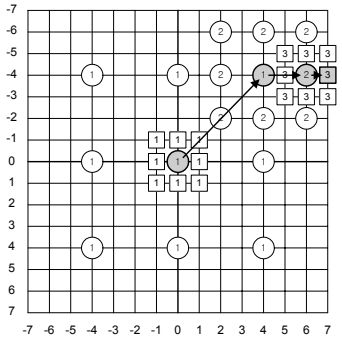

그림 1. NTSS 탐색 경로 예

2.3 Four Step Search (4SS)

4SS은 움직임 벡터가 탐색영역의 중심 부분으로 분포되어 있는 특성을 이용하여 제안되었다. 4SS은 TSS보다 좋은 성능을 보이며 NTSS와 비슷한 성능을 보인다.

그림 2는 탐색영역이 최대 ±7에 대한 4SS의 탐색 경로의 예를 나타내며, 각 과정은 아래와 같다.

첫 번째 단계 : 탐색 크기를 3로 하는 9개의 탐색점에서 최소 MBD인 탐색점을 다음 단계의 중심점으로 한다. 최소 MBD인 탐색점이 중심점이면, 네 번째 단계로 이동하고 아니면 두 번째 단계로 이동한다.

두 번째 단계 : 이전 단계의 탐색 방법과 같으며 중복되는 탐색점에 대해서는 탐색하지 않는다. 최소 MBD인 탐색점이 중심점이면, 네 번째 단계로 이동하고 아니면 세 번째 단계로 이동한다..

세 번째 단계 : 이전 단계의 탐색 방법과 같으며, 네 번째 단계로 이동한다.

네 번째 단계 : 탐색 크기를 1로 하는 9개의 탐색점에서 최소 MBD인 탐색점을 구하여 움직임 벡터를 구한다.

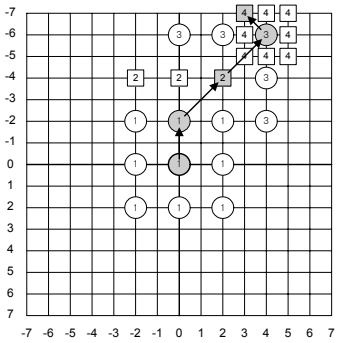

그림 2. 4SS 탐색 경로 예

III. 제안된 알고리즘

블록 정합 방법으로 효율적인 움직임 예측을 하기 위해 필요한 요소로는 탐색 패턴과 탐색 크기가 있다. 움직임의 분포가 넓게 형성되어 있는 영상에 대해서 고정된 탐색 크기를 가지고 있는 것은 적합하지 않다. 제안된 알고리즘은 인접 블록의 움직임 차의 누적분포도를 이용하여 탐색 크기와 최대 반복회수를 구한다. 각 단계별 MBD 산출 방법은 곱셈기를 사용하지 않고, 간단한 하드웨어 구현을 할 수 있는 차의 절대값 합(SAD: Sum of Absolute Difference)을 사용하여 구한다.

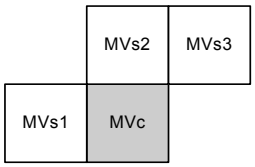

그림 3. 참조할 인접 매크로 블록

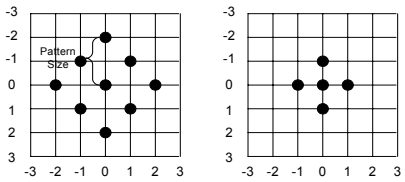

그림 4. 탐색 패턴

초기 탐색 중심점은 인접한 매크로 블록의 움직임 벡터값을 사용한다. 그림 3은 현재 매크로 블록과 인접한 매크로 블록을 나타낸다.

제안된 알고리즘은 세 단계이고, 탐색 패턴은 두가지로 이루어진다. 그림 4는 제안된 알고리즘의 탐색 패턴이다. 첫 번째 단계에서는 그림 3을 사용하여 세 개의 인접한 매크로 블록의 움직임 벡터값(MVs1, MVs2, MVs3), 중간값(MVm)과 중간값의 반대값(MVcm)에서 최소 SAD를 구하여 초기 탐색 중심점으로 사용한다. 첫 번째 단계에서 패턴의 크기는 인접한 블록의 움직임 벡터중 가장 큰 값으로 한다. 탐색점 중 최소 SAD 탐색점을 두번째 단계의 중심점으로 사용한다.

그림 5은 첫 번째 단계에서 구한 중심점에서 FS로 찾은 움직임 벡터 차의 누적 분포도를 나타낸다. x축은 움직임 벡터 차를 나타내며, y축은 누적값을 나타낸다. 움직임 벡터 차의 누적값이 80%까지 차이값을 d1라 하고 95%까지 차이값을 d2라 한다.

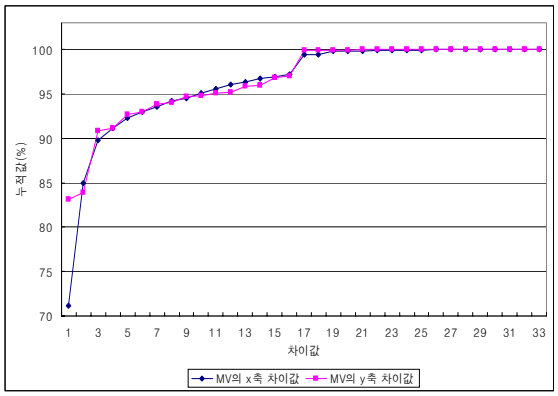

그림 5. 인접 블록간의 움직임 벡터 차의 누적 분포도

두 번째 단계에서는 패턴의 크기를 d1로 하고, 탐색 패턴의 최대 탐색 반복 횟수(CMax)는 d2/d1로 구하여 SAD가 중심 위치에서 최소일때 까지 반복한다.

마지막 단계로 패턴 크기가 1인 탐색 패턴을 적용하여 최종 움직임 벡터를 구한다.

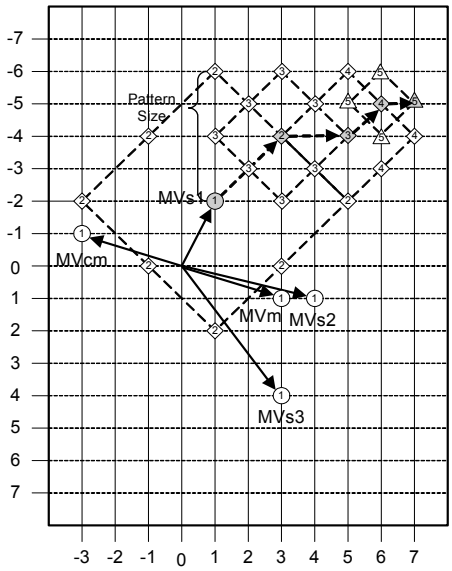

그림 6. 제안된 알고리즘의 예측 경로 예

그림 6은 제안된 블록 정합 알고리즘의 예측 경로의 예를 나타낸다.

Ⅳ. 실험 결과

본 논문에서 사용된 영상은 VGA(640x480 화소, 8 비트/화소, 30 프레임/초)인 “Flower Garden”, "Susie", ”Flowers", "Tennis" 이며 30프레임씩 수행하였다. 움직임 벡터의 최대 범위는 (-16,-16)~(16,16)으로 하였다. 움직임 예측 성능을 검증하기 위하여 PSNR(Peak Signal-to-Nosie Ratio)을 사용하였다.

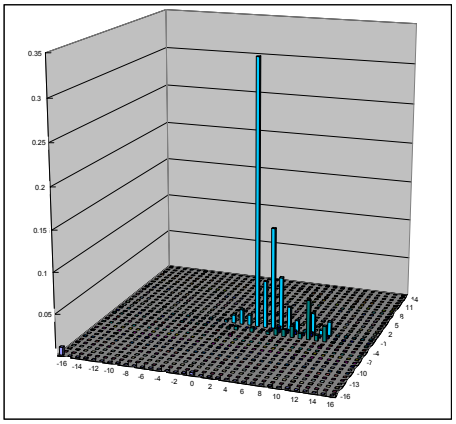

그림 7. “Flower Garden” 영상의 움직임 벡터 확률 분포도

그림 7은 FS로 구한 "Flower Garden" 영상의 움직임 벡터 확률 분포도를 나타낸다.

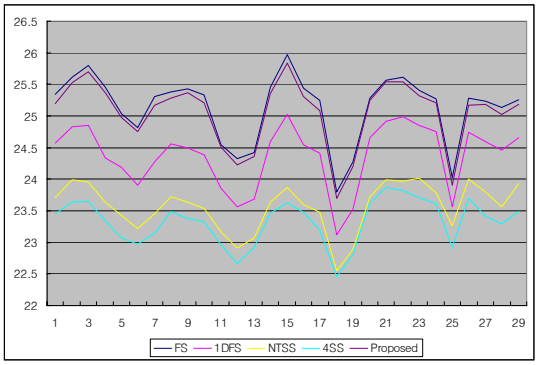

그림 8. "Flower Garden" 영상에서의 PSNR(dB) 그래프

그림 8은 "Flower Garden" 영상 30 프레임으로 제안된 알고리즘을 NTSS(New Three Step Search), 4SS(Four Step Search), 1DFS(One Demensional Full Search) , FS(Full Search) 와 PSNR을 비교하였다. 중복되는 탐색점 개수는 평균 탐색점 개수에서 제외하였다.

표 1. 영상별 평균 PSNR(dB)

블록 정합 알고리즘	영상별 평균 PSNR(dB)			
	Flower Garden	Susie	Flowers	Tennis
FS	25.13	33.90	29.14	26.97
1DFS	24.37	30.37	27.17	25.17
NTSS	23.57	29.03	25.69	25.76
4SS	23.33	28.73	24.12	25.50
Proposed	25.05	33.33	28.92	26.72

표 2. 영상별 평균 탐색점 개수

블록 정합 알고리즘	영상별 평균 탐색점 개수			
	Flower Garden	Susie	Flowers	Tennis
1DFS	127.27	130.78	129.85	125.86
NTSS	25.03	26.20	23.80	26.00
4SS	15.44	18.40	14.00	16.40
Proposed	20.28	19.26	19.69	20.18

제안된 알고리즘은 4SS에 비해 탐색점 개수는 5개 정도 증가 하였으나, 그 외 알고리즘에서는 감소하였고, 비교한 모든 알고리즘에 비해 성능은 평균 PSNR은 1.73~3.08 dB 향상되었다.

V. 결론

본 논문에서는 영상의 움직임 해석을 통해 패턴의 크기를 가변적으로 적용하였고, 패턴의 반복 횟수를 최대 탐색 반복 횟수로 제한하였다. 초기 탐색 중심점은 인접한 매크로 블록의 움직임 벡터를 이용하였다.

제안된 알고리즘은 움직임이 큰 영상에 불필요한 탐색점 개수와 지역적 최소값에 빠질 위험을 감소시켰다. 비교한 다른 알고리즘에 비해 평균 탐색점 개수와 평균 PSNR이 향상되었음을 볼 수 있었다.

참고문헌

[1] Renxiang Li, Bing Zeng, and Ming L. Linu, "A New Tree-Step Search Algorithm for Block Motion Estimatio", IEEE Trans. Circuit and Syst. for Video Tech. vol.4, no.4, pp.438-442, Aug.1994

[2] Lai-Man Po and Wing-Chung Ma, "A Novel Four-Step Algorithm for Fast Block motion Estimation", IEEE Trans. Circuit and Syst. for Video Tech. vol.6, no.3, pp.313-317, June.1996

[3] Mei-Juan Chen, Liang-Gee Chen and Tzi-Dar Chiueh, "One-Dimensional Full Search Motion Estimation Algorithm For Video Coding", IEEE Trans. Circuit and Syst. for Video Tech.,vol.4, no.5, pp.504-509, Oct.1994

[4] Shan Zhu and Kai-Kuang Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Trans. on image Processing, vol.9, no.2, pp.287-290, Feb.2000

[5] T.Koga, K.Iinuma, A.Hirano, Y.Lijima, and T.Ishiguro, "Motion compensated interface coding for video conferencing" in Proc. NTC 81, pp.C9.6.1 -9.6.5, New Orleans, LA, Nov./Dec.1981.

[6] Iain E.G. Richardson, H.264 and MPEG-4, Wiley, 2003

[7] S.Lee and S.-I.Chae, "Motion estimation algorithm using low resolution quantisation", Electronics Tetters, vol.32, no.7, pp.647-648, Mar.1996