인접 블록의 움직임 벡터를 이용한 움직임 예측 알고리즘

전영현°, 윤종호, 조화현, 최명렬
{impword°, sfw1179, chh, choimy}@asic.hanyang.ac.kr

Motion estimation algorithm using motion vectors of neighboring blocks

Younghyun Jun°, Jongho Yun, Hwahyun Cho, Myungryul Choi
Dept.of EEIC, Hanyang University

요 약

본 논문에서는 인접 블록의 움직임 벡터를 이용한 움직임 예측 알고리즘을 제안하였다. 제안된 알고리즘은 두개의 탐색 단계로 구성된다. 첫 번째 단계는 인접 블록의 움직임 벡터를 이용한 탐색점에서 오차값이 제일 작은 위치를 초기 시작 위치로 사용하였다. 두 번째 단계는 첫 번째 단계에서 찾은 시작 위치에서 정확한 움직임 예측을 위한 패턴과 방법을 적용 하였다. 제안된 알고리즘은 움직임 벡터가 클 경우에 불필요한 탐색점 개수를 줄이고, 성능저하의 원인인 지역적 최소값(local minimum)에 빠질 위험을 감소시켰다. 제안된 알고리즘은 NTSS, 4SS, 1DFS, ARPS와 비교 했으며, 성능면에서는 평균 PSNR이 0.27~2.56dB 향상되었고, 탐색점 개수면에서는 평균 27개 감소 하였다.

1. 서 론

MPEG1/x 와 H.261/26x과 같은 많은 비디오 압축 표준에서는 블록 정합 움직임 예측 기법을 사용하여 시간 영역의 많은 상관관계를 가지는 부분을 제거한다. 현재 프레임의 기준 블록은 이전 참조 프레임의 탐색영역에서 최고의 정합 블록을 찾는다. 정합된 블록에서 움직임 벡터와 정합 블록간의 차이값을 부호화 한다. 블록 정합 알고리즘 중에서 전역탐색(FS: Full Search) 알고리즘은 참조블록의 탐색영역 내의 모든 블록과 현재 프레임의 기준블록을 비교하는 방법으로 가장 정확한 블록 정합을 얻을 수 있지만, 탐색영역이 커짐에 따라 많은 연산량을 필요로 하는 단점이 있다. 이러한 단점을 해결하기 위해 NTSS(New Three Step Search)[1], 4SS(Four Step Search)[2], 1DFS(One Demensional Full Search)[3], ARPS(Adaptive Rood Pattern Search)[4]등과 같은 다양한 고속 블록 정합 알고리즘이 개발되었다. 고속 블록 정합 알고리즘을 사용하면 FS보다 탐색점 개수가 줄어들기 때문에 정확도는 감소하나 탐색 속도는 향상된다.

본 논문에서는 인접 블록의 움직임 벡터를 이용한 탐색 패턴과 방법으로 불필요한 탐색점 개수와 성능저하를 줄이는 움직임 예측 알고리즘을 제안하였다. 2장에서는 전역탐색 알고리즘과 ARPS 알고리즘에 대하여 고찰하고, 3장에서는 제안하는 알고리즘에 대하여 설명한다. 4장에서는 모의실험을 통하여 성능을 비교하며, 5장에서는 결론을 맺는다.

2. 기존 블록 정합 알고리즘

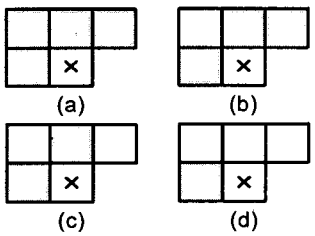
1) 전역 탐색 알고리즘 (FS: Full Search)

블록 정합 알고리즘은 현재 프레임을 기준블록(N×M의 블록)으로 나눈다. 이전 참조 프레임의 탐색영역(Search Window)에서 기준블록의 중심위치를 이동시켜 정합기준값과 비교하여 가장 최적의 정합블럭을 찾는다. 정합블럭과 기준블럭의 상대적 위치를 움직임 벡터값으로 정한다.

블록 정합 알고리즘 중에서 전역탐색 알고리즘은 모든 블록에 대해서 모든 화소를 비교하는 방법으로 성능은 가장 우수하나, 화면의 크기와 전송량이 커짐에 따라 연산량이 증가하는 단점이 있다.

2) Adaptive Rood Pattern Search (ARPS)

ARPS 알고리즘은 기준블록의 정확한 움직임 예측을 위해 두가지 요소를 고려한다. 첫 번째는 참조할 인접 블록의 타입 결정이고, 두 번째는 탐색 패턴이다.



(x : 기준 블록)

그림 1. 참조할 인접 블럭 타입

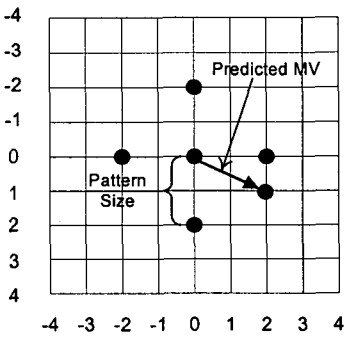
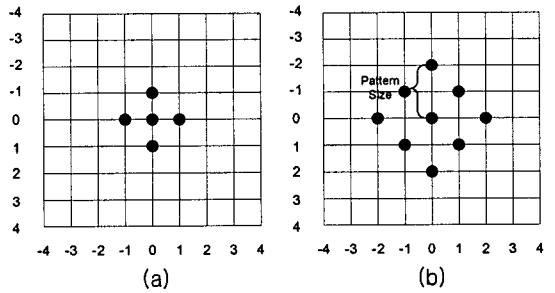


그림 2. Adaptive rood pattern(ARP)

4가지의 인접 블록 타입 중 ARPS는 그림 1.(d)와 같이 기준블록에 대하여 좌측블록의 움직임 벡터를 초기 탐색점으로 하고, 그림 2의 패턴 크기는 좌측블록의 움직임 벡터

의 x, y 값 중 제일 큰값으로 정한다. 첫 번째 단계에서 정한 5개의 탐색점 중 오차값이 제일 작은 위치를 다음 탐색의 중심 위치로 정한다.



(a) Small diamond search pattern (SDSP)
(b) Large diamond search pattern (LDSP)
그림 3. Diamond Search(DS) 탐색 패턴

그림 3은 Diamond Search(DS)[5] 알고리즘의 탐색 패턴이다. 두 번째 단계는 첫 번째 단계에서 정한 중심 위치에서 그림 3.(a)의 패턴을 이용하여 오차값이 제일 작은 위치를 그 다음 탐색 패턴의 중심 위치로 사용한다. 탐색 패턴의 중심 위치에서 제일 작은 오차 값일때 탐색을 중지하고 그때의 위치값을 기준블록의 움직임 벡터로 정한다. ARPS 알고리즘은 두 번째 단계의 탐색 패턴 크기가 작아서 영상의 움직임 벡터 크기가 작은 영상에서는 효율적이다. 그러나 움직임이 큰 영상에서 불필요한 탐색점 개수와 지역적 최소값에 빠질 위험이 있다.

3. 제안된 블록 정합 알고리즘

영상의 움직임 벡터 크기의 분포가 넓게 형성되어 있는 영상에서는 고정된 크기의 탐색 패턴은 적합하지 않다. 예를 들면 그림 3의 LDSP는 두 픽셀보다 작은 움직임 벡터를 찾기에는 불필요한 탐색을 해야 하고, 두 픽셀보다 큰 움직임 벡터를 찾기에는 패턴 크기가 작아서 많은 탐색 경로와 지역적 최소값에 빠질 수 있다. NTSS와 4SS 알고리즘은 패턴의 크기가 고정되어 있어 움직임 벡터가 클 때는 탐색 영역 크기에 증감함에 따라 움직임 벡터를 정확히 탐색할 수 없다.

제안된 알고리즘은 두개의 탐색 단계로 이루어지고, 탐색 패턴은 두가지를 사용하였다. 각 단계별 최소 오차값 산출 방법은 곱셈기를 사용하지 않고, 간단한 하드웨어 구현을 할 수 있는 차의 절대값 합(SAD: Sum of Absolute Difference)[7]을 사용하였다. 첫 번째 단계는 그림 1.(b)을 사용하여 인접한 세 개 블록의 움직임 벡터값(MVs_1, MVs_2, MVs_3), 중간값(MVm), 중간값의 반대값(MVcm)들 중에서 SAD가 최소인 위치를 구하여 다음 단계의 중심 위치로 정하였다. 첫 번째 단계는 인접한 블록의 움직임 벡터 중 가장 큰 값을 그림 3.(b) 패턴 크기로 정하였다. 탐색 패턴의 탐색점에서 제일 작은 SAD 위치를 그 다음 중심 위치로 정하였다.

그림 4는 첫 번째 단계에서 구한 중심 위치에서 FS로 찾은 움직임 벡터 차의 누적 분포도를 나타낸다. x축은 움직임 벡터 차를 나타내며, y축은 누적 분포를 나타낸다. 움직임 벡터 차의 누적값의 95% 까지 거리를 d1이라 하고 80% 이상까지 거리를 d2라 한다. 두 번째 단계에서의 그림 3.(b)의 패턴 크기를 d2로 정한다. 탐색 패턴의 최대 탐색 반복 횟수(C_{Max})를 d1/d2로 구하여 SAD가 중심 위치에서 최소일때 까지 반복한다. 마지막으로 패턴 크기가 1인 그림 3.(a) 탐색 패턴을 한번 적용하여 최종 움직임 벡터를 구한다.

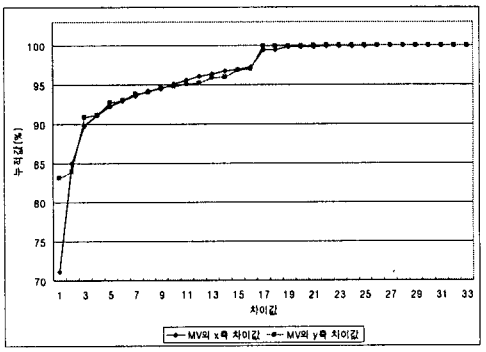


그림 4. 인접 블록간의 움직임 벡터 차의 누적 분포도

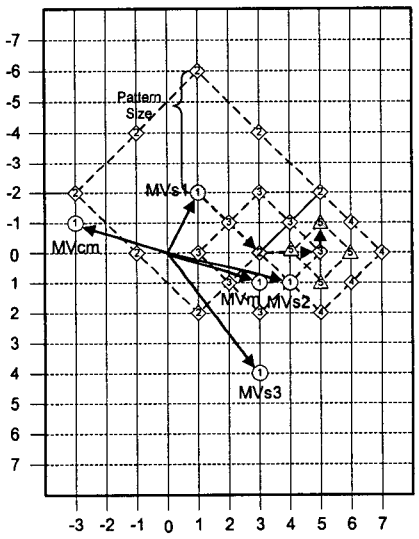


그림 5. 제안된 알고리즘의 예측 경로 예

그림 5는 제안된 블록 정합 알고리즘의 예측 경로 예를 나타낸다.

4. 모의실험 결과

본 논문에서는 움직임 예측 성능을 검증하기 위하여 PSNR (Peak Signal-to-Nosie Ratio)를 사용하였다. 모의 실험에 사용된 영상은 VGA(640×480 화소, 8 비트/화소, 30 프레임/초)인 "Flower Garden", "Susie", "Flowers"와 "Tennis" 이며 30 프레임씩 수행하였고, 움직임 벡터의 범위는 (-16,-16)~(16,16)으로 하였다.

그림 6은 전역탐색 알고리즘으로 구한 "Flower Garden" 영상의 움직임 벡터 분포도를 나타낸다.

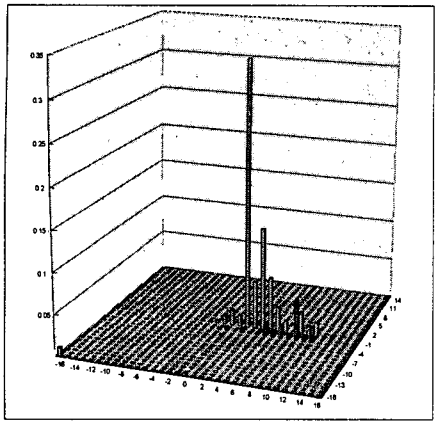


그림 6. "Flower Garden"영상의 움직임 벡터의 분포도

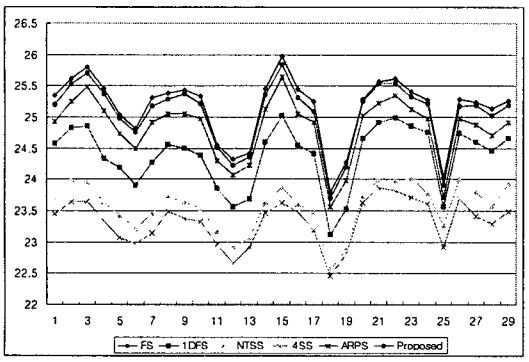


그림 7. "Flower Garden"영상에서의 PSNR(dB) 그래프

그림 7은 "Flower Garden" 영상 30 프레임으로 제안된 알고리즘을 NTSS(New Three Step Search), 4SS(Four Step Search), 1DFS(One Demensional Full Search), ARPS(Adaptive Rood Pattern Search) 알고리즘과 PSNR을 비교하였다. 중복되는 탐색점은 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
ARPS	24.82	33.29	28.00	26.76
Proposed	25.05	33.33	28.92	26.72

표 1, 2는 비교되는 알고리즘과 제안된 알고리즘에 대하여 평균 PSNR과 평균 탐색점 개수를 구한 결과이며, 제안된 알고리즘은 4SS, ARPS 알고리즘에 비해 평균 탐색점 개수는 최대 5개 정도 증가 하였지만, 평균 PSNR은 모두 향상되었음을 알 수 있었다.

* 본 연구 보고서는 정보통신부의 출연금으로 수행한 정보통신 연구개발사업의 연구결과임.

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
ARPS	17.84	18.80	17.37	18.18
Proposed	20.28	19.26	19.69	20.18

제안된 알고리즘은 비교한 알고리즘에 비해 성능면에서는 평균 PSNR이 0.27~2.56 dB 향상되었고, 탐색점 개수면에서는 평균 27개 감소하였다.

5. 결 론

본 논문에서는 인접한 블록의 움직임 벡터를 이용한 움직임 예측 알고리즘을 제안하였다. 움직임 벡터가 큰 영상에서 움직임 벡터를 찾을 때 패턴의 크기를 가변적으로 적용하였고, 최대 탐색 반복 횟수(C_{max})로 제한하였다.

제안된 알고리즘은 움직임이 큰 영상에서 불필요한 탐색점 개수와 지역적 최소값에 빠질 위험을 감소시켰다. 비교되는 알고리즘에 비해 탐색점 개수면에서는 감소되었고, 성능면에서는 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] Yao Nie and Kai-Kuang Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation", IEEE Trans. on image Processing, vol.11, no.12, pp.1442-1449, Dec.2002

[5] 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

[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