적응적 탐색범위를 사용한 블록정합 알고리듬

강 문 철, 배 황 식, 정 정 화
한양대학교 전자통신전파공학과 CAD 및 통신회로 연구실
전화 : 02-2290-0558 / 핸드폰 : 011-9530-3672

A fast block matching algorithm with adaptive search range

Abstract

본 논문에서는 MPEG-2, MPEG-4, H.263 등에서 블록정합을 위해 사용되는 움직임 추정(Motion Estimation) 기법에서 적응적 탐색 범위를 기존의 알고리듬에 적용시킴으로써 계산량을 줄이고 화질도 개선하는 방법을 제안한다. 제안된 알고리듬은 먼저 이웃한 움직임 벡터(Motion Vector)의 위치를 이용하여 예상된 움직임 벡터를 찾고 이 예상된 움직임 벡터의 X, Y 값의 크기를 작은 값, 중간 값, 큰 값, 세 가지로 분류해서 탐색범위를 적응적으로 변화시켜 움직임 벡터가 있을 확률이 큰 범위를 집중적으로 찾는다. 그리고 각 분류에서 작은 값일 때는 전역 탐색을 적용하고 큰 값일 때는 기존의 알고리듬을 적용시키고 중간값일 때는 3단계탐색 기법을 적용시켜 더 적합한 움직임 벡터를 찾도록 하였다. 그리고 작은 값일 때 구해진 움직임 벡터의 SAD(Sum of Absolute Difference) 값과 이웃한 움직임 벡터의 SAD값을 비교해 국소점에 빠졌다고 판단이 되면 다시 탐색 범위를 조정해서 움직임 벡터를 구함으로써 국소점에 빠지는 경우를 줄였다.

I. 서론

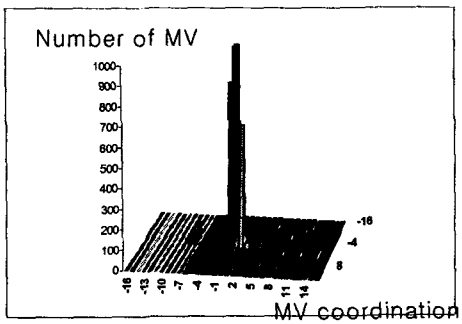
최근 인터넷과 무선 통신등의 발달로 이를 이용한 동영상 서비스에 대한 요구가 높아지고 있다. MPEG-2, MPEG-4, H.26L등에서는 보통 한 프레임 내의 공간적인 중복성을 없애기 위해 DCT/Q, Wavelet 같은 알고리즘을 쓰고, 통계적 중복성을 없애기 위해서 Huffman Coding, Arithmetic Coding을 이용하고, 프레임간의 시간적 중복성을 없애기 위해서는 움직임 추정(Motion Estimation) 방법을 적용한다. 특히 동영상에는 시간적인 연관성이 많기 때문에 이를 이용한 움직임 추정 기법을 사용해서 데이터의 압축효율을 높일 수 있다.

움직임 추정 방법 중 전역탐색은 정해진 탐색 범위를 전부 검사하기 때문에 가장 오차가 적은 움직임 벡터를 찾을 수는 있지만 계산량이 많이 요구되기 때문에 실제로 적용시키기가 어렵다. 이를 개선하기 위해서 많은 알고리듬들이 제안되었다. 탐색점을 줄이는 새로운 3단계탐색 기법[1], 4단계탐색 기법[2], DS(Diamond Search) 기법[3], 작은 해상도에서 움직임 벡터를 찾은 다음 원래의 해상도에서 더 자세하게 찾는 Multi-resolution 기법[4][5][7], 움직임 벡터의 시간적 공간적 상관성을 이용한 기법[5]들이 있다.

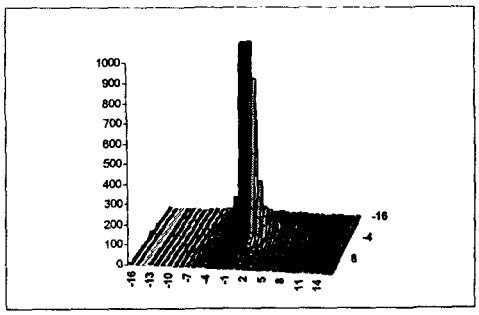
실제 실험한 결과 그림1과 같이 움직임 벡터는 대부분이 ±2 범위 내에 있는 걸 알 수 있고 이웃한 움직임 벡터와 비슷한 위치에 존재한다는 것을 알 수 있다. 위의 제안된 알고리듬들은 탐색 범위의 전 영역을 사용하기 때문에 비효율적일 수 있고 국소점에 빠질 수도 있다.

본 논문에서는 이웃한 움직임 벡터의 값의 중간값을 취해서 예상된 움직임 벡터를 구하고, 이렇게 구해진 움직임 벡터의 크기를 세 가지로 분류하여 각각의 경우에 대해서 해당되는 알고리듬을 적용시킨다.

본 논문의 구성은 II장에서는 제안된 알고리듬을 설



Claire



Football

그림1. Claire와 Football 영상의 움직임 벡터의 분포

명하고 III장에서는 실험 결과를 보이고 IV장에서는 결론을 맺는다.

II. 제안한 움직임 추정 알고리듬

이 장에서는 본 논문에서 제안하는 알고리듬에 대해서 설명한다. 제안된 알고리듬은 기존의 움직임 추정에 쓰이는 알고리듬들에 적용시켜서 상대적으로 탐색 범위가 작은 경우에는 전역 탐색과 3단계탐색 기법을 사용하고 범위가 큰 경우에는 적용시킨 본래의 알고리듬을 사용한다. 본 논문에서는 3단계탐색 기법에 적용시켜서 실험을 하였다.

이전에 제안된 알고리듬[6]에서는 적응적인 탐색 범위를 적용시키지만 한 화면을 두 그룹으로 분리하고 한 그룹을 먼저 일반적인 알고리듬을 적용시킴으로써 한 화면에 있는 매크로 블록의 절반만 적응적인 탐색 범위를 적용시킬 수 있다. 그렇지만 본 논문에서 제안된 알고리듬은 그림2와 같이 위와 좌·우 경계되는 부분을 제외한 나머지 전 부분에서 적응적으로 탬색 범위를 적용할 수 있다.

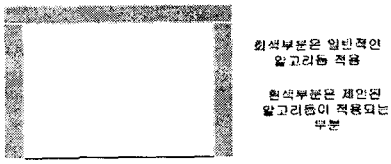


그림2. 한 화면 내에서의 알고리듬 적용범위

제안된 알고리듬은 다음과 같은 순서로 진행된다.

첫째, 그림4와 같이 이웃한 세 개 블록의 움직임 벡터를 이용하여 현재 블록의 예측된 움직임 벡터를 구한다. [7]

둘째, 첫 번째에서 구한 예측된 움직임 벡터의 X, Y 값의 절대치의 크기를 세 가지로 분류해서 X, Y 값 각각이 1보다 작거나 같을 경우는 탐색 범위를 ±1로 정해서 전역 탐색을 실시하고 1보다 크고 3같거나 작으면 탐색 범위를 ±4로 정해서 이 범위에서 우수한 성능을 나타내는 3단계탐색 기법을 적용시킨다. 그리고 위의 두 경우가 아니면 탐색 범위와 알고리듬은 본래의 것을 사용한다.

셋째, 두 번째에서 탐색 범위가 ±1 일 경우는 탐색 범위가 매우 작으므로 국소점에 빠질 확률이 크다. 그러므로 탐색 범위가 ±1 일 때 구한 움직임 벡터의 SAD값이 주위 블록의 SAD값과 비교해서 식(3)의 문턱값을 넘어가면 국소점에 빠졌다고 판단하고 탐색 범위를 ±4 로 정해서 3단계 탐색 기법을 적용하고 아니라면 그 움직임 벡터를 현재 블록의 움직임 벡터로 한다.

그림3는 제안된 알고리듬의 순서도를 나타내고 있다. 어두운 부분은 위에서 설명된 각각을 표시하고 있다. A부분은 이웃한 움직임 벡터를 이용하여 현재 블록의 움직을 벡터를 구하고 B 부분에서 탐색 범위를 결정하고 C 부분에서는 국소점에 빠졌는지를 검사한다.

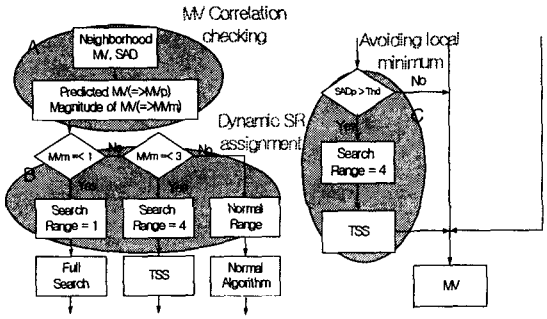


그림3. 제안된 알고리듬의 순서도

현재 블록의 움직임 벡터를 예측하기 위하여 그림2와 같이 상, 우상, 좌측 블록의 움직임 벡터를 이용한다[7]. 현재 블록의 예측된 움직임 벡터는 아래의 식과 같이 이웃한 블록의 움직임 벡터 MV_1, MV_2, MV_3 의 X, Y 좌표 각각의 중간값을 계산해서 구한다.

$MV_1 = (X_1, Y_1)$, $MV_2 = (X_2, Y_2)$, $MV_3 = (X_3, Y_3)$

$MVp = (Median(X_1, X_2, X_3), Median(Y_1, Y_2, Y_3))$

식(1) : 예측된 움직임 벡터 계산

그림4와 같이 이웃한 블록을 선택하면 제일 근접한 블록들이므로 공간적인 상관성도 높게 되고 하드웨어로 구현할 때 한 행에 대한 정보만 저장하면 되므로 메모리를 적게 차지하게 된다.

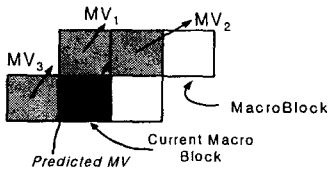


그림4. 움직임 벡터 예측

2.1 탐색 범위와 알고리듬 결정

현재 블록의 예측된 움직임 벡터의 크기(MVm)를 세 가지로 분류하여 각각의 경우에 해당하는 탐색 범위와 알고리듬을 적용한다. 분류는 다음식에 따른다.

(i) $|Xp| \le 1$ and $|Yp| \le 1$

(ii) $1 < |Xp| \le 3$ and $1 < |Yp| \le 3$

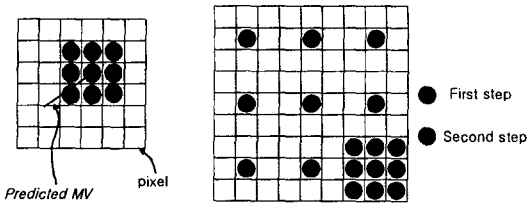
(iii) $|Xp| > 3$ and $|Yp| > 3$

식 (2) : 탐색 범위를 분류하기 위한 조건식
(단, MVp = (Xp, Yp))

(i)의 경우 주위 블록의 움직임이 작으므로 현재 블록의 움직임도 크지 않다고 판단하고 그림 5의 a)에서 보는 바와 같이 탐색 범위를 ±1로 정한다. 그리고 이 범위 내에서 전역 탐색을 수행한다.

(ii)의 경우에는 탐색 범위를 ±4로 한다. 이 범위일 경우는 그림5의 b)에서 보는 바와 같이 연산량과 화질을 고려할 때 3단계탐색 기법을 적용하는 것이 적당하다고 판단되었다.

(iii)의 경우에는 움직임이 크기 때문에 이웃한 블록과 현재 블록과의 공간적 상관성이 별로 없다고 판단해서 이웃한 블록의 움직임 벡터를 이용해서 움직임 벡터를 구하지 않고 원래의 알고리듬을 그대로 적용해서 움직임 벡터를 구한다.



(a) (i)의 경우 (b) (ii)의 경우

그림5. 예측된 움직임 벡터의 크기에 따른 적응적인 탐색 범위와 알고리듬

2.2 국소점 판별과 움직임 벡터

2.1절의 (i)의 경우에 구한 움직임 벡터는 탐색 영역이 매우 작기 때문에 국소점에 빠질 수가 있다. 이것을 보완하기 위해 제안된 알고리듬의 마지막 단계에서 이 경우 구해진 움직임 벡터가 국소점에 빠졌는지를 식(3)을 가지고 판별해서 새로 움직임 벡터를 구한다. 움직임 벡터가 국소점에 빠졌는지를 판단하기 위해 이웃한 블록의 SAD와 비교를 한다. 다음 3식을 만족하면 현재 블록이 이웃한 블록보다 정합에러가 상당히 크게 되므로 국소점에 빠졌다고 가정할 수 있다.

$SADp > SADn * 1.2$ -- 식 (3)

(SADp : ±1 탐색범위에서 구해진 움직임 벡터로 인한 SAD
SADn : 그림4에서 세 개의 블록의 SAD값의 평균치
SADn * 1.2 : 문턱값(Thd))

국소점에 빠졌다고 판단이 되면 2.1절의 (ii)을 실행하고 여기서 나온 SAD와 위의 SADp의 값을 다시 비교한 후 더 작은 값을 가지게 되는 움직임 벡터를 취하게 된다.

III. 실험 결과

제안된 알고리듬을 검증하기 위해 시험 영상중 움직임이 적은 Claire(352 X 288) 100 프레임과 Miss America(352 X 288) 100 프레임, 움직임이 많은 Football(352 X 240) 100 프레임을 가지고 모의 실험을 하였다. 기본 탐색 범위는 ±16으로 하고 3 단계 탐색 기법에 제안된 알고리듬을 적용시켰다. 화질의 평가를 위해서 PSNR(peak signal noise ratio)를 사용하였고 계산량 비교를 위해서는 실제 한번의 연산(한 픽셀과

한 픽셀의 차)을 이용하였다.

다음 표에서 보는 바와 같이 정적인 영상인 Claire와 Miss America 의 경우에는 3단계 탐색 기법만 썼을 경우보다 계산량은 감소하고 화질은 좀 더 좋아진 것을 알 수 있다. 움직임이 큰 Football 영상의 경우는 계산량은 45%가량 줄었지만 화질은 약간 더 나빠진 것을 알 수 있다.

	Claire	Football	MissA
FS	41.1377	22.9135	37.6733
TSS	41.0277	21.9713	37.0858
Proposed	41.0391	21.3404	37.2157

표1. Claire와 Football 영상의 PSNR

	Claire	Football	MissA
FS	1	1	1
TSS	0.031	0.031	0.031
Proposed	0.015	0.017	0.027

표2. Claire와 Football 영상의 전역 탐색 기법의 계산량을 1로 보았을 때 상대적인 계산량

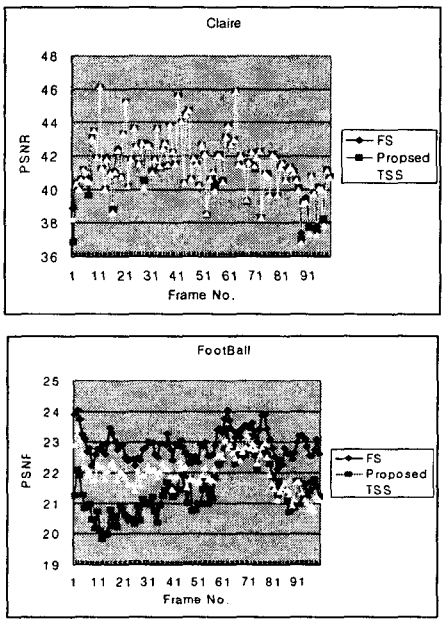


그림6. Claire와 Football영상의 각 프레임별 PSNR

Ⅳ. 결론 및 추후 연구

본 논문에서는 움직임 벡터의 특징인 공간적 상관성을 이용하여 탐색 범위를 적응적으로 변화시키면서 움직임 벡터를 찾았다. 탐색 범위를 적응적으로 변화시킴으로써 탐색 영역을 전부 사용하지 않아 계산량을 많이 줄일 수 있었다.

추후에 더 많은 실험을 통해서 문턱값과 탐색 범위를 조정하면 더 좋은 결과를 얻을 수 있을 것이다.

참고문헌

[1] R. Li, B.Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation", *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 4, No. 4, pp. 438-442, Aug. 1994

[2] Lai-Man Po and Wing-Chung Ma, "A novel four-step search algorithm for fast block motion estimation", *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 6, No. 3, pp. 313-317, Jun. 1996

[3] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block-matching motion estimation", *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 9, No. 2, pp. 287-290, Feb. 2000

[4] Jinwen, M. Omair Ahmad, and M. N. S. Swamy, "New techniques for multi-resolution motion estimation", *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 12, No. 9, pp. 793-802, Sep. 2002

[5] Junavit Chalidabhongse and C.C Jay Kuo, "Fast motion vector estimation using multiresolution spatio-temporal correlations", *IEEE Transaction Circuits and Systems for Video Techology*, vol. 7, No. 3, pp. 477-488, Jun. 1997

[6] Lurng, "Dynamic search range motion estimation for video coding", *IEEE First Workshop Multimedia Signal Processing, 1997*, pp 207-212, Jun. 1997

[7] Jae Hun Lee, Kyoung Won Lim, Byung Cheol Song, and Jong Beom Ra, "A fast multi-resolution block matching algorithm and its LSI architecture for low bit-rate video coding", *IEEE Transaction on Circuits and Systems for Video Techology*, vol. 11, No. 12, pp. 1289-1301, Dec. 2001