2003년도 대한전자공학회 하계종합학술대회 제26권 제1호

효과적인 동적 블록크기를 이용한 움직임 예측

김재경, 공상열, 최태선
광주과학기술원 기전공학과
신호영상연구실

Motion estimation algorithm with efficient variable block size.

Jae-kyung Kim, Sang-yeul Kong, Tae-sun Choi
Kwangju Institute of Science & Technology
Department of Mechtronics
E-mail : rla49@yahoo.co.kr

요약

본 논문에서는 효과적으로 블록 크기를 변화시키는 움직임 예측에 대하여 제시하고 있다. 블록안의 움직임의 정도에 따라 블록 크기를 채택하는 방식으로, 임의의 프레임에서의 블록의 수는 정확한 움직임을 나타내기 위해 변화된다. 이것은 움직임과 보충적인 데이터사이의 비트할당이 가변적이 되고, 프레임에 기초한 전체적인 비트율 역시 변화하게 된다. 특히, 본 논문은 동적 블록 크기 방법의 대표적인 쿼드 트리(quad tree) 방법의 단점을 보완하는 방향에서 연구되었으며, 성능 향상을 위한 새로운 방법도 아울러 덧붙여져 있다. 즉, 보통 사용하는 쿼드 트리 방식의 분할 대신에 각각의 쿼드 부분의 프레임 차를 이용하여 가장 큰 블록의 동질성 테스트를 실시하여 분할한다. 또한, 분할과 재결합 방식을 함께 적용하여 불필요한 블록의 개수가 많아 지는 것을 방지하여, 큰 계산량 감소와 높은 이미지 질을 달성하도록 하였다. 자연히, 계산량은 기존 방법보다 약 20-70 % 정도 감소 했으며, 이미지 질도 크게 향상되는 결과를 가져오게 되었다.

I. 서론

블록 매칭 방법은 블록안 각각의 화소의 움직임이 무시되고, 단일한 움직임 벡터로 모든 화소의 움직임이 대치됨을 가정하고 있다. 그러나, 같은 블록안의 화소도 움직이는 물체와 배경사이의 경계를 포함하는 블록에서와 같이, 다른 움직임을 갖을 수 있다. 그리고, 블록개수와 블록의 오류 모두 움직임 벡터가 엔코딩될 때, 동시에 최소와 되는 것이 이상적이지만, 이 두 가지는 , 특히 고정된 블록을 사용할 경우에는 서로 상반된 관계를 갖는다. 이런 문제점들은 동적 블록 크기 매칭 알고리즘으로 해결될 수 있는데, 이 동적 블록 크기 알고리즘은 정지되거나 동질한 운동을 갖는다고 판단된 블록에 대해 큰 블록을 사용하고, 복잡한 운동을 포함하는 블록에 대해서는 작은 블록을 사용하는 방법으로 이로 인해 어느 정도 블록 오류를 작게 만들어 낸다. 그런데, 이 동적 블록 크기 방법중 대표적으로 사용되는 쿼드 트리 알고리즘은 동질성 테스트를 통해 블록을 단순히 쿼드 트리 구조로 나눠 움직임 예측을 수행하는데, 지엽적인 부분의 강한 움직임에 의해 4 부분으로 나눠지며, 따라서 블록의 개수도 많아지는 단점을 갖고 있다. 또한 단순한 동질성 테스트로 인해 실질적으로 효과적인 블록 분할이 이루어 지지 못하는 점에 착안 다음과 같은 보다 효율적으로 블록을 나누고 필요 없다면 다시 합치는 방법을 제안하게 되었다. 또한, 쿼드 트리의 많아지는 계산량을 줄이기 위한 방법도 아울러 첨가 시켜 최적화 시켜 보았다.

II. 제안된 알고리즘(Proposed algorithm)

2.1 첫번째 단계

제안된 엔코딩 과정은 크게 3 가지 부분으로 나눌 수 있다. 첫번째 단계에서는 일단 동질성 테스트를 위해, 프레임차(frame difference)를 각각의 쿼드 부분에 대해 구하고, 이 값이 정하여진 문턱치 값보다 크지 않으면, 쿼드 부분의 블록을 하나의 큰 블록으로 다시 결합

시킨다. 그러나, 이 경우에도, 한 쿼드 부분이 동질성 테스트를 통과하지 못했을 경우에는(즉 그 부분의 움직임정도가 크다고 판단되나 다른 부분의 움직임이 많이 적어 움직임이 없다고 판단될 경우에) 하나의 블록으로 다시 결합시키지 않고 다음 단계로 진행한다. 이 과정에 걸리는 계산량은 기존의 쿼드 트리 알고리즘과 당연히 동일하다. 기존의 동질성 테스트를 나눠서 하는 과정이기 때문이다. 그리고 나서, 전역 탐색을 실시하여 움직임 벡터를 결정한다. 이 단계에서, 합쳐진 블록은 움직임이 적다고 판단되었으므로, 탐색 거리를 줄일 수 있다. 그래서, 본 알고리즘에서는 ±4 를 적용하였다. 이 과정에서 계산량의 감소와 이미지 질을 유지하면서, 큰 블록을 적용하기 때문에 전달되는 움직임 벡터의 수를 상당부분 줄이는 효과가 발생하게 된다.

2.2 두 번째 단계: 직행 분할(Direct Segmentation) 과 세부 탐색(Detailed Search)

다음 두 번째 단계로써, 움직임이 크고 넓게 분포한 블록으로 판단될 경우 바로 16-서브 블록(sub-block)으로 분할 한다. 여기서, 전 단계에서 계산된 프레임 차를 그대로 이용할 수 있다. 즉, 프레임 차가 문턱치 값보다 크면, 본 블록을 16 개의 작은 블록으로 나누는 것이다. 움직임이 크다고 판단되므로 이 블록에 대한 세부적인 움직임 벡터를 찾는 것은 전체 움직임 예측의 코딩 오류를 작게 하고, 계산적 노력을 신축적인 계산을 통해 줄여 줄 수 있다. 세부 탐색을 위해 이 안의 서브블록을 정지 블록(stationary block)과 움직임 블록(motion block)으로 구분 짓고, 정지 블록에 대해 작은 탐색범위를 움직임 블록에 대해 넓은 탐색 범위를 할당한다. 여기서 가장 중요한 것은 이 두 블록을 어떻게 구분 짓느냐 하는 점이다. 그래서, 연속적인 프레임 사이에서 움직임 정도를 파악할 수 있으므로 프레임 차의 블록 이미지를 일단 얻는다. 이때 노이즈(noise)가 크게 프레임차 블록 이미지에 영향을 미칠 수 있으므로, 노이즈 효과를 방지하기 위해, 로 패스 필터(low pass filter)를 블록이미지를 얻기전에 프레임에 적용한다. 그리고, 움직임 블록과 정지 블록을 구분 짓기 위해 적절한 문턱치 값을 사용해야 하는데, 프레임차 블록 이미지안의 값이 크다면, 움직임이 크다고 판단할 수 있으므로, 간단히 하기 위해, 프레임차의 평균을 문턱치 값으로 하고, 이 값을 기준으로 움직임 화소와 정지 화소로 구분 짓는다. 이때, 움직임 화소로 판단된 화소에 1 을 그렇지 않으면 0 을 할당하는 1-비트 바이너리 전환(binary transform)을 수행한다. 그리고, 서브 블록 안의 1 의 개수가 어떤 문턱치 값보다 크면, 그 블록에 대해 움직임 서브 블록으로 간주한다. 그렇지 않을 경우 정지 서브 블록으로 본다. 여기서, 기준이 되는 문턱치 값은 1 의 개수의 평균으로 하는 것이 적당하다고 보여진다. 이렇게 구분된 블록에 대해 서로 다른 탐색 범위를 설정하여 효과적인 탐색이 되도록 한다. 본 논문에서는 움직임 서브블록에 대해서는 탐색 범위를 7×7, 그리고 정지서브블록에 대해서는 2×2 로 정하였다. 이렇게 정의된 탐색범위로 전역탐색을 적용하여 움직임 벡터를 찾는다. 이 과정에서도 계산량의 감소와 코딩 오류의 감소도 같이 얻게 된다.

2.3 세 번째 단계: 분할과 재결합 단계 (Segmentation and re-merging step)

마지막으로, 위의 두 가지 단계에 적용되지 않는 블록에 대한 과정을 제시한다. 우리는 통계적인 경험상 비슷한 움직임을 보이는 블록들은 거의 같은 움직임 벡터를 갖는 다는 사실을 쉽게 알 수 있었다. 여기서, 이런 성질을 이용하여 중간크기 블록과 가장 작은 블록에 대한 재결합 과정을 다룬다.
일단, 쿼드 트리 방법과 동일하게, 프레임 차가 미리 정하여진 문턱치 값보다 크다면, 그 블록을 쿼드 구조로 나눈다. 여기서, 이 나누어진 4 개의 블록중 나누어질 필요가 없는 블록에 대해서는 다시 그 블록들을 결합시킨다. 여기서도, 절대 프레임 차를 결합의 기준으로 역시 이용할 수 있다.

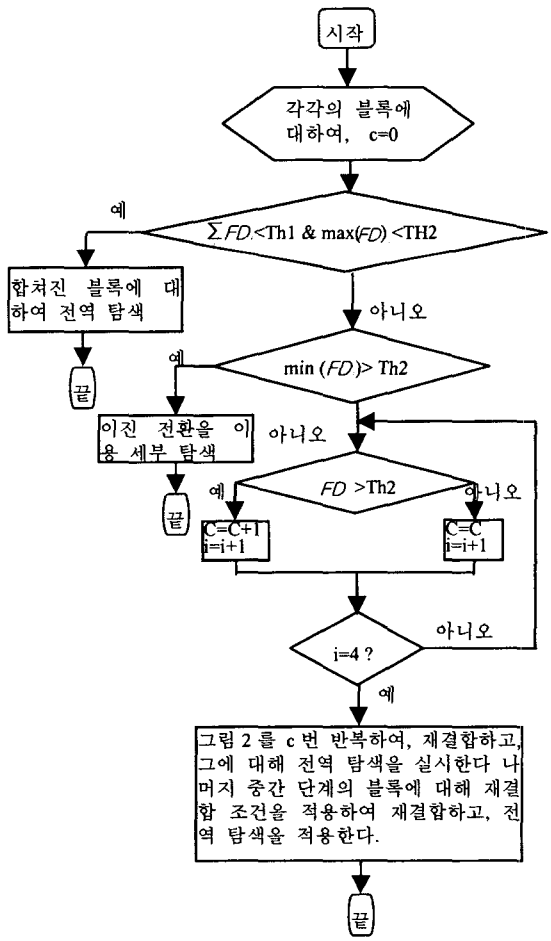

그림 1 제안 알고리즘의 순서도

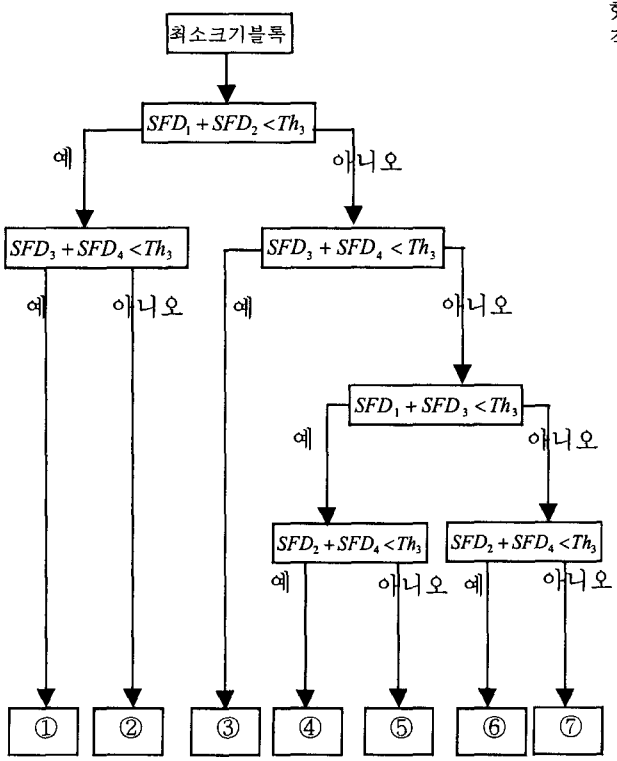

그림 2. 최소 크기 블록에 대한 재결합

가장 작은 크기 블록에 대한 결합 과정은 다음과 같다. 먼저, 왼쪽 위 위치와 오른쪽 위 위치의 서브 프레임 차의 합이 문턱치 값보다 큰지 작은지를 비교한다. 크지 않다면, 그 두개의 서브 블록에 대해 재결합과정을 수행하고 그 블록에 대해 전역 탐색을 적용하여 하나의 움직임 벡터를 할당 시킨다. 이는 이 두 서브 블록들이 작은 움직임을 갖는다고 판단 되었기 때문이다. 그리고, 나머지 두 위치의 블록에 대해서도 절대 서브 프레임 차(absolute sub-frame difference)의 합이 문턱치 값보다 크지 않다면, 재결합을 수행하여 하나의 블록으로 간주하고 그렇지 않으며, 재 결합하지 않는다. 즉, 이 세 번째 최소 크기의 서브블록에 대해서, 2 개씩 결합을 위하여, 절대 서브 프레임 차의 합을 기준으로 하여, 문턱치 값보다 크지 않으면, 재결합과정을 수행하고, 그렇지 않으면, 분할된 상태의 블록으로 간주하여 그 블록들에 대하여 전역 탐색을 적용 , 움직임 벡터를 그 블록에 할당한다.

다음으로, 중간 단계의 블록에 대한 재결합 과정은 다음과 같다. 단지 하나의 중간 크기의 블록만이 존재할 경우는 당연히 그 블록에 대해서 전역 탐색을 적용한다. 두개의 중간 크기 블록이 존재할 경우 그 블록에 대해서, 절대 프레임 차의 합을 문턱치 값과 비교하여, 재결합을 할 것인지 결정하면 된다.

세 개의 블록이 중간 단계의 블록인 경우는 보통 물체의 운동이 좌우 방향이 크다는 점을 고려하여 좌우로 인접한 블록에 대해 재결합 테스트를 먼저하고, 실패했을 경우에 한해 상하 방향에 대한 재결합 테스트를 적용한다.

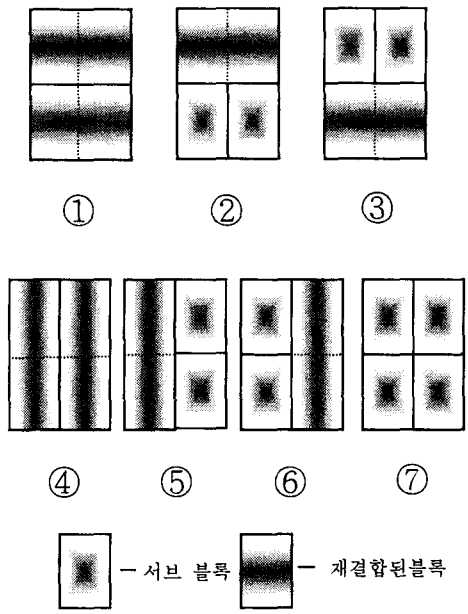

그림 3. 재결합 서브 블록의 유형

III. 시뮬레이션 결과

제안된 알고리즘에 대한 성능 평가를 위하여, 6 개의 대표적이며, 서로 다른 성격의 비디오 데이터들을 사용하였다. 그것은 세일스맨 (Salesman), 클레어 (Claire), 수지 (Suzie), 포맨 (Foreman), 카폰 (Carphone), 풋볼(football) 등이다. 움직임 예측은 화소당 8 비트로 동일하게 양자화된 비디오 시퀀스의 휘도와 움직임영상의 명암 성분에 대해 행하여 졌다. 이 시뮬레이션에서는 제안된 알고리즘을 일반적으로 가장 널리 사용되고 있는 전역 탐색알고리즘과 쿼드트리 방법을 이용한 VBS 와 비교 하였다. 최대 움직임 이동은 수평, 수직방향으로 모두 ± 7 이다.최고 신호와 노이즈대 비율은 보상된 이미지의 정확성이라는 점에서 비교되었으며, 한 블록당 평균 계산량은 속도 측면에서 나타냈으며, 블록 개수는 전달되는 움직임 벡터의 데이터량과 관계되는 것으로 성능비교를 위해 시뮬레이션하였다.

표 1 에서 볼 수 있듯이, 제안된 알고리즘은 전역 탐색과 쿼드 트리 방법보다 움직임 벡터를 정확히 빨리 찾아 내고 있음을 알 수 있다. 특히, 움직임이 비교적 높은 수지, 카폰 과 풋볼과 같은 비디오 시퀀스에서 이미지 질이 상당히 좋아졌음을 볼 수 있다. 역시, 한 블록당 평균 탐색 수도 기존 방법보다 우수함을 볼 수 있다. 거의 전역 탐색 방법보다, 20-70% 정도 계산량의 감소도 가져왔다.

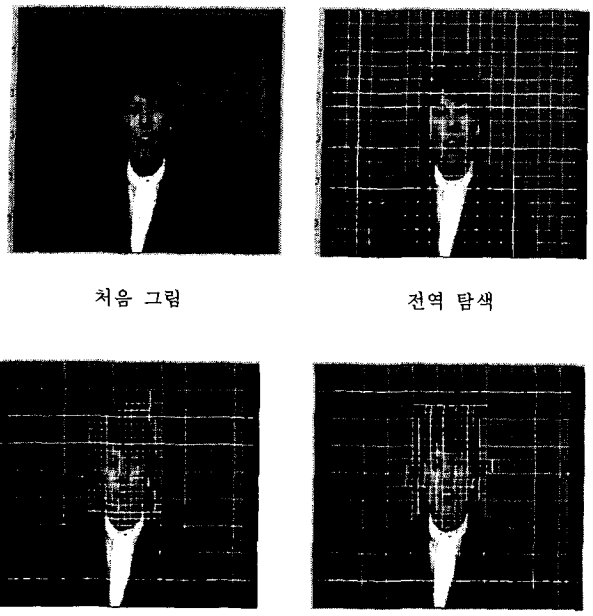

그림 4 각 알고리즘의 블록 분할

표 1 여러 알고리즘의 성능 비교

알고리즘	평균 PSNR (dB)	평균 블록수	전탐색대비 매치	알고리즘	평균 PSNR (dB)	평균 블록수	전탐색대비 매치
포맨				클레어			
제안	34.0790	416.5510	0.38	제안	42.3439	310.5918	0.54
전탐색	33.8619	396	1	전탐색	41.8608	396	1
쿼드	33.9602	458.0204	0.45	쿼드	42.3341	310.8571	0.81
수지				세일즈맨			
제안	37.9386	343.5306	0.28	제안	35.8566	392.9	0.8
전탐색	37.7407	396	1	전탐색	35.3157	396	1
쿼드	37.8584	354.5347	0.37	쿼드	35.8239	394.7	1.03
카폰				풋볼			
제안	35.1479	312.1224	0.28	제안	28.2273	1352	0.33
전탐색	34.8297	396	1	전탐색	27.7213	1320	1
쿼드	35.0043	321.5510	0.36	쿼드	28.0694	1401	0.42

IV. 결론

본 논문에서는 효과적으로 블록 크기를 변화시키는 움직임 예측에 대하여 제시하였다. 첫번째 과정에서는 블록을 나누기 위해 각 quad 부분의 프레임차(frame difference)를 먼저 계산하여 합칠 것인지를 결정하고, 두 번째에서는 모두 나눠지는 경우에 대해 상세한 전역 탐색 알고리즘을 적용한다. 그리고, 마지막으로, 각각의 가장 작은 크기로 나눠지는 경우에 대해 분할과 합성을 반복하는 방법이다. 여기서, 블록의 재합침 과정의 기준으로 나눔의 기준인 프레임 차를 그대로 적용하였다. 결과에서 알 수 있듯이 계산량과 블록의 수, 블록 오류의 정확성 모두 기존 방법보다 크게 향상된 걸 볼 수 있었다.

참고문헌

[1] J.R.Jain and A.K.Jain, "Displacement measurement and its application in interframe image coding", IEEE Trans.Commun., Vol. COM-29, No.12, pp.1799-1808,December 1981.

[2] ISO/IEC DIS 11172, " Information Technology-Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s," Draft International Standard. Nov. 1992.

[3] CCITT SG XV, Draft revision of Recommendation H.261, Document 572, Mar. 1990.

[4] T.Koya, K.Iinuma, A.Hirano, Y.Iiyima, and T.Ishiguro, "Motion compensated interframe coding for video conferencing", Proc.NTC81, pp.G5.3.1-G5.3.5, New Orleans, LA, December 1981.

[5] M.J. Chen, L.G. Chen and T.D. Chiueh, " One dimensional Motion Estimation Algorithm for Video Coding", IEEE Trans. CAS for Video Technology, Vol.4,No.5,pp.504-509, October 1994.

[6] M.H.Chan, Y.B.Yu,A.G.Constantinides, "Variable size block matching motion compensation with applications to video coding", IEE Processings, Vol.137, No.4, August 1990.

[7] V.Seferidis, M.Ghanbari, "Generalised block-matching motion estimation using quad-tree structured spatial decomposition" , IEE Proceedings-Vision Image Signal Processing, Vol.141, No.6, pp.446-452, December 1994.