Proceedings of the 11th
KACC, October 1996

Quadtree 를 이용한 가변 block 움직임 추정

The Variable-sized block matching motion estimation using Quadtree

°이 원희*,김 상기*,변 재응*,김 재영*,이 범로*,정 진현*

*광운 대학교 제어계측 공학과

Abstracts: The block matching algorithm for the motion estimation is relatively simple to implement, and thus widely applied in image sequence coding such as H.261, MPEG-1 and MPEG-2. Most techniques of the block matching method use fixed-size blocks for the motion estimation. And their success relies on the assumption that the motion within each block is uniform. But if the block size is increased to reduce the number of motion vectors for high data compression, the estimated image brings about many errors. In this paper, the variable-sized blocks are usedto solve this problem. And the top down method is used to select the block size.

Keywords: Block-matching, Block size, Quad-tree, Bottom-up, Estimation

1.서론

움직임 보상은 video 이미지의 코딩에서 중요한 과제이다. 현재 프레임의 예측은 하나 또는 하나 이상의 전 프레임으로부터 이루어 진다. 그리고 이 움직임 예측과 그에 따른 에러가 저장 매체에 저장 되거나 전송 매체에 의해 전송 된다. 그러므로, 예측에 의해 부호화한 프레임과 그로 인한 데이터의 압축율의 성과는 전적으로 움직임 보상과 에러 신호의 정확성에 달려 있다. 말하자면 움직임 보상으로 복원된 이미지의 정확성과 얼마나 적은 데이터량으로 압축이 이루어 졌느냐 하는 문제가 바로 그것이다. 현재 비디오 코덱의 표준안인 H.261 이나 MPEG 등에서는 기존의 BMA(Block Matching Algorithm)를 채택하고 있다. 이 방법은 이미지 프레임을 일정한 크기의 block 들로 나누고, 그리고 각각의 block 은 같은 움직임을 갖는다고 가정한다. 그리고 현재 프레임의 각 block 에 대해 전 프레임에서 가장 잘 matching 되는 block 을 찾게 된다. 이때 가장 잘 matching 되는 block 을 찾는 기준은 mean square error 나 mean absolute error 같은 에러 값으로 결정한다. 가장 잘 matching 되는 block 을 찾게 되어 두 블럭 간의 displacement 를 움직임 벡터로 취하게 된다. 여기서 계산 속도나 정확도는 search method, search window size 등에 따라 달라진다. 그러나 이 방법은 작은 block 으로 나누어 수행하기에는 너무 많은 계산 시간이 걸리게 된다. 보통 이 방법은 16 × 16 block 을 사용하는데 이는 경계부분에서 심한 blocking 현상을 야기 시킬 수 있다. 이를 개선 하기 위해 기존의 fixed-size block matching 과는 다르게 variable-size block matching 방법을 도입한다. 즉 움직임이 없는 부분에 대해서는 크기가 큰 block 을 지정하고 많은 움직임을 갖는 부분에 대해서는 block 의 크기를 작게 지정한다. 이렇게 함으로써 움직임이 많은 부분은 더 좋은 움직임 보상이 이루어 질 수 있다. 그러나 이 variable-size block matching 방법에서 가장 중요한 문제가 되는것은 바로 서로 다른 크기의 block 을 어떻게 할당하는가 하는 것이다. 바로 이 문제가 움직임 보상의 좋은 결과와 뛰어난 압축율을 뒷받침 하게 된다. block 의 크기를 결정하는 방법에는 대략 2 가지로 나눌 수 있다. 먼저 이미지 프레임을 초기에 상당히 큰 block 으로 취하고 motion estimation 을 수행한다. 그리고 각 block 에 대해서 가장 잘 matching 되는 block 과의 error 가 우리가 먼저 정해 놓은 threshold 값보다 작으면 그 block 은 그대로 결과로 나온 움직임 벡터를 채택하고 error 값이 크게 되면 더 작은 block 으로 쪼개게 된다. 이 과정을 미리 설정해 놓은 값(최대 block 수 또는 최소 error)에 도달 할때 까지 수행한다. 두번째로는 이미지 프레임을 초기에 아주 작게 나누고 motion estimation 을 수행한다.그리고 같은 움직임 벡터를 갖는 block 들을 서로 묶어 주는 방법이다. 이 두가지가 variable-size block matching algorithm 을 수행하는 대표적인 방법이다. 그러나 위의 두가지도 또한 여러가지 세부적인 방법들에 의해 수행될 수 있다. 본 논문에서는 첫번째 방법인 quadtree 방법을 이용하였다. 그리고 block 을 나눌때는 PSNR 값을 기준으로 했다. 복원되는 이미지는 기존의 방법과 비교하여 비슷하거나 더 나은 PSNR 값을 갖도록 하였다. 또 경계부분을 작은 block 으로 나누어 줌으로써 blocking 현상을 줄여 보았다.

2.본문

1)fixed-size block matching algorithm

fixed-size block matching 방법은 위에서 언급 했듯이 먼저 이미지를 일정한 크기의 block 으로 나눈다. 그리고 그 block 내의 픽셀들은 같은 움직임을 갖는다고 가정하고 block 의 움직임 벡터를 구하는 방법이다. 이를 수행하는 방법에는 여러가지가 있지만 대략 3 가지를 예를 들면 full search 방법,three-step search 방법,2D logarithm 방법이 있다.

•Full search method

이 방법은 block 을 M × N 으로 나누었을때 이 block 에 가장 잘 matching 되는 이전 프레임을 찾아 가는 방법이다. 이때 이미지 전 영역을 검색 할 수 없으므로 대부분 search window 를 주어 그 영역에서만 estimation 을 수행한다. 즉 움직임 벡터를 (d_1,d_2) 라 하면 이 d_1,d_2 는 $-M_2 \le d_1 \le M_1$ 그리고 $-M_2 \le d_2 \le M_2$ 가 된다. 여기서 M_1, M_2 가 바로 search range 가 된다. 이 방법은 비교적 움직임 벡터를 잘 찾아 내지만 많은 계산량을 요구한다.그래서 이 방법 보다는 three-step search 방법이나 2D logarithm 방법이 많이 쓰인다.

•Three-Step search method

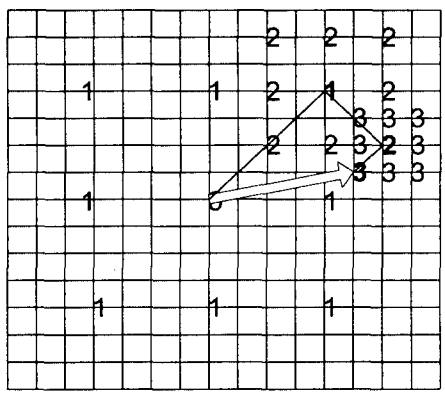


그림 1. Three -step search 방법
Fig. 1.Three-step search method

Three-step search 방법은 3 단계를 거쳐 움직임 벡터를 찾는 방법이다. 그림 1 에서 보듯이 첫번째 단계로 "0"과 "1"로 표시 된 9 개의 점에서 비교하게 된다. 여기서 움직임이 없다면 "0" point 로 아니면 다른 점으로 matching 된다. 그리고 2 번째 단계로 최적의 matching point 에서 다시 크기를 반으로 줄여 다시 첫번째 단계와 같은 계산을 수행한다. 3 번째 단계도 마찬가지로 수행한다.이 3 단계로 움직임 벡터를 찾아 낸다. 물론 최적의 matching 은 가장 적은 error 를 갖는 점이 된다.

•Cross search method

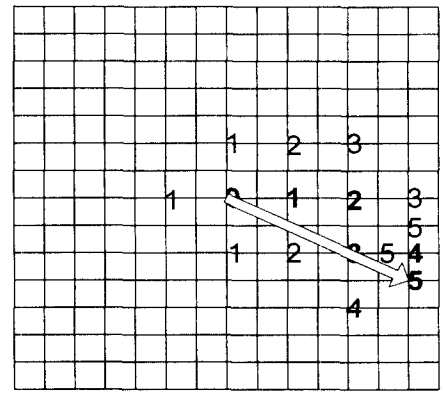


그림 2. Cross search 방법
Fig. 2. Cross search method

Cross search 방법은 그림과 같이 각 단계에서 상하좌우 4 방향의 점을 비교한다. 그리고 그점을 다시 기준으로 4 방향을 기준으로 하여 matching point 를 찾게된다. 이 계산을 몇 단계까지 하느냐 하는것은 Search window 를 미리 정하던지 아니면 몇단계까지 수행할 것인가 하는 것을 정한다.

2)Variable-size block matching algorithm

•Quad-tree 를 이용한 Top-down 방법

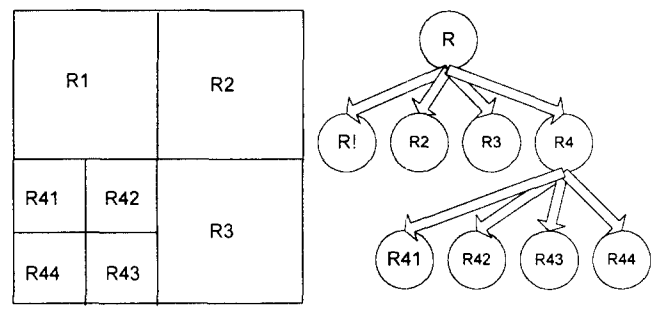


그림 3. 분할된 이미지와 그의 Quadtree
Fig. 3. Patitioned image and corresponding quadtree

이미지 한 프레임을 큰 block 들로 나누고 motion estimation 을 수행한다. 그림과 같이 큰 block 으로 나누어서 estimation 을 수행하고 우리가 threshold 로 준 error 값보다 크게 되면 큰 block 을 4 개의 block 으로 다시 쪼갠다. 즉 $ES_{error} < TH_{error}$ 이면 그 block R_i 를 그대로 취하고 $ES_{error} > TH_{error}$ 이면 R_i 는 R_{ij} 로 4 개의 block 으로 나눈다. 이 과정을 반복해서 수행한다. 더 이상 block 이 쪼개 지지 않을때 까지 하고 그렇지 못할 경우에는 최소 block size 를 정해서 그 block 크기까지만 수행한다.

이렇게 하므로써 움직임이 없거나 같은 움직임을 갖는 영역 들에 대해서는 큰 block 에 대한 움직임 벡터정보만을 가지면 되므로 압축율이 기존의 방법보다 좋아지게 된다. 또 움직임이 많은 부분에 대해서는 작은 block 으로 움직임 벡터를 표현하므로 더 좋은 estimation 이 이루어 질 수 있다.

•bottom-up 방법

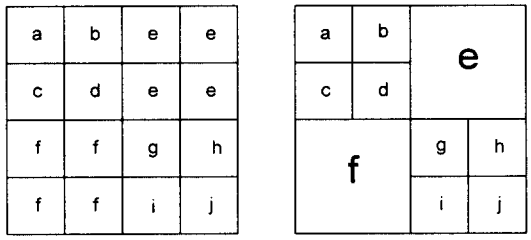


그림 4. 움직임 벡터에 따른 block merging
Fig. 4. Block merging according to motion vector

초기에 이미지를 작은 크기의 block 으로 나누어 estimation 을 수행한다. 그리고 결과로 나온 움직임 벡터를 가지고 같은 움직임 벡터를 갖는 block 들을 merging 해서 큰 bloock 으로 나타낸다.이 큰 블럭에 대해 하나의 움직임 벡터를 표현한다. 여기서도 마찬가지로 4 개의 block 에서 움직임 벡터를 비교할때 비슷한 움직임 벡터를 갖으면 merging 해 주는데 그 임계값은 우리가 정하게 된다. 이렇게 하므로써 움직임 벡터의 정보량을 줄여서 압축을 실행한다. 또 이방법은 이미지 segmentation 에도 적용 될 수 있다

3.실험

PC 모의 실험에서는 기존의 Fixed-block matching 방법과 Variable -block matching 방법을 비교하여 보았다. Fixed-Block matching 방법에서는 block 의 크기를 16 × 16 으로 하였다. 또 Variable-block matching 방법에서는 Quad-tree 방법으로 큰 block 으로 부터 작은 block 으로 4 개씩 쪼개나가는 방법을 이용하였다. 가장 큰 block 의 크기는 64 × 64 로 하였고 또 가장 작은 block 의 크기를 4 × 4 로 하였다. 그리고 그 기준은 복원한 이미지의 block 과 원 이미지 block 과의 에러값으로 하였다. 그리고 각각의 모든 block 에 대해서 search 방법은 마찬가지로 BMA 를 이용하였다. 여기서 에러값은 주로 minimum MSE(mean square error), 또는 minimum MAE(mean absolute error)를 기준으로 한다. 식(1)과 식(2)은 MSE 와 MAE 를 나타낸 식이다.

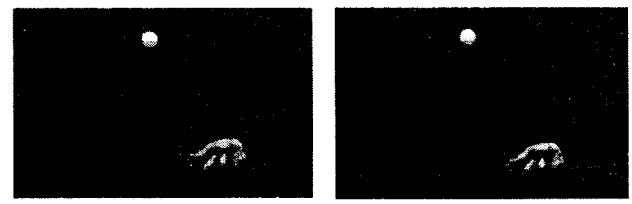
$$MSE = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}[U(m,n)-U_R(m+i,n+j)]}{MN} \quad (1)$$

$$MAE = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[U(m,n)-U_R(m+i,n+j)] \quad (2)$$

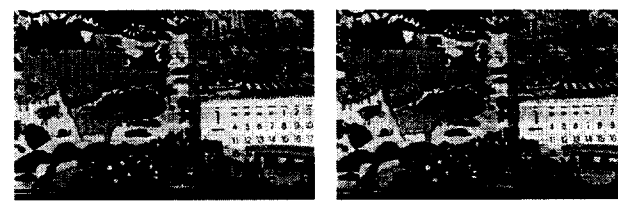
여기서, $U(m,n)$ 은 현재 프레임의 block 이고 $U_R(m,n)$ 은 현재 프레임을 복원한 이전 프레임의 block 이다. 또 block 을 나누어 나갈때 쓴 방법은 PSNR 값을 threshold 값으로 정하였다. PSNR 값은 이미지의 화질 비교에 쓰이는 것으로 식(3)은 PSNR 값을 나타낸다.

$$10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}[U(m,n)-U_R(m+i,n+j)]} \quad (3)$$

테스트 이미지는 "Pingpong"이미지 그리고, "Mobile and calendar" 이미지를 320 × 192 로 잘라서 사용하였다. 다음 그림들은 이전 프레임, 다음 프레임의 이미지이다.



<그림 5>"pingpong" 이미지의 프레임 1 과 프레임 2
Fig 5.frame1 and frame2 of "pingpong"



<그림 6>"mobile and calendar" 이미지의 프레임 1 과 프레임 2
Fig 6. frame1 and frame2 of "mobile and calendar"

1) 16x16 fixed-block matching 방법

three-step 방법을 이용하였다. 위의 그림 7,8 은 각 이미지에 대해 두 프레임으로부터 얻어낸 움직임 벡터와 그로 부터 얻어낸 복원 이미지이다.

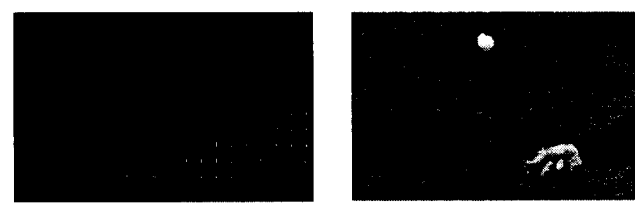


그림 7. "pingpong"의 움직임 벡터와 복원한 이미지
Fig7.motion vector and estimated image of "pingpong"

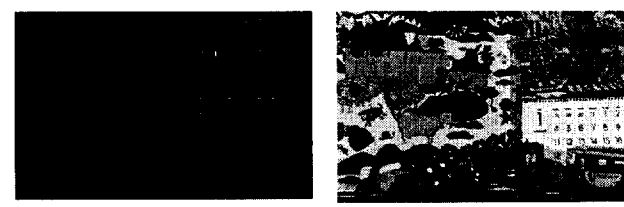


그림 8. "mobile and calendar"의 움직임 벡터와 복원한 이미지지
Fig 8. motion vector and estimated image of "mobile and calendar"

그림 9"pingpong"이미지의 multi-grid block

Fig 9.Multi-grid of "pingpong" image

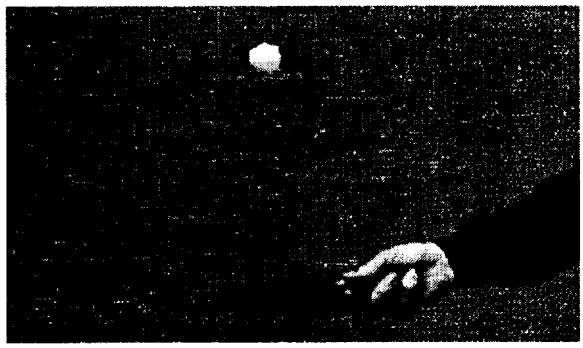


그림 10. 복원된 "pingpong"이미지

Fig 10. estimated image of "pingpong"

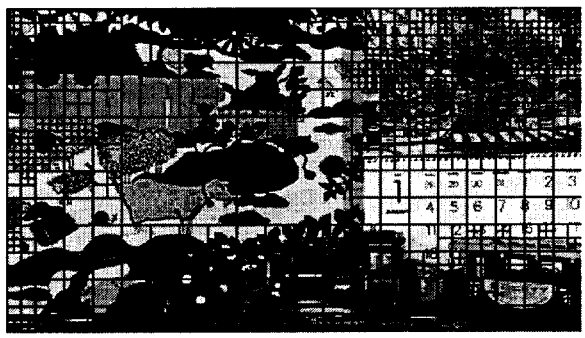


그림 11. "mobile and calendar" 이미지의 multi-grid block

Fig 11. Multi-grid of "mobile and calendar"

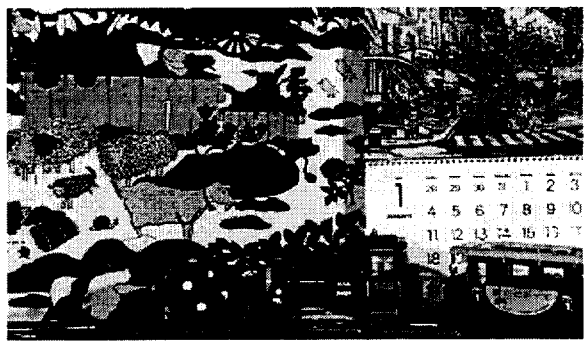


그림 12. 복원된 "mobile and calendar" 이미지

Fig.12 estimated image of "mobile and calendar"

2)Variable-sized block matching 방법

그림 9,10,11,12 는 초기의 block 크기를 64 × 64 로 하고 최소 크기를 4 × 4 로 하여 top-down 방식으로 estimation 한 결과이다. 그에 따른 다양한 block 의 크기와 복원된 이미지를 표현했다.

표 1. PSNR 값의 비교

Table1. PSNR value comparison

PSNR 값	three-step method	top-down
pingpong	30.289817	31.361313
mobile and calendar	23.359824	24.398095

4.결과

본 논문에서는 Variable-block matching 방법으로 움직임 추정을 하고 이를 기존의 Fixed-block matching 방법과 비교해 보았으며 화질 평가기준은 PSNR 값으로 비교하였다. 본논문의 취지는 속도에 중점을 두었으며 각 블럭마다의 움직임 추정 방법으로는 Three-Step BMA 를 이용하여 계산시간을 최소화 하였다. Fixed-block matching 방법은 블럭 크기가 16 × 16 으로 고정되어 경계부분을 잘 찾지못하여 많은 오차가 생기는 반면에 Quadtree 를 이용한 Variable-block matching 방법은 에러가 큰 경계부분을 적은 block 으로 세밀히 조사하여 경계부분을 잘 찾게 된다. 즉 블럭화 현상을 줄여준다. Fixed-block matching 방법에 비해 시간이 조금 더 걸리지만 Full-Search BMA 에 비해 계산량이 훨씬 적어 속도면에서 뛰어나다. 여기서는 최초의 block 을 64 로 하고 최소 크기를 4 로 하였는데 초기 block 을 더 크게 잡고 최소값을 어느정도 크게 하여도 이미지가 잘 복원된다면 이 방법은 더 큰 효과를 가져올 수 있다.

5.참고 문헌

[1] A.MURAT TEKALP, "Digital Video processing", prentice Hall

[2] Gonzalles, "Digital Image Processing". Adison Wesley Publishing company

[3] J.R Jain & A.K.Jain, "Dispacement measurement and its application in interframe image coding",IEEE Trans.Commun., Vol. COM-29,pp. 1799-1808, Dec.1981.

[4] M.H. Chan, Y.B. YU & A.G Constantinides, "Variable size block matching motion compensation with applications to video coding", IEE Proceedings, Vol. 137, Pt. I, No. 4, pp.205-212,Aug.1990.

[5] ANIL K. JAIN, "fundamentals of digital image processing", prentice Hall