서브블록을 이용한 블록 정합 알고리즘

김성희, 오정수

부경대학교

Block Matching Algorithm Using Sub-blocks

Seong-hee Kim, Jeong-su Oh

Pukyong National University

E-mail : ojs@pknu.ac.kr

요 약

본 논문은 블록 정합에 크게 기여하는 화소들만을 이용함으로 정합 계산량을 줄이는 변형된 블록 정합 알고리즘을 제안한다. 알고리즘 구현과 부가 정보를 고려하여 제안된 알고리즘은 정합 블록을 서브블록들로 나누고, 서브블록의 복잡도를 이용해 일부의 서브블록들만을 선택해 정합을 수행한다. 모의실험 결과는 제안된 알고리즘이 일부의 서브블록만으로 유효한 정합을 수행하는 것을 보여준다.

ABSTRACT

This paper proposes a modified block matching algorithm which reduces an amount of matching computation by using only pixels contributing greatly to block matching. In consideration of algorithm implementation and additional information, the proposed algorithm divides a matching block into sub-blocks, selects some sub-blocks using their complexities, and execute the block mating with them. Simulation results show that the proposed algorithm performs a valid block matching with some sub-blocks.

키워드

motion estimation, block matching, sub-block matching

Ⅰ. 서 론

오늘날 영상 생성, 영상 편집, 영상 전달을 위한 전자 장비 및 소프트웨어의 급속한 발전으로 동영상은 누구나 쉽게 접할 수 있는 정보 전달 매체가 되었다. 그러나 동영상은 방대한 데이터로 구성되어 저장이나 전송 시 압축은 필수적이다. 동영상 압축의 대표적인 기법은 영상의 시간적 중복성을 제거하는 움직임 추정 (motion estimation)이다. 이는 블록 정합 알고리즘 (block matching algorithm)을 기반으로 수행된다[1]. 블록 정합 알고리즘은 현 프레임을 일정 크기의 정합 블록 (matching block)으로 나누고, 정합 블록을 이전 프레임 탐색 영역 내의 후보 블록들과 비교하여 정합 블록과 가장 유사한 블록을 찾는 것이다. 모든 후보 블록들과 비교하는 알고리즘을 전역 탐색 알고리즘 (full search algorithm : FSA)이라 한다. 전역 탐색 알고리즘은 방대한 계산량을 요구해 시스템 구현에 큰 문제가 되고 있다. 그래서 정합 계산량을 줄이기 위한 많은 고속 알고리즘들이 연구되고 있는데 크게 제한된 후보 블록들만을 대상으로 하는 알고리즘과 정합 과정에 불필요한 후보 블록들을 제거하는 알고리즘으로 나눌 수 있다[2, 3, 4]. 전역 탐색 알고리즘과 비교하여 전자는 화질 저하가 있지만 정합 계산량을 많이 줄일 수 있고, 후자는 전자보다 정합 계산량의 감소는 다소 적지만 화질 저하가 없다.

본 논문은 정합 블록 내 일부 화소가 정합에 크게 영향을 준다는 사실에 근거해 정합 블록의 일부 화소들만을 이용한 블록 정합 알고리즘을 제안한다. 정합 화소는 에지 (edge)와 같이 불연속적인 밝기 변화가 있는 복잡한 영역에서 선택되는데 알고리즘의 구현과 부가 정보를 고려하여 작은 블록 단위로 선택된다. 그래서 제안된 알고리즘은 정합 블록을 서브블록들로 나누고, 각 서브블록의 영상 복잡도를 계산하여 복잡도가 큰 일부의 서브블록들만을 이용해 블록 정합을 수행한다. 모의실험 결과는 제안된 알고리즘이 일부의 서브블록들로도 유사 블록을 충분히 찾을 수 있음을 보여준다. 또한 제안된 알고리즘을 기존 고속 알고리즘에 그대로 적용할 수 있어 움직임 추정에서 발생하는 계산량을 크게 줄일 것이다.

II. 블록 정합 알고리즘

전역 탐색 알고리즘은 그림 1과 같이 현재 프레임의 정합 블록과 이전 프레임의 탐색 영역 내의 모든 후보 블록들과 비교하여 가장 유사한 블록을 찾는 것이다. 이때 가장 유사한 블록의 위치를 움직임 벡터라 한다. 블록간의 유사성은 식 (1)의 *SAD* (sum of absolute difference)에 의해 평가된다.

$$SAD(x,y)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|f_t(i,j)-f_{t-1}(i+x,j+y)\right| \quad (1)$$

여기서 $f_t(\cdot)$와 $f_{t-1}(\cdot)$는 각각 현재와 이전 프레임이고, N은 정합 블록의 크기이고, (i,j)와 (x,y)는 각각 정합 블록과 탐색 영역 내의 후보 블록의 위치이다. 블록 정합 알고리즘은 x,y 각각을 $-M/2$ 화소에서 $M/2$ 화소까지 이동시키면서 $(M+1)^2$의 후보 블록에 대해 *SAD*를 계산하고, 최소 *SAD*가 되는 후보 블록의 (x,y)가 움직임 벡터가 된다. 따라서 한 정합 블록의 정합을 위해 $(M+1)^2+N^2$의 덧셈과 뺄셈이 요구된다.

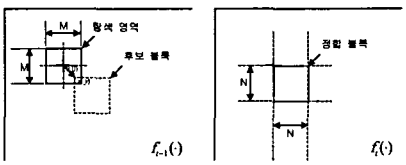


그림 1. 블록 정합

전역 탐색 알고리즘의 방대한 정합 계산량은 시스템 구현에 어려움을 주고 있다. 그래서 정합 계산량을 줄이기 위한 고속 블록 정합 알고리즘들이 제안되고 있는데 첫 번째 부류가 탐색 영역에서 □□ □ □□의 후보 블록중 일부의 후보 블록에서만 정합을 수행한다. 그 대표적인 예가 3 단계 탐색법 (three step search)이다. 3 단계 탐색법은 1 단계에서 원점을 포함한 9개 후보 블록에서 최소 *SAD*를 찾고, 2 단계에서 1 단계의 최소 *SAD* 블록을 중심으로 간격이 좁혀진 9개의 후보 블록에서 다시 최소 *SAD*를 찾고, 3 단계에서 2 단계의 최소 *SAD* 블록을 중심으로 간격이 다시 좁혀진 9개의 후보 블록에서 다시 최소 *SAD*를 찾는다. 3 단계에서 최소 *SAD*를 갖는 후보 블록의 좌표가 움직임 벡터가 된다. 만약 탐색 영역이 ± □인 경우 탐색을 위한 후보 블록이 25개이므로 정합 계산량이 전역 탐색 알고리즘의 11.11 (25/225)%이다[3]. 그러나 3 단계 탐색법은 초기 단계에서 국부 최소에 빠지는 경우 잘못된 움직임 추정을 수행하는 문제를 갖는다.

고속 블록 정합 알고리즘의 또 다른 부류는 전역 탐색 알고리즘과 같이 탐색 영역의 모든 후보 블록들에 대해 정합을 수행한다. 그러나 후보 블록의 정합 중 유사 블록의 가능성을 판단하여 불가능한 후보 블록은 정합을 중지시키는 알고리즘이다. PDE (partial difference elimination)가 대표적인 예이다. PDE는 블록 정합 오차를 계산할 때, 일정 간격 마다 그때까지의 부분 블록 정합 오차와 이전 최소 블록 정합 오차를 비교한다[4, 5, 6]. 부분 블록 정합 오차가 더 크다면, 나머지 부분 블록 정합 오차의 결과에 상관없이 해당 후보 블록은 유사 블록이 될 수 없으므로 정합 과정을 그만두고 다음 후보 블록으로 이동한다. 부분 블록 정합 오차는 식 (2)를 이용한다.

$$pSAD^k(x,y)=\sum_{i=0}^{k}\sum_{j=0}^{N-1}\left|f_t(i,j)-f_{t-1}(i+x,j+y)\right|,\quad k=1,2,\ldots,K \quad (2)$$

□는 계산되는 부분 블록의 수이고, 부분 블록 정합 오차는 정합 블록의 한 줄 혹은 그 크기의 서브블록 단위로 비교된다. PDE는 탐색 영역 내의 모든 후보 블록들에 대해서 탐색을 하기 때문에 화질은 전역 탐색 알고리즘과 같으면서도 많은 계산량을 줄일 수 있다.

III. 서브 블록을 이용한 블록정합 알고리즘

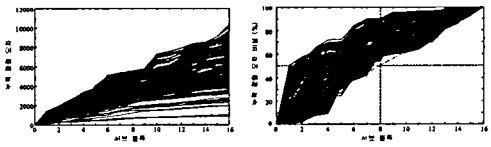
블록 정합에서 블록 내의 모든 화소들이 정합에 동등하게 기여하는 것은 아니라는 사실에 근거해 정합에 크게 기여하는 화소들만을 이용한 블록 정합 알고리즘을 제안하고 있다.

3.1 정합 특성

블록 정합이란 앞에서 언급한 것처럼 두 블록 영상의 유사성을 비교하는 것이다. 블록처럼 작은 영역에서 유사성을 평가함에 있어 밝기 변화의 불연속성을 나타내는 에지는 시각적인 비교뿐만 아니라 식 (1)과 (2)에서 정의된 정량적인 평가에서도 중요한 요소이다. 즉 에지는 시각적으로 민감하여 영상 비교에서 다른 영역보다 우선되어지고, 잘못된 결과는 시각적으로 더 분명히 나타난다. 그리고 정량적 평가에서도 에지 영역이 다른 영역 보다 더 크게 나타난다. 따라서 에지 영역은 블록 정합 알고리즘에서 다른 영역보다 더 기여도가 크다고 할 수 있다.

그림 2은 정합 블록 내의 화소들이 정합 오차에 미치는 영향을 보여주기 위한 것이다. 그림 2(a)는 한 정합 블록의 움직임 추정을 위해 탐색되는 225개의 후보 블록에 대한 정합 오차를 누적한 것이고, 그림 2(b)는 누적 정합 오차를 백분율로 표시하고 있다. 여기서 정합 오차는 정합 블록을 4x4 서브블록들로 나누어 서브블록 단위로 계산되고 있고, 에지의 영향을 보기 위해 에지가 큰 서브블록의 정합 오차를 먼저 계산하고 있다. 그림 2(a)에서는 최소 정합 오차를 갖는 후보 블록은 서브블록의 수에 관계없이 최소 혹은 그에

근사하는 부분 정합 오차를 갖는 것을 알 수 있다. 이는 일부의 서브블록으로도 적절한 유사 블록을 찾을 수 있음을 의미한다. 그림 2(b)에서 누적 정합 오차가 전반부에서 큰 값을 갖는 것은 에지가 큰 서브블록들에 의한 오차가 전체 정합 오차의 상당 부분을 차지하고 있다. 이는 에지 영역이 블록 정합에 있어 기여도가 큼을 의미한다. 대각 점선은 모든 화소가 같은 오차를 가질 때 누적 정합 오차 비율이고, 누적 정합 오차의 후반부에서 급격하게 증가하는 오차는 정합 블록이 아닌 후보 블록의 에지에 의한 영향이다.



(a) 누적 정합 오차 (b) 누적 정합 오차 비율

그림 2. 누적 정합 오차

3.2 정합 화소의 선택

제안된 알고리즘처럼 제한된 화소를 이용한 블록 정합에서 정합 화소를 선택하는 것은 가장 중요한 요소이다. 정합 특성에서 나타난 것처럼 에지 영역의 화소들은 다른 영역 화소들보다 블록 정합에 더 크게 기여하고 있다. 그래서 본 논문에서는 에지 영역을 표현하기 위해 영상 복잡도를 정의하고, 영상 복잡도가 높은 영역의 화소들을 정합 화소로 사용한다. 화소 단위의 영상 복잡도는 식 (3)과 같은 마스크를 이용해 계산한다.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3)$$

정합 화소를 화소 단위로 선택할 때 그들을 위한 부가 정보가 많아지는 문제가 발생한다. 그래서 본 논문에서는 16x16 정합 블록을 16개의 4x4 서브블록들로 나누고, 식 (3)으로 계산된 화소 단위 복잡도를 서브블록 단위로 합산하여 서브블록 단위의 복잡도를 계산하고, 복잡도가 큰 서브블록을 정합 화소들로 선택한다. 정합에 사용되는 서브블록의 수는 필요에 따라 결정된다.

3.3 서브블록을 이용한 블록 정합 알고리즘

본 논문에서 제안된 서브블록을 이용한 블록 정합 알고리즘은 다음과 같은 과정을 통해 수행된다.

(1) 화소 단위의 영상 복잡도를 계산한다.
(2) 정합 블록을 4x4의 서브블록으로 나눈다.
(3) 서브블록에서 16 화소의 영상 복잡도를 더해 서브블록의 영상 복잡도를 계산한다.
(4) 정합 블록에서 서브블록의 영상 복잡도를 크기에 따라 내림차순으로 정렬하고 정해진 수의 서브블록을 선택한다.
(5) 선택된 서브블록들만을 이용해 블록 정합을 수행한다.

제안된 알고리즘은 영상의 화질과 정합 계산량을 고려하여 블록 정합을 위해 사용되는 서브블록의 수를 조절할 수 있고, 기존 고속 블록 정합 알고리즘에 적용시킬 수 있다. 기존 알고리즘에 적용시킬 때 전송 채널 상태에 따라 서브블록의 수를 조절하면 비트율 제어 (bit rate control)에도 활용할 수 있다.

IV. 모의 실험 결과

본 장에서는 제안된 알고리즘의 성능을 평가하기 위해 서브블록의 수에 따라 제안된 알고리즘과 전역 탐색 알고리즘의 움직임 벡터가 일치하는 정합 블록의 수를 확인하고, 화질을 평가하고 있다. 모의실험은 144x176인 QCIF 해상도의 'foreman', 'stefan', 'carphone'의 30[Hz]와 15[Hz] 연속 영상을 가지고 수행되었다. 여기서 탐색 영역은 정합 블록을 중심으로 ±7이고, 정합 블록 크기는 16x16이다.

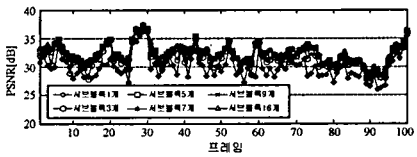
표 1은 30[Hz] 100프레임과 15[Hz] 50프레임의 실험 영상에서 서브블록의 수에 따른 제안된 알고리즘의 두가지 성능을 비교하고 있다. 첫 번째 성능은 검출된 움직임 벡터와 전역 탐색 알고리즘의 움직임 벡터와 일치하는 평균 정합 블록의 수를 보여주고 있다. 1개의 서브블록만을 사용해도 30[Hz] 영상에서는 영상에 따라 51~67[%]의 정합 블록이, 15[Hz] 영상에서는 42~51[%]의 정합 블록이 전역 탐색 알고리즘과 움직임 벡터가 일치하는 것을, 그리고 모든 서브블록을 사용해야만 전역 탐색 알고리즘과 움직임 벡터가 일치하는 경우는 5개의 정합 블록도 안 되는 것을 확인시켜 주고 있다. 결과적으로 블록 정합 알고리즘에서 블록 정합을 위해 전체 화소를 필요로 하는 정합 블록은 소수에 불구하고, 대부분의 정합 블록에서는 필요이상의 화소들이 사용되고 있다.

두 번째 성능은 움직임 벡터를 검출하여 재구성한 영상의 평균 화질을 PSNR로 보여주고 있다 이는 최적의 움직임 벡터를 검출하지 못 했을 때 영상에 미치는 영향을 검토할 수 있다. 서브블록을 1개만 사용할 때의 화질은 전체 서브블록을 사용할 때의 화질의 82~ 90[%]을 유지하고 있다. 그리고 30[Hz] 영상과 15[Hz] 영상에서 각각 7개 이상의 서브블록과 8개 이상의 서브블록을 사용하면 전체 블록을 사용한 경우의 화질의 99[%] 이상을 유지하고 있다. 이는 서브블록을 사용하여 최적의 유사 블록을 검출하지 못해도 그에 근사한 유사 블록을 검출하므로 그 영향이 적음을 나타내고, 일정 수 이상의 서브블록을 사용하면 화질 열화를 느낄 수 없음을 알 수 있다. 표에서 적은 수의 서브블록을 사용했는데도 미세하게 PSNR이 높게 나오는 경우가 발생하는데 이는 블록 정합시 평가 기준과 PSNR 계산시 평가 기준의 다르기 때문이다.

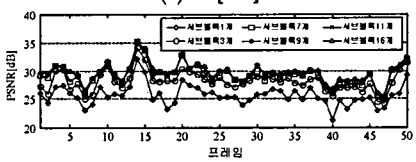
표 1. 서브블록의 수에 따른 알고리즘의 성능

서브블록수	움직임 벡터가 FSA와 일치하는 평균 블록의 수						평균 PSNR[dB]					
	30Hz			15Hz			30Hz			15Hz		
	carphone	foreman	stefan	carphone	foreman	stefan	carphone	foreman	stefan	carphone	foreman	stefan
1	50 52	51 58	66 15	42 12	43 02	51 00	27 99	28 01	23 02	25 65	25 65	20 28
2	63 90	62 00	74 86	56 70	54 62	61 66	31 03	30 03	24 04	29 37	27 47	21 29
3	69 51	68 48	79 16	62 50	60 98	67 68	32 41	31 08	24 52	30 82	28 15	21 74
4	74 22	72 50	82 23	67 76	65 74	70 92	33 08	31 59	24 76	31 25	28 70	21 94
5	77 54	75 87	84 72	72 10	68 98	74 20	33 47	31 99	24 99	31 60	29 07	22 16
6	80 00	78 12	86 62	75 24	72 00	76 56	33 64	32 21	25 08	31 77	29 28	22 26
7	81 96	80 55	88 63	77 16	75 20	78 52	33 78	32 30	25 15	31 91	29 31	22 36
8	83 73	82 99	89 97	79 54	77 58	80 78	33 86	32 42	25 21	32 01	29 49	22 41
9	85 45	85 00	90 95	81 78	80 64	83 28	33 93	32 49	25 22	32 05	29 57	22 48
10	87 20	86 78	92 46	83 46	82 44	84 86	33 95	32 53	25 27	32 08	29 63	22 52
11	89 02	88 18	93 41	85 58	84 98	87 00	33 98	32 54	25 29	32 14	29 67	22 56
12	90 35	90 17	94 37	87 44	87 48	88 26	34 00	32 55	25 31	32 15	29 69	22 60
13	91 72	92 01	95 36	89 28	89 56	89 76	34 00	32 56	25 32	32 17	29 68	22 62
14	93 18	94 03	96 35	91 82	92 04	92 46	34 02	32 57	25 33	32 17	29 71	22 65
15	95 19	96 05	97 26	94 50	94 74	94 68	34 01	32 56	25 34	32 17	29 70	22 66
16	99 00	99 00	99 00	99 00	99 00	99 00	34 01	32 55	25 34	32 16	29 70	22 67

그림 3와 4는 각각 연속 영상 'foreman'과 'stefan'에서 서브블록의 수에 따른 재구성된 영상의 화질을 프레임별로 보여주고 있다. 30[Hz] 영상에서는 서브블록의 수를 1, 3, 5, 7, 9, 16 개를 사용하고, 15[Hz] 영상에서는 서브블록의 수를 1, 3, 5, 9, 11, 16 개를 사용하였다. 30[Hz] 영상에서는 5개 이상의 서브블록, 15[Hz] 영상에서는 7개 이상의 서브블록을 사용하면 제안된 알고리즘은 FSA와 대등한 화질을 보여준다. 그리고 'stefan'처럼 복잡한 영상일수록 제안된 알고리즘은 더 효율적이다.

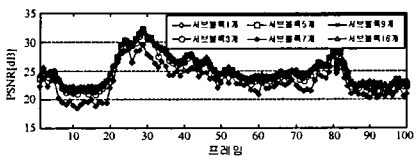


(a) 30[Hz]

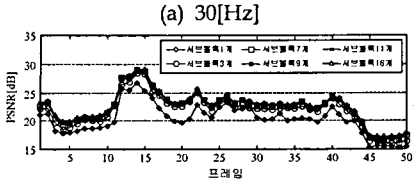


(b) 15[Hz]

그림 3. 연속 영상 'foreman'



(a) 30[Hz]



(b) 15[Hz]

그림 4. 연속 영상 'stefan'

V. 결 론

본 논문에서는 블록 정합에 크게 기여하는 화소만을 이용한 블록 정합 알고리즘을 제안하였다. 제안된 알고리즘은 정합 화소를 선택하기 위해 영상 복잡도를 정의하고, 알고리즘 구현과 부가정보를 고려하여 정합 블록을 서브블록들로 나누고, 서브블록 단위로 복잡도를 계산해 복잡도가 높은 일부의 서브블록들만을 이용해 정합을 수행한다. 제안된 알고리즘의 성능 평가를 위해 연속 영상에서 서브블록의 수를 변화시키면서 전역 탐색 알고리즘의 결과와 비교하였다. 모의실험 결과 적은 수의 서브블록으로도 충분히 유사 블록을 검출할 수 있음을 확인시켜 주었고, 정확한 유사 블록을 검출하지 못해도 근사 블록을 검출하여 화질에 영향이 적었고, 일정 서브블록 이상을 사용하면 전역 탐색 알고리즘과 대등한 화질을 보여주었다.

제안된 알고리즘은 기존 고속 움직임 추정 알고리즘에 그대로 적용시킬 수 있어 추가적으로 계산량을 감소시킬 수 있고, 서브블록의 수를 전송로의 상태에 따라 제어하면 전송 비트율을 제어할 수도 있다.

참고문헌

[1] Frederic Dufaux and Fabrice moscheni, "Motion Estimation Technique for Digital TV : A Review and a New Contribution," Proceedings of the IEEE, vol.83, pp.858~876, Jun. 1995.

[2] J.R.Jain and A.K.Jain,"Displacement measurement and its application in interframe image coding,"IEEE Trans. Commun, vol. 29, pp. 1799~1806, Dec. 1981.

[3] T.Koga, K.Iinuma, A.Hirano, Y.Iigima and T.Ishiguro, Motion Compensated interframe coding for video conferencing, " Proc. Nat. Telecommun. Conf.. pp. G5.3.1-5.3.5, New Orleans, USA, Nov. 1981.

[4] ITU-T Recommendation H.263 software implementation, Digital Video Coding Group at Telenor R&D, 1995.

[5] J.N.Kim and et al., "Adaptive matching scan algorithm based on gradient magnitude for fast full search in motion estimation," IEEE Trans. Consumer Electronics, vol.45, pp. 762~772, Aug.1999.

[6] J.N.Kim and et al., "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan,"IEEE Trans. Circuits Syst. For Video Technol., vol. 10,pp. 1040~1048. Oct.2000.