論文
5-7-7

경계점의 절대 오차 평균을 이용한 개선된 연속 제거 알고리즘 (An Advanced Successive Elimination Algorithm Using Mean Absolute Difference of Neighboring Search Points)

정수목(Soo-Mok Jung)[1)]

요약

본 논문에서는 경계점의 절대 오차 평균을 이용한 개선된 연속제거알고리즘을 제안하였다. 움직임추정에서 탐색점은 경계점의 절대오차 평균을 이용하여 많은 연산량을 필요로 하는 정합 평가 없이 제거될 수 있다. 적응적 MAD계산 알고리즘을 적용하여 신속하게 탐색점을 제거한다. 따라서 제안된 알고리즘에서는 정합평가 횟수가 감소하게 되고 탐색점제거가 신속하게 발생하여 연산량이 감소하게 된다. 제안된 알고리즘의 효율성을 실험결과로 증명하였다

ABSTRACT

In this paper, an advanced successive elimination algorithm was proposed using mean absolute difference of neighboring search points. By using mean absolute difference of neighboring search points, the search point in motion estimation can be eliminated efficiently without matching evaluation that requires very intensive computations. By using adaptive MAD calculation algorithm, the candidate matching block can be eliminated early. So, the number of matching evaluations of the proposed algorithm can be reduced. The efficiency of the proposed algorithm was verified by experimental results.

Key words: Successive Elimination Algorithm, Motion estimation, Mean Absolute Difference, Motion vector

논문접수 : 2004. 6. 25.
심사완료 : 2004. 7. 20.

1) 정회원 : 삼육대학교 컴퓨터학과 부교수

I. 서론

동영상은 초당 15~30프레임이 연속적으로 상영되기 때문에 데이터량이 매우 크다. 따라서 동영상 데이터를 저장하거나 전송하기 전에 데이터를 압축하는 것이 필요하다.

동영상 압축에서는 두 가지 중복성(redundancy)을 제거함으로 동영상 데이터를 압축한다. 한 프레임 내에는 공간적인 중복성(spatial redundancy)이 존재하고 연속적인 프레임사이에는 시간적인 중복성(temporal redundancy)이 존재한다. 이러한 중복성을 제거하면 동영상 데이터를 효과적으로 압축할 수 있다.

시간적인 중복성을 제거하기 위하여 사용되는 움직임 추정기법은 블록 정합 알고리즘(BMA: block matching algorithm)과 화소 순환 알고리즘(PRA: pel-recursive algorithm)으로 나누어진다. 두 기법 중 블록 정합알고리즘의 구현이 간단하기 때문에 블록 정합 알고리즘이 CCITT H.261[1], ITU-T H.263[2] 그리고 MPEG[3] 등 여러 비디오 코딩 표준(video coding standards)으로 널리 채택되어 왔다.

블록 정합 알고리즘에서는 데이터를 압축하고자하는 현재 프레임을 임의의 작은 블록으로 나눈 뒤, 각 블록에 대하여 이전 프레임에 설정된 탐색영역에서 정합 척도가 최적인 블록을 찾아 두 블록간의 변위에 해당하는 움직임 벡터를 찾는다.

블록 정합 알고리즘 중 전역 탐색 알고리즘(FSA: Full Search Algorithm)은 이전 프레임상에 정의된 탐색 윈도우 내의 모든 정합 블록 후보(candidate matching block)들 중에서 최적 정합 블록을 찾음으로 최적 움직임 벡터(global optimum motion vector)를 찾지만, 전역 탐색을 수행하기 위하여 매우 많은 연산량을 필요로 하기 때문에 실제적인 응용에서는 사용되지 않는다.

전역탐색알고리즘의 연산량을 감소시키기 위하여 3단계탐색[4], 2-D 로그탐색[5], 직교탐색[6], 일차원전역탐색(one-dimensional full search)[7] 등 여러 알고리즘들이 개발되어 왔다. 이러한 고속 블록 정합 알고리즘들은 움직임 보상된 잔여 에러 면(motion compensated residual error surface)이 움직임 벡터의 변위에 대한 볼록 함수(convex function)라는 가정에 기초하고 있다[8]. 그러나 일반적으로 동영상에서는 이러한 가정은 실제적이지 않다[9]. 그러므로 이러한 고속 블록 정합 알고리즘들을 사용하여 얻어진 움직임 벡터는 근본적으로 국소적 최적치(local optimum)가 된다.

Li와 Salari에 의해서 제안된 연속 제거 알고리즘(SEA: successive elimination algorithm)[10]은 이러한 볼록성 가정(convexity assumption) 없이 전역 탐색 알고리즘의 연산량을 감소시켰다.

본 논문에서는 경계점의 절대 오차 평균[11]을 이용하는 기법과 본 저자가 [12]에서 제안한 기법을 연속제거알고리즘에 적용하여 연속제거알고리즘의 연산량을 감소시키는 고속 알고리즘을 제안하였다.

II. 연속 제거 알고리즘

전역탐색 알고리즘의 매우 큰 연산량을 감소시키기 위하여 제안된 알고리즘들([4]~[7])은 움직임 보상된 잔여 에러 면이 움직임 벡터의 변위에 대한 볼록 함수를 가정하고 탐색윈도우 내에서 조사하는 탐색 점의 개수를 제한함으로 탐색의 복잡도를 줄이는 방법들이다. 따라서 이러한 알고리즘들을 사용하면 FSA에서 구해지는 최적 움직임 벡터가 아닌 최적 움직임 벡터와 근사한 움직임 벡터 값을 구하게 된다.

이러한 알고리즘들의 문제점을 해결하기 위하여 움직임 추정의 정확도는 FSA와 동일하나 FSA보다 매우 빠른 움직임 벡터 추정 알고리즘을 Li 와 Salari[10]가 제안하였다.

SEA에서는 현재 프레임내의 템플릿 블록의 움직임 벡터를 이전 프레임상의 동일한 위치에 대응하는 블록과의 변위 (0,0)을 초기 움직임 벡터로 한다. SEA는 블록의 sum norm을 사용하여 탐색윈도우 내의 불필요한 탐색점을 제거한다.

크기가 NxN인 블록 B의 sum norm은 식 (1)과 같이 정의되고 $B(i,j)$는 블록 B의 (i,j)번째 픽셀의 그레이 값을 나타낸다.

$$S_B = \sum_{i=1}^{N} \sum_{j=1}^{N} |B(i,j)| \quad (1)$$

S_T를 템플릿 블록 T의 sum norm으로, S_X를 정합 블록 후보 X의 sum norm으로, MAD_{min}은 탐색 과정 중에 현재 최소 MAD(current minimal MAD)로 둔다. $MAD(T,X)$를 T와 X 사이의 MAD로 두면 $MAD(T,X)$는 식 (2)와 같이 정의된다. $T(i,j)$와 $X(i,j)$는 각각 T와 X의 (i,j)번째 픽셀의 그레이 값이다.

$$MAD(T,X) = \sum_{i=1}^{N} \sum_{j=1}^{N} |T(i,j)-X(i,j)| \quad (2)$$

Li와 Salari는 [10]에서 식 (3)이 성립함을 증명하였다.

$$MAD(T,X) \geq MAD\_SB = |S_T - S_X| \quad (3)$$

SEA에서는 식 (3)에 기초하여 $|S_T-S_X| \geq MAD_{min}$을 가지는 정합 후보 블록 X를 제거할 수 있어 탐색 시간을 절약할 수 있다.

연속 제거 알고리즘의 절차는 아래와 같다

step1 이전 프레임 상에 있는 탐색윈도우 내에서 초기 탐색점 선택.

step2 초기 탐색점에서의 MAD 계산. 계산된 MAD를 현재 최소 MAD로 함. (MAD_{min_curr}=MAD)

step3 while(all the search points are tested in the search window?){

step4 탐색윈도우내에서 탐색점 선택

step5 선택된 탐색점에서 MAD 계산

step6 if(MAD_{min_curr}> MAD)
MAD_{min_curr}=MAD;

step7 }

step8 the minimum MAD= MAD_{min_curr}

step9 최소 MAD를 가지는 탐색점에서 움직임 벡터(MV)계산

step10 the optimum motion vector =MV

Ⅲ. 경계점의 Mean Absolute Difference를 이용한 개선된 연속제거 알고리즘

움직임 추정과정에서는 움직임 벡터를 구하기 위하여 최적의 정합블록을 찾아야 한다. 최적의 정합블록은 step5~6에서와 같은 블록정합과정을 통하여 찾는데, 이러한 블록정합에는 매우 많은 연산량이 소요된다.

따라서 블록 정합을 가능한 적게 수행하여 움직임 벡터를 찾을 수 있다면 연산량의 감소를 얻을 수 있다. 연속 제거 알고리즘의 step 5~6의 연산량을 줄이기 위하여 저자가 제안한 [12]에서와 같이 템플릿 블록과 정합후보블록을 4개의 서브블록으로 분할하고 각 서브블록에서 8개의 탐색점을 표본 추출한 후 표본 추출된 탐색점들에 대하여 각 서브블록에 대한 부분 MAD를 구한다. 구해진 4개의 서브블록의 부분MAD를 크기순대로 정렬한다. 정렬된 서브블록의 순서에 따라 step 5~6을 수행하면 계산되는 MAD값이 신속하게 증가하게 되어 step5에서 탐색점이 신속하게 제거된다.

또한 탐색윈도우내에서 선택된 탐색점에 대하여 최적의 정합블록을 찾기 위하여 연산량이 매우 큰 블록 정합(step 5~6)을 수행하기 전에 이웃 경계에 있는 탐색점의 MAD로부터 현재 탐색점에서의 MAD 최소값을 예측하여 예

측된 값이 현재까지의 MAD 최소값인 MAD_{min_curr} 보다 크다면 블록 정합을 수행하지 않고 탐색점을 제거할 수 있어 연산량을 감소시킬 수 있다.

현재 프레임상에 있는 NxN 크기를 갖는 움직임 벡터를 찾아 압축을 수행하고자 하는 템플릿(template) 블록 $\boldsymbol{C}$와 움직임벡터 (x, y)를 가지는 탐색윈도우내의 정합 후보 블록(candidate matching block) $\boldsymbol{M}(x,y)$ 는 식 (4), (5)와 같이 표현된다.

$$\boldsymbol{C}=[\boldsymbol{C}_0 \quad \boldsymbol{C}_1, \ldots\ldots \boldsymbol{C}_{N-1}] \tag{4}$$

$$\boldsymbol{M}(x,y)=[\boldsymbol{M}_0(x,y)\ \boldsymbol{M}_1(x,y)\ldots\ldots\boldsymbol{M}_{N-1}(x,y)] \tag{5}$$

이때 $\boldsymbol{C}_i$와 $\boldsymbol{M}_i(x,y)$는 아래와 같다.

$\boldsymbol{C}_i=[C_{0,i}\ C_{1,i}\ \ldots\ldots\ C_{N-1,i}]^T$,

$\boldsymbol{M}_i(x,y)=[m_{y,x+i}\ m_{y+1,x+i}\ \ldots\ldots\ldots m_{y+N-1,x+i}]^T$

현재 프레임상에서 움직임벡터를 구하여 압축하려고 하는 블록인 템프릿 블록과 정합 후보블록간의 오차블록 $\boldsymbol{D}(x,y)$와 탐색점 (x,y)에서 MAD는 식 (6), (7)과 같이 표현된다.

$$\boldsymbol{D}(x,y)=\boldsymbol{M}(x,y)-\boldsymbol{C}= [\boldsymbol{M}_0(x,y)-\boldsymbol{C}_0\ \boldsymbol{M}_1(x,y)-\boldsymbol{C}_1\ldots\ldots \boldsymbol{M}_{N-1}(x,y)-\boldsymbol{C}_{N-1}] \tag{6}$$

$$\mathrm{MAD}(x,y)=\|\boldsymbol{M}(x,y)-\boldsymbol{C}\| = \|\boldsymbol{D}(x,y)\| \tag{7}$$

(x,y)위치에 있는 정합 후보블록의 인접위치 $(x+1,y)$ 탐색점에서의 오차블록 $\mathrm{D}(x+1,y)$은 식 (8)과 같다.

$$\boldsymbol{D}(x+1,y)=\boldsymbol{M}(x+1,y)-\boldsymbol{C}=[\boldsymbol{M}_1(x,y)-\boldsymbol{C}_0\ \boldsymbol{M}_2(x,y)-\boldsymbol{C}_1\ldots\ldots\ \boldsymbol{M}_N(x,y)-\boldsymbol{C}_{N-1}] \tag{8}$$

현재 프레임상의 템플릿 블록 내의 화소들의 이웃 화소간의 수평방향으로의 화소값의 차 $\boldsymbol{C}_{DH}$를 템플릿 블록 $\boldsymbol{C}$로 표현하면 식 (9)와 같이 된다.

$$\boldsymbol{C}_{DH}=[\boldsymbol{C}_0-\boldsymbol{C}_{N-1}\quad \boldsymbol{C}_1-\boldsymbol{C}_0\ \ldots\ldots\ \boldsymbol{C}_{N-1}-\boldsymbol{C}_{N-2}] \tag{9}$$

식(6)의 오차블록 $\boldsymbol{D}(x,y)$을 변형하여 새로운 오차블록 $\boldsymbol{D}_{H+}(x,y)$는 식(10)과 같이 표현될 수 있다.

$$\boldsymbol{D}_{H+}(x,y)=[\boldsymbol{M}_N(x,y)-\boldsymbol{C}_0\ \boldsymbol{M}_1(x,y)-\boldsymbol{C}_1\ldots\ldots \boldsymbol{M}_{N-1}(x,y)-\boldsymbol{C}_{N-1}] \tag{10}$$

그러므로 탐색점 $(x+1,y)$에서의 MAD는 식(11)과 같이 구해지고 식 (12)의 삼각 부등식(triangular inequality)을 이용하여 탐색윈도우내의 탐색점 $(x+1,y)$에서의 MAD는 식(13)와 같이 표현되어 질 수 있음이 [11]에서 증명되었다.

$$\mathrm{MAD}(x+1,y)=\|\boldsymbol{D}_{H+}(x,y)+\boldsymbol{C}_{DH}\| \tag{11}$$

$$|\ \|\boldsymbol{A}_1\|-\|\boldsymbol{A}_2\|\ |\le\|\boldsymbol{A}_1+\boldsymbol{A}_2\| \le \|\boldsymbol{A}_1\|+\|\boldsymbol{A}_1\| \tag{12}$$

$$|\ \|\boldsymbol{D}_{H+}(x,y)\| - \|\boldsymbol{C}_{DH}\|\ | \le \mathrm{MAD}(x+1,y) \le\|\boldsymbol{D}_{H+}(x,y)\| + \|\boldsymbol{C}_{DH}\| \tag{13}$$

이 식에서 MAD의 최소값을 구하는 연산량은 MAD를 실제 구하는 연산량보다 훨씬 적다. 이는 $\|\boldsymbol{C}_{DH}\|$를 전체 탐색영역에 대하여 한번만 계산하면 되고 $\|\boldsymbol{D}_{H+}(x,y)\|$는 N개의 열 중에서 첫 번째 열을 대치시키는 과정이 필요하기 때문이다.

따라서 MAD의 최소값을 구하지 않고 보다 적은 연산량을 필요로 하는 MAD의 최소값을 예측함으로 최적정합블록을 찾기 위하여 정합 후보 블록에서 예측된 MAD최소값이 현재까지의 MAD최소값보다 클 때,

블록정합과정을 생략할 수 있다.

연속 제거알고리즘의 성능을 개선하기 위하여 위에서 기술한 두 가지 기법을 연속제거알고리즘에 적용한 제안된 알고리즘의 절차는 연속제거알고리즘의 step4~5를 아래와 같이 변형하면 된다.

step4 탐색윈도우내에서 탐색점 선택

step4.1 경계점의 MAD값을 이용하여 최소 MAD값을 예측

step4.2 if(예측된 최소MAD값 ≥ MAD_{min_curr}) goto 7

step4.3 선택된 탐색점을 좌측상단으로 하는 정합 후보블록을 4개의 서브블록으로 분할 후 각 8개의 탐색점을 표본 추출

step4.4 각 서브블록의 8개 표본 추출된

탐색점에 대하여 부분MAD 계산 후 이를 정렬

step5 정렬된 서브블록 순으로 MAD를 계산

Ⅳ. 실험결과

본 연구의 실험에서는 176x144 pixel 크기를 가지는 stefan.qcif와 foreman.qcif의 50 프레임(frame)을 사용하였다. 움직임 벡터를 구하는 블록의 크기는 16x16 pixel로 하였고 탐색 윈도우의 크기는 31x31 pixel이고 움직임 벡터는 정수값만을 고려하였다. 이러한 실험환경은 연속 제거 알고리즘[10]에서의 실험환경과 동일하다.

표1에서 (in row)는 16x16 블록의 1행에 대한 sum norm을 구하는데 필요한 연산량을 기본 단위로 표시하였음을 나타낸다.

연속제거 알고리즘과 제안된 알고리즘에 대하여 나선형 탐색(spiral search)기법을 사용하여 탐색의 효율을 증가시켰다.

제안된 알고리즘을 각 비디오 클립에 대하여 실험한 결과는 표1과 같다.

표1. 실험결과

video	알고리즘	평균정합 회수/fr.	평균row/ 정합평가	오버헤드 (in rows)	연산량	연산량 감소율
stefan	SEA	35,110.3	6.34	8,671.5	231,270.8	
	Proposed	32,135.1	5.48	12,613.2	188,713.5	18.4%
foreman	SEA	19,293.8	5.98	6,053.4	121,430.3	
	Proposed	18,136.5	5.20	11,274.9	105,584.7	13.1%

삼각부등식을 이용하여 경계점에서의 평균 절대 오차를 이용하여 현재의 탐색점의 최소 MAD를 정확하게 예측하기 때문에 표1에서 보는 바와 같이 제안한 알고리즘은 연속 제거 알고리즘과 같이 움직임 추정 정확도가 100%로 유지된다.

제안된 움직임 추정 알고리즘을 stefan.qcif, foreman.qcif에 적용하였을 때, 연속 제거 알고리즘의 연산량이 최대 18.4% 감소하였다.

Ⅴ. 결론

본 논문에서는 경계점의 평균 절대치 오차의 값과 적응적인 MAD계산 알고리즘을 이용하여 연속제거알고리즘의 연산량을 감소시키는 효율적인 움직임 추정기법을 제안하였다.

제안된 알고리즘은 연속 제거 알고리즘에서 사용한 삼각부등식을 경계점의 평균 절대치 오차에 적용하여 현재 탐색점의 평균절대치 오차의 최소값을 예측하고 이를 바탕으로 계산 집중적인 연산인 블록정합을 수행해야하는 탐색점들을 효과적으로 제거하였기 때문에 움직임 추정의 정확도는 100%유지되면서 움직임 추정에 필요한 연산량을 효과적으로 감소시키고, 적응적인 MAD계산알고리즘을 사용하여 정합후보 블록이 신속하게 제거되어 연산량을 감소시키는 알고리즘이다

stefan.qcif와 foreman.qcif에 대하여 제안된 알고리즘을 적용하였을 때 움직임 추정의 정확도는 100% 유지되었고 MSEA의 연산량이 최대 18.4% 감소하였다.

제안된 알고리즘은 low bit-rate와 뛰어난 품질의 코딩을 필요로 하는 비디오 코딩 애플리케이션에 매우 효율적인 움직임 추정 알고리즘이다.

References

[1] CCITT Standard H.261, "Video codec for audiovisual services at px64 kbit/s," ITU, 1990.

[2] ITU-T DRAFT Standard H.263, "Video coding for narrow telecommunication channel at (below) 64kbit/s," ITU, Apr. 1995.

[3] ISO-IEC JTC1/SC2/WG11, "Preliminary

text for MPEG video coding standard," ISO, Aug. 1990.

[4] T. Koga, K. Iinuma, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," Proc. Nat. Telecommunications Conf., Nov. 1981, pp. G5.3.1- G.5.3.5

[5] J. R. Jain, and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Commun., Dec. 1981, vol. COMM-29, pp. 1799-1808

[6] A. Puri, H.-M. Hang, and D. L. Schilling, "An efficient block matching algorithm for motion compensated coding," Proc. Int. Conf. Acoust., Speech, Signal Processing, 1987, pp. 25.4.1-25.4.4

[7] M. J. Chen, L. G. Chen and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," IEEE Trans. Circuits Syst. Video Technol., Oct. 1994, vol. 4, pp. 504-509

[8] B. Liu, and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," IEEE Trans. Circuits Syst. Video Technol., Apr. 1993, vol. 3, no. 2, pp. 148-157

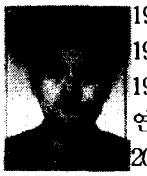
[9] K. H. K. Chow and M. L. Liou, "Genetic motion search algorithm for video compression," IEEE Trans. Circuits Syst. Video Technol., Dec. 1993, vol. 3, pp. 440-446

[10] W. Li, and E. Salari, "Successive Elimination Algorithm for Motion Estimation," IEEE Trans. Image Processing, Jan. 1995, vol. 4, No.1, pp. 105-107

[11] W.S. Cheong, B.K. Lee etc., "A fast Block Matching Algorithm Using Mean Absolute Error of Neighbor Search Point and Search Region Reduction," The Journal of Korean Institute Communication Science, Vol. 25, No.1B, Jan. 2000.

[12] S. M. Jung, S. C. Shin, H. Baik and M. S. Park, "Efficient multilvel successive elimination algorithms for block matching motion estimation," IEE Proc.-Vis. Image Signal Process., Vol. 149, No.2, April 2002.

정수목



1984 경북대학교 전자공학과(학사)
1986 경북대대학원 전자공학과(석사)
1986~1991 LG 정보 통신 연구소 연구원
2002 고려대대학원 컴퓨터학과(박사)
1998~현재 삼육대학교 컴퓨터학과 부교수
관심분야: 멀티미디어, 컴퓨터시스템