논문-00-5-2-11

움직임 벡터의 시공간 상관성을 이용한 새로운 고속 블럭 정합 움직임 추정 방식

남 재 열*, 서 재 수*, 곽 진 석**, 이 명 호**, 송 근 원 ***

New Fast Block-Matching Motion Estimation using Temporal and Spatial Correlation of Motion Vectors

Jae Yeal Nam*, Jae Soo Seo*, Jin Suk Kwak**, Myoung Ho Lee** and Kun-Woen Song***

요 약

본 논문은 움직임 벡터의 높은 시·공간 상관도 정보를 이용하여 계산량을 줄이면서 움직임 추정의 정확도를 높일 수 있는 새로운 블럭 정합 움직임 추정 방식을 제안한다. 제안된 방식은 기존의 고속 움직임 추정 방식들이 이용하는 탐색 영역내의 일관된 첫 번째 탐색 위치에서 움직임 벡터를 찾는 것이 아니라 움직임 벡터의 높은 시·공간 상관도 정보를 이용하여 보다 정확한 탐색 영역을 찾아 탐색 영역을 보정함으로써 보다 정확한 첫 번째 탐색 위치를 중심으로 움직임 벡터를 탐색한다. 즉, 본 논문에서 제안하는 방식의 핵심은 움직임 추정의 정확도를 높이기 위해서 보다 정확한 첫 번째 탐색 위치를 찾는 것이다. 따라서 움직임 벡터의 시간적인 상관성을 이용하기 위해서 현재 프레임 블럭과 같은 좌표를 갖는 이전 프레임 블럭의 방향성을 조사한다. 또한 공간적인 상관성을 이용하기 위해서 현재 프레임내의 이웃 블럭들의 방향성을 조사한다. 이러한 블럭들이 갖는 방향성을 바탕으로 움직임 추정을 위한 첫 번째 탐색 위치를 결정하게 되고 그 위치를 중심으로 일정한 탐색 패턴에 따라 움직임 벡터를 탐색하는 방식이다. 실험 결과 제안된 방식은 기존의 대표적인 고속 탐색 방식들에 비해 PSNR (Peak-to-Signal Noise Ratio) 값에 있어서 평균적으로 1.7dB 개선되고 영상에 따라 최고 3.6dB 정도 우수한 결과를 나타낸다. 또한 탐색 횟수에서는 기존의 대표적인 고속 탐색 알고리즘인 3단계 탐색 알고리즘 (Three-step search algorithm) 보다 평균 50% 이상을 줄일 수 있었고, 정확한 움직임 벡터를 찾는 비교에 있어서도 월등히 우수한 결과를 나타내었다. 또한 제안된 방식은 정량적인 결과뿐만 아니라 부호화후 복호화한 영상의 화질에 있어서도 다른 고속 탐색 알고리즘 보다 월등히 우수한 화질을 제공한다.

Abstract

This paper introduces a new technique that reduces the search times and improves the accuracy of motion estimation using high temporal and spatial correlation of motion vector. Instead of using the fixed first search point of previously proposed search algorithms, the proposed method finds more accurate first search point as to compensating searching area using high temporal and spatial correlation of motion vector. Therefore, the main idea of proposed method is to find first search point to improve the performance of motion estimation and reduce the search times. The proposed method utilizes the direction of the same coordinate block of the previous frame compared with a block of the current frame to use temporal correlation and the direction of the adjacent blocks of the current frame to use spatial correlation. Based on these directions, we compute the first search point. We search the motion vector in the middle of computed first search point with two fixed search patterns. Using that idea, an efficient adaptive predicted direction search algorithm (APDSA) for block matching motion estimation is proposed. In the experimental results show that the PSNR values are improved up to the 3.6dB as depend on the image sequences and advanced about 1.7dB on an average. The results of the comparison show that the performance of the proposed APDSA algorithm is better than those of other fast search algorithms whether the image sequence contains fast or slow motion, and is similar to the performance of the FS (Full Search) algorithm. Simulation results also show that the performance of the APDSA scheme gives better subjective picture quality than the other fast search algorithms and is closer to that of the FS algorithm.

I. 서 론

오늘날 전세계적으로 사용자가 폭발적으로 증가하고 있는 인터넷 환경에서 단순한 문자 서비스만으로는 사용자의 다양한 시각적 욕구를 충족시키지 못함으로 인해서 동영상 서비스에 대한 요구가 증가되고 있다. 그러나 기존의 네트워크 속도로는 대용량의 동영상 전송에는 많은 문제점이 있기 때문에 데이터의 용량을 줄일 수 있는 많은 비디오 압축 국제 표준방식들이 (H.261, H.263, MPEG-1, MPEG-2, MPEG-4) 개발되었다.[1][2][3][4][5] 특히, 차세대 이동 통신 기술로 각광 받고 있는 IMT-2000용 단말기를 통해 동영상을 전송하고자 할 때 핵심기법으로 MPEG-4 비디오 객체 지향 부호화 기법이 이용 될 수 있을 것이다. 그러나 현재의 MPEG-4 비디오 압축에서 사용되는 블럭 정합 움직임 추정 방식은 동영상 부호화를 위한 전체 계산량의 약 80%에 해당되는 많은 계산량을 필요로 한다. 따라서 저 전력 고 효율을 필요로 하는 IMT-2000용 단말기의 VLSI 칩 구현을 위해서는 획기적으로 계산량을 줄이면서 전역 탐색과 비슷한 성능을 나타낼 수 있는 새로운 움직임 추정 방식이 요구되고 있다.[6][7]

따라서 본 논문에서는 움직임 벡터의 높은 시·공간적인 상관성과 가운데 중심적인 특성(Center-biased property)을 이용함으로써 동영상 압축을 위해서 획기적으로 계산량을 줄이면서도 움직임 추정의 정확성을 높임으로써 예측 성능을 향상시킬 수 있는 새로운 블럭 정합 움직임 추정 알고리즘을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 제 Ⅱ절에서는 기존의 제시된 고속 탐색 알고리즘의 특성과 문제점을 제시한다. 제 Ⅲ절에서는 본 논문에서 이용하는 움직임 벡터의 정보를 이용하여 블록의 방향성을 계산하는 과정을 간단히 설명한다. 제 Ⅳ절에서는 본 논문에서 제안하는 적응적 예측 방향성 탐색 알고리즘을 설명하고, 제 Ⅴ절에서는 PSNR, MSE(Mean Square Error), 탐색 횟수, 전역 탐색 알고리즘의 블럭과 비교하여 같은 좌표의 움직임 벡터를 갖는 블럭의 개수 등과 같은 평가기준을 통해서 다른 탐색 알고리즘과 성능을 비교 분석하였다. 제 Ⅵ절에서는 결론을 내린다.

* 계명대학교 컴퓨터·전자 공학부
School of Computer and Electronics Engineering, Keimyung University
** 한국 전자 통신 연구원 무선·방송 기술 연구소
Broadcasting Technology Department, Radio & Broadcasting Technology Laboratory, Electronics and Telecommunications Research Institute
*** 위덕대학교 전자공학과
Department of Electronics Engineering, Uiduk University

Ⅱ. 기존의 고속 탐색 알고리즘

대표적인 블럭 정합 움직임 추정 기법으로는 전역 탐색 알고리즘 (Full Search Algorithm - FSA)이 있다. 전역 탐색 알고리즘은 최상의 움직임 벡터를 찾기 위해서 현재 프레임 블럭과 같은 좌표를 갖는 이전 프레임 블럭의 위치를 중심으로 탐색 영역에 포함되는 모든 점들을 조사하여 움직임 벡터를 찾는다. 탐색 범위를 ±w라 할 때, 탐색 영역에 포함되는 모든 점들을 탐색 할 경우 $(2w+1)^2$ 개의 블럭 정합 탐색 횟수를 갖게 되고 최대 ±15화소의 탐색 영역을 갖는다면 961번의 탐색을 하게 된다. 따라서 전역 탐색 알고리즘은 움직임 추정 기법들 중에서 최적의 움직임 벡터를 찾는 장점이 있는 반면에 탐색 영역에 포함되는 모든 점들을 탐색하기 때문에 다른 고속 움직임 추정 기법들에 비해서 계산량이 너무 많다는 단점이 있다. 따라서, 전역 탐색 기법의 단점인 과도한 계산량을 줄일 수 있는 고속 탐색 알고리즘들이 많이 연구되었다.[8~18]

이러한 고속 탐색 알고리즘들 중에서 대표적인 방식은 3단계 탐색 알고리즘 (Three-Step Search Algorithm - TSS)이다[8]. 3단계 탐색 알고리즘은 일정한 패턴에 따라서 27개점들에 대해서 탐색을 수행하기 때문에 계산량을 줄일 수 있는 가장 간단하면서도 효율적인 알고리즘이다. 그러나 3단계 탐색 알고리즘의 경우 모든 영상들에 대해서 일정한 탐색 패턴을 적용하기 때문에 움직임이 거의 없는 영상의 경우 문제가 없지만 움직임이 많은 영상의 경우 첫 번째 탐색이 잘못 되었을 경우 local optima에 빠질 수 있는 단점이 존재한다. 따라서 이러한 단점을 보완하기 위해서 4단계 탐색 알고리즘 (Four-Step Search Algorithm - FSS)이 나오게 되었고[9], 이 방식은 3단계 탐색 알고리즘 보다 첫 번째 탐색 범위를 줄임으로서 local optima에 빠지는 단점을 보완하려고 하였다. 그러나 이 방식 또한 모든 영상에 대해서 일정한 패턴을 적용함으로써 움직임이 많은 영상의 경우 그 효율이 떨어짐을 볼 수 있다. 따라서, 움직임 벡터의 가운데 중심적인 특성을 이용한 다이아몬드 탐색 알고리즘 (Unrestricted Center-Biased Diamond Search Algorithm - UCBDS)이 제안되었다[10]. 이 알고리즘은 대부분의 움직임 벡터가 ±3화소 이내에 포함된다는 특성을 이용함으로써 움직임 벡터를 찾는 확률을 높이려는 방식이다. 그러나 움직임 벡터가 가운데 중심으로 ±3화소 이내에 분포되어 있는 영상의 경우에 적합하지만 급격한 움직임이 있는 영상의 경우에는 움직임 벡터가 ±3화소를 넘어 가는 경우가 많기 때

문에 압축 성능이 떨어짐으로 움직임이 많은 영상의 경우에는 적합하지 않다.

새로이 제안되는 움직임 추정 알고리즘의 특성을 살펴본다면 이전까지의 고속 탐색 알고리즘은 단순히 현재 블럭의 움직임 벡터를 찾기 위해서 이전 프레임의 특정한 범위에 포함되는 점들에 대해서 탐색을 하게 되어 이전에 탐색된 움직임 벡터의 정보를 이용하지 못하였다. 따라서 이전에 탐색된 움직임 벡터들의 정보를 이용함으로써 기존의 고속 탐색 알고리즘을 개선한 방식이 예측 탐색 알고리즘 (Prediction Search Algorithm - PSA) 이다[11]. 이 방식은 인접한 이전 블럭들이 갖는 움직임 벡터의 정보를 이용함으로써 보다 정확한 움직임 벡터를 찾으려고 하였다. 그러나 이 방식은 인접한 블럭의 움직임 벡터의 연관성이 떨어질 경우 압축 성능이 현저히 떨어진다는 단점이 있다. 또한 시간적인 연관성을 이용하기 위해서 이전 프레임 블럭들의 움직임 벡터의 정보를 이용하는 방식인 예측 방향성 탐색 알고리즘(Predicted Direction Search Algorithm- PDSA)이 있다[12]. 이러한 방식은 기존의 고속 탐색 알고리즘보다 많은 성능의 향상을 가져오지만 첫 번째 탐색위치를 고정된 패턴에 따라서 이동하기 때문에 보다 정확한 움직임 벡터를 탐색 할 수 있는 가능성을 줄이는 결과를 가져온다.

결국, 모든 고속 탐색 알고리즘들은 계산량을 줄이기 위해서 탐색 영역에 포함되는 특정한 몇몇 점들만 조사하여 움직임 벡터를 찾기 때문에 국부적인 탐색을 하게 된다. 즉, 현재의 탐색 단계에서 몇몇의 점들이 가장 작은 SAD(Sum Absolute Differences) 값을 갖는 점과 비슷한 SAD 값을 갖게 되면 다음 탐색 단계의 탐색 방향이 정확하지 않게 되어 국부적인 탐색을 하게 되고[13], 결국에는 압축 영상의 성능이 저하되는 단점이 있다. 따라서 본 논문에서는 움직임 벡터의 가운데 중심적인 특성, 연속된 프레임에서 움직임 벡터의 시·공간적인 상관성 등을 이용하여 첫 번째 탐색 위치를 보다 정확하게 예측하고 첫 번째 탐색 위치를 보정함으로써 다른 고속 탐색 알고리즘보다 압축 성능을 향상시키면서도 부족한 정보로 인해서 야기되는 국부적인 탐색을 방지할 수 있는 적응적 예측 방향성 탐색 알고리즘(Adaptive Predicted Direction Search Algorithm-APDSA)을 제안한다.

Ⅲ. 블럭의 방향성 계산

본 논문은 이전의 예측 방향성 탐색 알고리즘에서 제안된 9개의 방향성을 이용한다[12]. 따라서 움직임 벡터를 이용하여 블록이 갖는 방향성을 계산하는 과정을 간단히 살펴본다면 다음과 같다. 우선 움직임 벡터를 이용하여 움직임 벡터가 갖는 각도를 계산한다.

$$Radian = asin\left(\frac{|y|}{\sqrt{(x^2+y^2)}}\right)$$
$$MVAngle = Radian \times 180 \quad (1)$$

asin은 arcsine 함수이고, x, y는 움직임 벡터의 변위이다. MVAngle은 움직임 벡터가 1사분면에 위치 할 때 움직임 벡터의 각도이다.

계산된 1사분면의 각도를 이용하여 움직임 벡터가 갖는 최종적인 각도를 계산하기 위해서는 움직임 벡터의 위치, 즉 몇 사분면인가를 고려하여야 한다. 최종적인 각도를 계산하는 과정은 표 1 에 정의되어 있다.

표 1. 움직임 벡터의 위치를 고려한 최종적인 각도 계산
Table 1. The computation of final angle considering the aspects of motion vector

움직임 벡터의 위치	움직임 벡터의 최종 각도
1사 분면	FMVAngle = MVAngle
2사 분면	FMVAngle = 180 - MVAgle
3사 분면	FMVAngle = 180 + MVAgle
4사 분면	FMVAngle = 360 - MVAgle

표 1에서 계산된 움직임 벡터의 최종적인 각도를 바탕으로 결정되는 블럭들의 방향은 표 2에 정의되어 있다.

표 2. 움직임 벡터의 각도와 블럭의 방향과의 관계
Table 2. The relationship between the final angle of motion vector and the direction of a block

움직임 벡터의 최종 각도 (*FMVAangle*)	블럭의 방향
FMVAngle = 0°	0 방향
0° < *FMVAngle* < 22.5° and 337.5° ≤ *FMVAngle* < 360°	① 방향
22.5° ≤ *FMVAngle* < 67.5°	② 방향
67..5° ≤ *FMVAngle* < 112.5°	③ 방향
112.5° ≤ *FMVAngle* < 157.5°	④ 방향
157.5° ≤ *FMVAngle* < 202.5°	⑤ 방향
202.5° ≤ *FMVAngle* < 247.5°	⑥ 방향
247.5° ≤ *FMVAngle* < 292.5°	⑦ 방향
292.5° ≤ *FMVAngle* < 337.5°	⑧ 방향

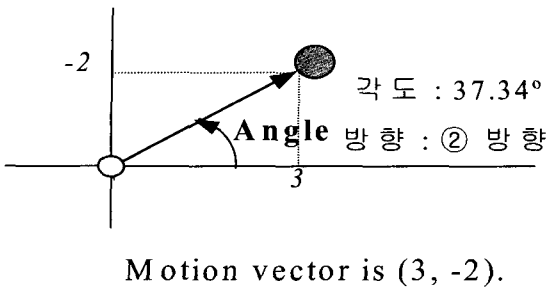


그림 1. (3, -2) 값을 갖는 움직임 벡터의 각도 및 블럭의 방향
Fig. 1. The angle of motion vector and direction of a block in case of motion vector is (3, -2)

결국 움직임 벡터가 갖는 정보를 이용하여 표 2에서 정의한 9개의 방향성을 본 논문에서 이용한다. 이러한 블럭의 방향성을 계산하는 간단한 예를 살펴본다면 그림 1과 같이 움직임 벡터가 (3, -2)의 값을 갖을 경우 식 (1)에 의해서 움직임 벡터가 갖는 각도는 37.34° 이고 블럭의 방향은 표 2에 따라 ②방향이 된다.

Ⅳ. 적응적 예측 방향성 탐색 알고리즘

본 논문에서 제안한 알고리즘은 움직임 벡터의 높은 시·공간적 상관성을 이용하기 위해서 앞 절에서 설명한 방식에 따라 각 블럭들의 방향성을 계산하고, 계산된 방향성을 바탕으로 블럭들에 새로운 보정된 벡터 값을 할당한다. 이러한 보정된 벡터 값을 이용하여 블럭 정합 움직임 추정을 위한 보다 정확한 첫 번째 탐색 위치를 결정하게 되고, 이 첫 번째 탐색 위치를 중심으로 현재 블럭의 움직임 벡터를 탐색하는 방식이다.

따라서 움직임 벡터의 시간적인 상관성을 이용하기 위해서는 그림 2와 같이 현재 프레임 블럭과 같은 좌표를 갖는 이전 프레임 블럭(블럭 P)의 방향성을 이용하게 되고, 공간적인 상관성을 이용하기 위해서는 그림 2와 같이 현재 프레임내의 이웃한 블럭들(블럭 1, 2, 3)의 방향성을 이용하게 된다.

그림 2와 같이 현재 프레임 블럭과 같은 좌표를 갖는 이전 프레임 블럭과 현재 프레임 내의 이웃한 블럭들이 갖는 움직임 벡터의 정보를 이용하여 각각의 블럭들에 대해서 앞 절에서 정의한 방향성을 계산하게 되고, 계산된 방향을 바탕으로 표 3과 같이 보정된 벡터 값을 할당하게 된다. 첫 번째 탐색 위치를 결정하는 과정에서 이전 블럭들이 갖는 움직임 벡터의 정보를 이용하여 계산된 블럭들의 방향에 따라 보정된 벡터 값은 (MV 값) UCBDS에서 제시된것과 같이 움직임 벡터가 ±3화소 이내에 80% 이상이 포함된다는 점을 이용하여 본 논문에서는 ±3화소 이내에 포함되는 점들을 조사하였을 경우 2화소 이동하였을 경우 가장 정확한 값이 된다. 즉, 움직임 벡터의 가운데 중심적인 특성과 기존에 제시된 고속 탐색알고리즘을 바탕으로[10] 본 논문에서의 실험 결과에 따라 가장 좋은 결과를 가져오는 2를 할당하였다. 블럭의 방향에 따라 새로이 보정된 벡터 값은 아래 표 3에 나타나 있다.

표 3. 방향에 따른 블럭의 보정된 벡터 값
Table 3. The offset vector values of block as to the direction

방향	*MV(Vx, Vy)*
0 방 향	(0 , 0)
① 방 향	(2 , 0)
② 방 향	(2 , -2)
③ 방 향	(0 , -2)
④ 방 향	(-2 , -2)
⑤ 방 향	(-2 , 0)
⑥ 방 향	(-2 , 2)
⑦ 방 향	(0 , 2)
⑧ 방 향	(2 , 2)

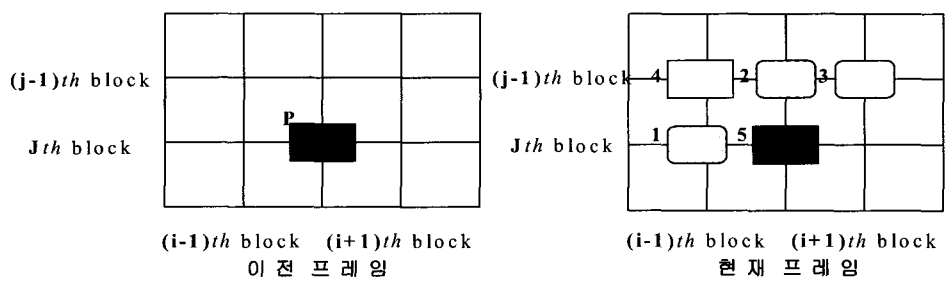


그림 2. 현재 블럭 5의 움직임 벡터를 찾기위해 이용되는 이전 프레임 블럭(P)과 현재 블럭 내의 이웃한 블럭들(블럭 1, 2, 3)
Fig. 2. The block (P) of the previous frame and the blocks (1, 2, 3) of the current frame which used to search motion vector of the current block(5)

결국 최종적으로 첫 번째 탐색 위치를 결정하기 위해서 보정된 벡터 값에 각각의 블럭들이 갖는 경중에 따라 가중치 값을 곱하게 되고 이렇게 계산된 위치로 현재 블럭의 움직임 추정을 위한 첫 번째 탐색 위치를 보정하게 된다. 첫 번째 탐색 위치를 결정하기 위한 계산 과정은 식 (2)와 같다.

$$W=[\ W_1,\ W_2,\ W_3,\ W_4]^T=\left[\frac{\alpha}{\beta},\frac{\gamma}{\beta},\frac{\gamma}{\beta},\frac{\gamma}{\beta}\right]^T \quad (2)$$

$$P=[\ P_x,\ P_y]^T=\left[\ \sum_{k=1}^{4} W_k\ V_{xk},\ \sum_{k=1}^{4} W_k\ V_{yk}\right]^T$$

여기서, W는 가중치 값이고 α ,β ,γ 는 상수이다. V_{xk}는 표 3에서 MV의 V_x 값이고, V_{yk}는 표 3에서 MV의 V_y 값이다. 즉, 블럭의 방향에 따른 새로운 보정된 벡터 값이다. P에서 P_x는 첫 번째 탐색 위치를 위해서 X 방향으로 이동하는 좌표이고, P_y는 Y방향으로 이동하는 좌표다. 그리고 상수 k값에서 1은 시간적 특성을 이용하는 현재 프레임 블럭과 같은 좌표를 갖는 이전 프레임 블럭이고, 2, 3, 4는 공간적 특성을 이용하는 현재 프레임내의 이웃한 블럭들을 나타낸다.

따라서 블럭들이 갖는 움직임 벡터의 높은 시·공간적 상관성과 가운데 중심적인 특성을 이용하여 본 논문에서 제안된 방식의 첫 번째 탐색 위치를 결정하는 간단한 예를 살펴본다면 표 4와 같다.

현재 블럭의 움직임 벡터를 찾기 위해서 첫 번째 탐색 위치를 결정하기 위해 이용하는 이전 프레임 블럭(블럭 P)과 현재 프레임 내의 이웃한 블럭들(블럭 1, 2, 3)이 갖는 움직임 벡터의 값과 이러한 움직임 벡터를 통해서 계산되는 각도, 방향, 보정된 벡터 값은 표 4에 나타나 있다.

따라서 움직임 벡터가 갖는 정보를 이용하여 표 4와 같이 움직임 벡터의 각도, 블록의 방향, 그리고 새로운 보정된 벡터 값들을 얻게 되고, 표 4에서 보정된 벡터 값을, $MV(V_x, V_y)$, 바탕으로 계산되는 첫 번째 탐색 위치는 다음과 같다.

$$W=[\ W_1,\ W_2,\ W_3,\ W_4]^T=\left[\frac{2}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5}\right]^T$$

$$P=[\ P_x,\ P_y]^T=\left[\ \sum_{k=1}^{4} W_k\ V_{xk},\ \sum_{k=1}^{4} W_k\ V_{yk}\right]^T$$

$$P_x=\left(\frac{2}{5}\times -2\right)+\left(\frac{1}{5}\times 0\right)+\left(\frac{1}{5}\times -2\right)+\left(\frac{1}{5}\times -2\right)$$
$$=(-1.6)=(-1)$$

$$P_y=\left(\frac{2}{5}\times -2\right)+\left(\frac{1}{5}\times -2\right)+\left(\frac{1}{5}\times -2\right)+\left(\frac{1}{5}\times -2\right)$$
$$=(-2)$$

따라서 제시한 예는 현재 블럭의 움직임 벡터를 찾기 위해서 첫 번째 탐색 위치를 현재 블럭의 좌측 상단의 위치에서 X 방향으로 -1을 이동하고 Y 방향으로 -2 만큼 이동하게 된다.

이렇게 이동한 첫 번째 탐색 위치(-1, -2)를 중심으로 그림 3과 같은 두 가지 탐색 패턴을 갖고 그림 4와 같이 움직임 벡터를 탐색하게 된다.

제안된 APDSA 알고리즘을 요약하면 다음과 같다.

단계 1 이전 블럭들이 갖는 움직임 벡터를 이용하여 각도를 계산한다. 계산된 각도를 이용하여 블럭의 방향을 결정하고 각각의 방향을 통해서 표 3과 같이 블럭들에 새로운 보정 벡터 값을 할당한다.

단계 2 식 (2)를 이용하여 계산된 첫 번째 탐색 위치를 3×3 탐색 영역의 중심점으로 하는 9개의 탐색 점들을 갖고 탐색을 시작한다. 만약 9개의 점들 가운데 가장 작은 SAD를 갖는 점이 이전 단계의 탐색에서와 같은 점을 갖는다면, 즉 가장 작은 SAD를 갖는 점이 탐색 영역의 가운데 점인 (α , α)라면 단계 3으로 이동, 그렇지 않으면 다음 과정을 반복한다.

a) 만약 이전의 탐색 과정에서 가장 작은 SAD 갖는 점이 수평·수직 방향이라면 [즉, (α −1, α −1), (α +1, α −1), (α −1, α +1), 또는 (α +1, α +1)], 그림 3의 (a)와 같은 탐색 패턴을 갖고 새로운 3개의 탐색 점을 추가하여 가장 작은 SAD를 갖는 점을 가운데 점으로 하여 움직임 벡터를 추정하게 된다.

표 4. 움직임 벡터에 따른 각도, 방향, 보정된 벡터 값
Table 4. The angle, direction, and offset vector values as to motion vector of blocks

	블 럭	움직임 벡터	각 도	방 향	$MV(V_x, V_y)$
이전 프레임	P	(-3, -2)	142.75°	④ 방향	(-2, -2)
현재 프레임	1	(0 , -3)	90.00°	③ 방향	(0 , -2)
	2	(-4, -3)	143.15°	④ 방향	(-2, -2)
	3	(-2, -2)	145.00°	④ 방향	(-2, -2)

b) 만약 이전의 탐색 과정에서 가장 작은 SAD 갖는 점이 대각 방향이라면 [즉, $(\alpha-1, \alpha)$, $(\alpha+1, \alpha)$, $(\alpha, \alpha-1)$, 또는 $(\alpha, \alpha+1)$], 그림 3의 (b)와 같은 탐색 패턴을 갖고 새로운 5개의 탐색 점을 추가하여 가장 작은 SAD를 갖는 점을 가운데 점으로 하여 움직임 벡터를 추정하게 된다.
단 ±7의 탐색 영역을 벗어나는 모든 점들은 무시한다. 매 단계마다 가장 작은 SAD를 갖는 점은 재정의 되고, 가장 작은 SAD를 갖는 점이 이전 단계에서 가장 작은 SAD를 갖는 점과 같다면 단계 3으로 이동, 그렇지 않으면 단계 2를 반복 수행한다.

단 계 3 탐색을 멈추고 최종적으로 원하는 움직임 벡터를 찾게 된다

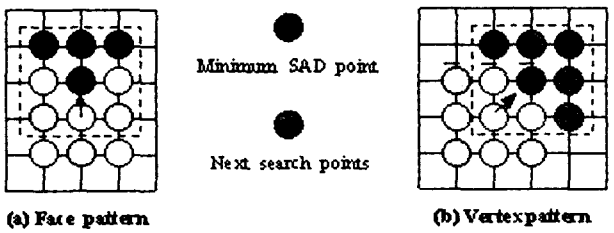


그림 3. 제안한 알고리즘의 두 가지 탐색 패턴
(a) 움직임 벡터가 가장 작은 SAD를 면에서 갖을 경우 탐색 과정 (b) 움직임 벡터가 가장 작은 SAD를 꼭지점에서 갖을 경우 탐색 과정
Fig. 3. Two searching patterns of the APDSA
(a) searching pattern in case of the motion vector has minimum SAD on the face point, (b) searching pattern in cae of the motion vector minimum SAD on the vertex point

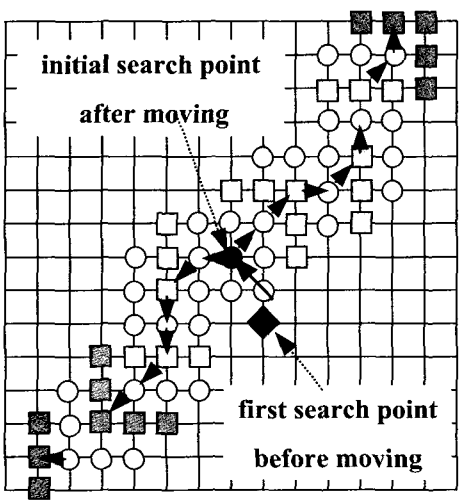


그림 4. 보정된 첫 번째 탐색 위치가 (-1, -2)인 경우 탐색 과정
Fig. 4. Searching porcess in case of offset vector values (-1, -2)

V. 실험 결과 및 분석

본 논문에서 제안한 APDSA 알고리즘의 성능을 평가하기 위해서 표 5와 같은 실험 환경을 이용하였다.

표 5. 실험 환경
Table 5. The simulation environments

Development software	Visual C++ 6.0
Coding scheme	MPEG-4 Video Coding Scheme
Frame size	QCIF(176 × 144), CIF (352 × 288)
Block size	16 × 16 pels
Coding bit rates	64kbps, 384kbps
Frame rate	10 frames/sec.
Number of frames	100frames

또한 제안된 알고리즘의 성능을 정량적으로 비교하기 위해서 다양한 기준들이 있지만[19] 본 논문에서는 다음과 같은 4개의 평가기준을 사용하였다.

1) 평균 PSNR 값
2) 평균 MSE (Mean-Square-Error) 값
3) 각 블럭에 대한 탐색 횟수
4) 전역 탐색과 비교하여 각 블럭이 같은 움직임 벡터의 좌표를 갖는 개수

본 논문에서 사용한 PSNR 값은 식 (3)을 이용하여 구하였다.

$$PSNR = 20 \log_{10}\left(\frac{255}{\mathrm{RMSE}}\right) \; dB$$
$$\mathrm{RMSE} = \sqrt{\frac{1}{XY}\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1}[f(i,j)-f'(i,j)]^2} \tag{3}$$

여기서, i, j는 블럭에서 x, y 좌표 값을 나타내고, $f(i,j)$는 i, j 좌표에서 원영상 블럭이 갖는 화소 값을 나타내고, $f(i,j)$는 i, j 좌표에서 비교 영상 블럭이 갖는 화소 값을 나타낸다.

기존의 고속 탐색 알고리즘과 제안한 방식의 성능을 비교함에 있어서 384kbps에서 CIF 영상을 실험한 결과는 표 6에서 표 9까지 나타내었고, 64kbps에서 QCIF 영상을 실험한 결과는 표 10에서 표 13까지 나타내었다. 기존의 고속 탐색 알고리즘과 제안한 알고리즘의 평균 PSNR 값을 비교하는 결과는 표 6과 표 10에 나타나 있다. 표 7과 표 11은 각 블럭에 대한 평균 MSE 값이다. 각 블럭에 대한

표 6. 재구성한 영상의 평균 PSNR 값 (CIF image)
Table 6. Average PSNR values of reconstructed video sequences (CIF image)

Image / ME Method	Sequence Image							
	Coastguard	Foreman	Mother	News	Stefan	Table	Average	Percentage
FSA (±15)	31.279	36.801	41.623	41.471	29.676	34.107	35.826	100%
TSS	29.200	32.917	41.547	41.346	25.636	30.703	33.558	93.67%
FSS	29.039	32.915	41.547	41.371	25.697	30.842	33.568	93.70%
UCBDS	28.620	32.695	41.558	41.414	25.543	30.797	33.438	93.33%
PSA	29.197	33.017	41.561	41.430	25.750	31.190	33.691	94.04%
PDSA	30.932	36.423	41.604	41.433	27.342	33.303	35.173	98.18%
APDSA	31.110	36.450	41.617	41.453	28.066	33.503	35.367	98.72%

표 7. 재구성한 영상의 평균 MSR 값 (CIF image)
Table 7. Average MSE values of reconstructed video sequences (CIF image)

Image / ME Method	Sequence Image							
	Coastguard	Foreman	Mother	News	Stefan	Table	Average	Percentage
FSA (±15)	1306.38	681.45	273.97	189.54	2569.98	958.85	830.03	100%
TSS	2361.07	1291.00	339.92	262.01	4022.45	2241.41	1752.08	211.19%
FSS	2296.60	1285.09	339.76	261.76	4038.71	2136.71	1726.44	208.00%
UCBDS	2261.28	1452.47	338.93	254.00	4047.73	2212.41	1761.19	212.18%
PSA	2397.02	1321.51	340.56	266.86	4104.91	2278.41	1784.88	215.04%
PDSA	1541.66	794.29	295.40	207.22	2602.57	1334.48	1127.60	135.85%
APDSA	1417.38	731.47	284.33	198.32	2481.42	1157.20	1045.02	125.90%

표 8. 재구성한 영상의 각 블럭에 대한 평균 탐색 횟수 (CIF image)
Table 8. Average search points per a block. (CIF image)

Image / ME Method	Sequence Image							
	Coastguard	Foreman	Mother	News	Stefan	Table	Average	Percentage
FSA (±15)	961	961	961	961	961	961	961	100%
TSS	27	27	27	27	27	27	27	2.81%
FSS	27.21	27.23	27.01	27.00	27.45	27.55	27.24	2.83%
UCBDS	14.08	14.62	13.17	13.15	14.70	16.55	14.38	1.50%
PSA	27.53	27.48	27.05	27.04	27.54	27.37	27.34	2.74%
PDSA	16.74	22.35	23.59	23.79	21.11	20.99	21.43	2.23%
APDSA	13.30	13.82	9.69	9.42	13.95	15.34	12.59	1.31%

표 9. 재구성한 영상이 전역 탐색과 비교해서 같은 움직임 벡터를 갖는 블럭의 개수 (CIF image)
Table 9. Average number of the matched block with the same distributions of the motion vector compared to that of the FS algorithm (CIF image)

Image / ME Method	Sequence Image							
	Coastguard	Foreman	Mother	News	Stefan	Table	Average	Percentage
FSA (±15)	396	396	396	396	396	396	396	100%
TSS	39.15	139.17	353.52	364.63	153.35	102.13	191.99	48.48%
FSS	39.44	139.44	353.58	364.66	153.55	108.46	193.19	48.79%
UCBDS	39.54	135.84	352.33	364.28	153.57	106.48	192.01	48.49%
PSA	35.49	141.92	353.64	364.80	153.21	106.40	192.58	48.63%
PDSA	341.14	326.53	383.26	388.90	259.65	200.84	316.72	79.98%
APDSA	326.76	350.78	394.20	394.24	278.25	221.99	327.70	82.75%

표 10. 재구성한 영상의 평균 PSNR 값 (QCIF image)
Table 10. Average PSNR values of reconstructed video sequences (QCIF image)

Image / ME Method	Sequence Image							
	Akiyo	Carphone	Coastguard	Foreman	Stefan	Salesman	Average	Percentage
FSA (±15)	41.657	40.322	38.703	39.443	36.795	39.654	39.429	100%
TSS	41.643	40.203	36.855	39.318	32.669	39.642	38.388	97.36%
FSS	41.643	40.201	36.874	39.324	32.740	39.645	38.388	97.36%
UCBDS	41.656	40.257	37.198	39.370	32.672	39.644	38.466	97.56%
PSA	41.643	40.148	36.879	39.332	32.471	39.648	38.353	97.27%
APDSA	41.657	40.309	38.645	39.432	35.462	39.649	39.192	99.40%

표 11. 재구성한 영상의 평균 MSR 값 (QCIF image)
Table 11. Average MSE values of reconstructed video sequences (QCIF image)

Image / ME Method	Sequence Image							
	Akiyo	Carphone	Coastguard	Foreman	Stefan	Salesman	Average	Percentage
FSA (±15)	144.65	531.00	981.84	737.48	1598.02	186.46	696.58	100%
TSS	152.61	704.66	1567.11	1110.20	3590.98	202.43	1232.22	176.90%
FSS	152.61	702.70	1588.73	1107.33	3592.53	202.03	1224.32	175.76%
UCBDS	148.50	674.62	1521.54	1102.38	3589.61	200.08	1206.12	173.15%
PSA	152.47	721.84	1684.70	1099.76	3578.68	202.42	1239.98	178.01%
APDSA	144.65	536.31	999.68	754.32	1888.21	187.36	751.76	107.91%

표 12. 재구성한 영상의 각 블럭에 대한 평균 탐색 횟수 (QCIF image)
Table 12. Average search points per a block (QCIF image)

Image / ME Method	Sequence Image							
	Akiyo	Carphone	Coastguard	Foreman	Stefan	Salesman	Average	Percentage
FSA (±15)	255	255	255	255	255	2552	255	100%
TSS	27	27	27	27	27	27	27	1.06%
FSS	26.90	26.94	26.93	26.97	27.73	26.91	27.06	1.06%
UCBDS	14.01	14.52	14.58	14.25	14.53	14.04	14.31	0.56%
PSA	27.00	27.01	26.72	27.07	27.07	27	26.98	1.06%
APDSA	9.03	10.63	11.37	11.63	12.09	9.10	10.63	0.42%

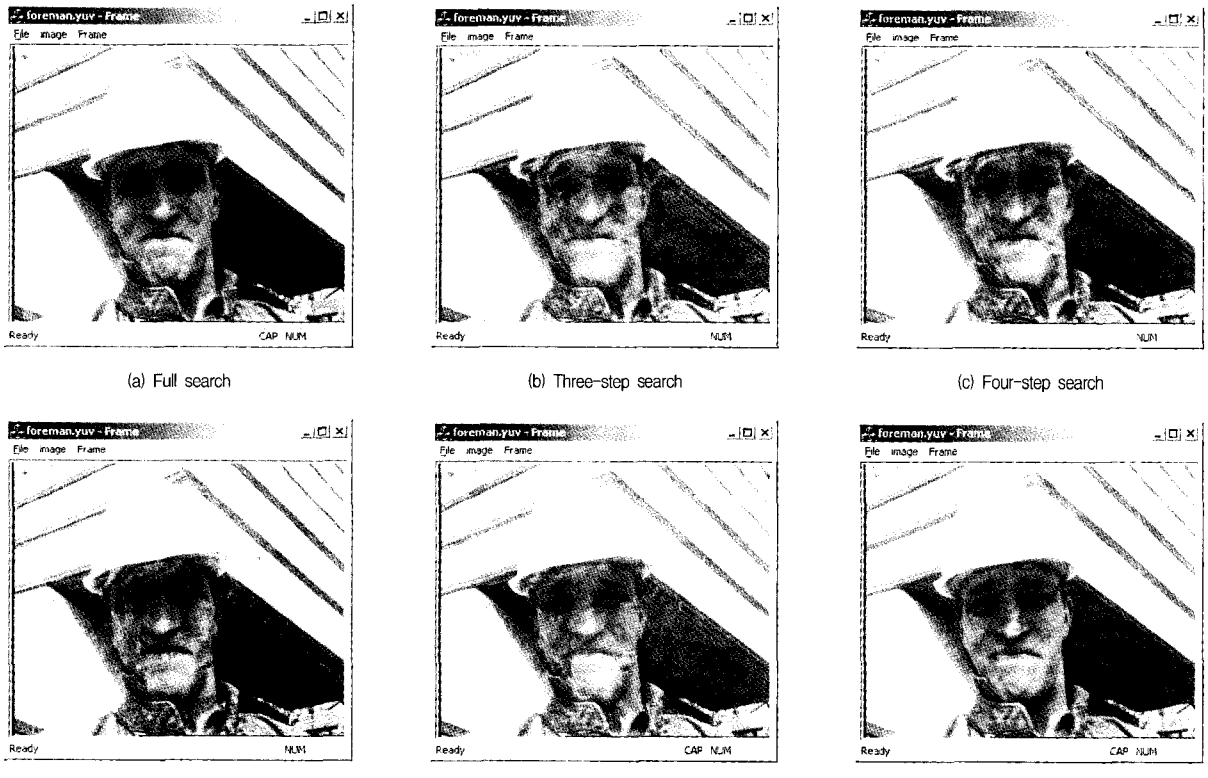
표 13. 재구성한 영상이 전역 탐색과 비교해서 같은 움직임 벡터를 갖는 블럭의 개수 (QCIF image)
Table 13. Average number of the matched block with the same distributions of the motion vector compared to that of the FS algorithm (QCIF image)

Image / ME Method	Sequence Image							
	Akiyo	Carphone	Coastguard	Foreman	Stefan	Salesman	Average	Percentage
FSA (±15)	99	99	99	99	99	99	99	100%
TSS	98.08	69.30	31.18	56.99	40.96	97.00	65.59	66.25%
FSS	98.08	69.33	31.95	57.04	40.96	97.00	65.73	66.39%
UCBDS	98.08	69.91	30.36	56.04	41.41	97.00	65.47	66.13%
PSA	98.08	69.11	31.93	56.97	28.70	97.00	63.63	64.27%
APDSA	99.00	95.24	87.00	91.64	82.51	98.88	92.38	93.31%

평균 탐색 횟수는 표 8과 표 12에 나타내었고 전역 탐색과 비교하여 각 블럭마다 같은 움직임 벡터를 갖는 개수를 나타내는 결과는 표 9와 13에 나타내었다.

이러한 실험 결과를 살펴본다면 본 논문에서 제안한 방식은 기존의 고속 탐색 알고리즘보다 모든 면에서 우수함을 볼 수 있다. 특히, PSNR 값에서는 기존의 고속 탐색 알고리즘 보다 평균적으로 1.7dB 정도 높게 나타나고, 급격한 움직임이 있는 영상(Foreman 영상)의 경우 최고 3.6dB정도의 우수함을 나타내고 있다. 정확한 움직임 벡터를 찾는 비교에 있어서도 영상의 종류에 상관없이 다른 고속 탐색 알고리즘보다 월등히 우수함을 볼 수 있었다. 또한, 영상의 시간적인 특성을 이용하는 PDSA(Predicted Direction Search Algorithm)보다 PSNR 값에 있어서는 유사하거나 영상에 따라 약간의 성능 향상을 가져오지만, 탐색 횟수에 있어서는 40% 가량을 줄일 수 있어 계산 속도를 획기적으로 개선하였다. 따라서 저 전력 고 효율을 필요로 하는 IMT-2000 용 단말기를 위해 적합한 방식으로 채택될 수 있을 것이다.

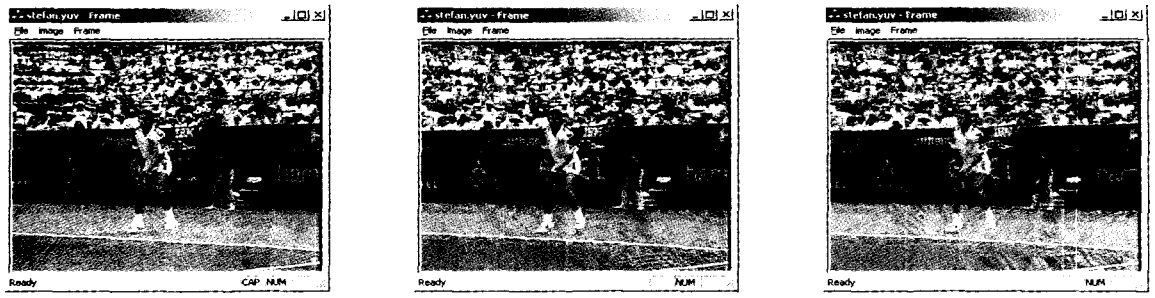
결국 이전 블럭들이 갖는 움직임 벡터들의 정보를 이용함으로써 현재 블럭의 움직임 벡터를 추정하기 위해서 첫 번째 탐색 위치를 보다 정확하게 보정함으로써 기존에 존재하는 고속 탐색 알고리즘 보다 정확한 움직임 벡터의 탐색을 가능하게 하였다. 따라서 본 논문에서 제안한 방식은 이전 블럭들의 움직임을 조사함으로써 현재 프레임 블럭의 움직임이 어느 방향에서 이동하였는가를 예측하고 그 방향으로 첫 번째 탐색 위치를 보정함으로써 움직임 추정의 성능을 높이려는 방식이다. 결론적으로 제안한 알고리즘의 경우 영상의 움직임 벡터의 높은 시·공간 상관성과 가운데 중심적인 특성을 이용함으로써 실험 결과에서 정량적으로 모든 면에서 우수함을 나타낼 뿐만 아니라 영상의 부호화후 복호화한 영상의 화질에 있어서도 그림 6, 7, 8에 나타나는 것과 같이 우수함을 볼 수 있었다.



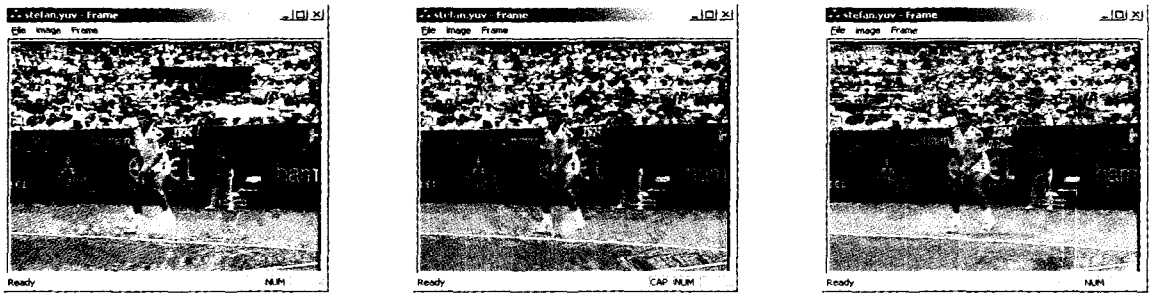
(a) Full search (b) Three-step search (c) Four-step search

(d) UCBDS (e) PSA (f) New proposed APDSA

그림 6. 재구성한 Forman 39번째 프레임 이미지
Fig. 6. The reconstructed Foreman image of 39th frame

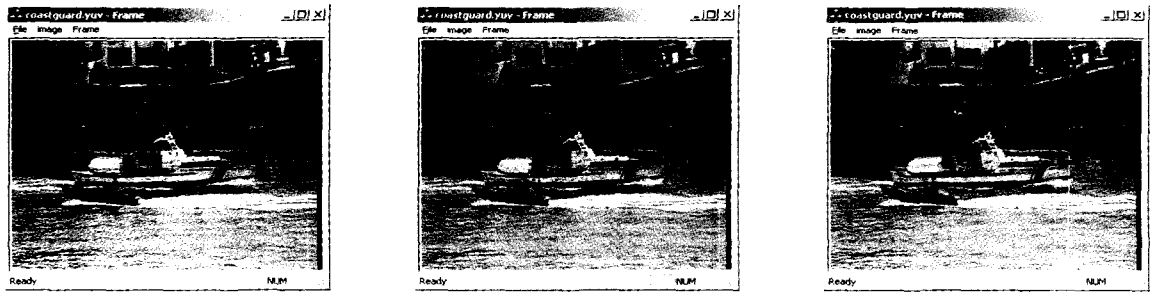


(a) Full search (b) Three-step search (c) Four-step search

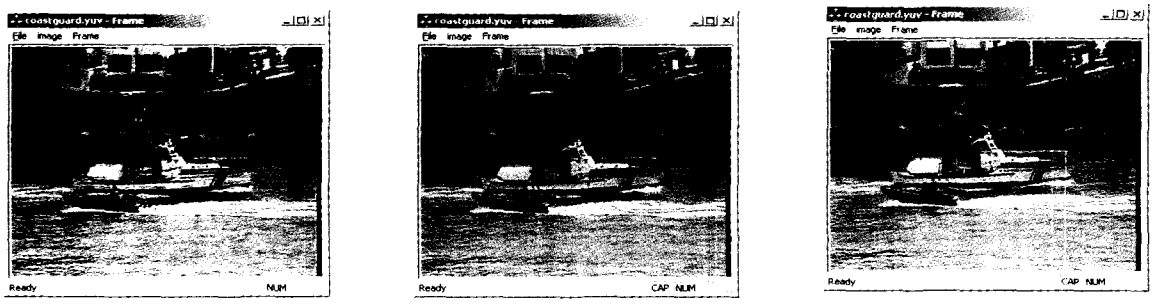


(d) UCBDS (e) PSA (f) New proposed APDSA

그림 7. 재구성한 Stefan 69번째 프레임 이미지
Fig 7. The reconstructed Stefan image of 69th frame



(a) Full search (b) Three-step search (c) Four-step search



(d) UCBDS (e) PSA (f) New proposed APDSA

그림 8. 재구성한 Coastugard 81번째 프레임 이미지
Fig 8. The reconstructed Coastguard image of 81th frame

Ⅵ. 결 론

본 논문에서 제안한 APDSA방식은 현재 프레임과 같은 좌표를 갖는 이전 프레임 블럭의 움직임 벡터 사이의 높은 상관성과 현재 프레임 내에서 이전 블럭과의 상관성, 즉 연속된 영상의 움직임 벡터는 시·공간적으로 높은 상관성을 갖는다는 점을 이용하여 현재 프레임 블럭의 움직임 추정을 위해서 첫 번째 탐색 위치를 보다 정확하게 예측하고 보정함으로써 실험 평가 기준들 중 PSNR, MSE, 탐색 횟수, FS 알고리즘과 비교하여 움직임 벡터가 같은 좌표를 갖는 블럭의 개수 등과 같은 모든면에 있어서 다른 고속 탐색 알고리즘들보다 월등히 우수한 결과를 나타내고 있다. 즉, 위의 표에 나타난 것과 같이 PSNR 값에 있어서 평균 1.7dB의 성능의 향상을 가져왔고, 평균 탐색 횟수에 있어서도 약 50% 이상을 줄였으며 전역 탐색과 비교하여 같은 움직임 벡터를 갖는 블럭의 개수 즉, 정확한 움직임 벡터를 찾을 확률에서 월등한 성능 향상을 가져왔다. 이러한 결과를 종합해 본다면 본 논문에서 제안한 알고리즘은 다른 고속 탐색 알고리즘보다 예측 성능과 그 이외의 부가적인 면에서 월등히 좋은 결과를 나타냄을 볼 수 있었다.

따라서 본 논문에서 제안한 방식은 기존의 대표적인 고속 탐색 알고리즘들이 고정되거나 부정확한 첫 번째 탐색 위치에서 움직임 벡터를 탐색함으로 인해서 예측 성능을 떨어뜨리는 결과를 가져온 단점을 보완하기 위해서 움직임 벡터의 높은 시·공간 상관성과 가운데 중심적인 특성을 이용하여 보다 정확한 첫 번째 탐색 위치를 찾아서 보정함으로써 이전의 방식들 보다 정확한 움직임 추정을 가능하게 하였다. 이러한 결과를 종합해 본다면 개발된 방식은 기존의 고속 탐색 알고리즘 보다 많은 계산량을 줄임으로써 저 전력에 적합하고, 압축 효율에 있어서 전역 탐색과 비슷한 결과를 가져옴으로 고 효율을 이룰 수 있었다. 따라서 본 논문에서 제안한 고속 블럭 정합 움직임 추정 알고리즘은 차세대 기술로 각광 받고 있고 또한 향후 서비스될 IMT-2000용 단말기 등에 사용될 수 있을 것이다.

참 고 문 헌

[1] International Telecommunication Union, "Video Codec for Audiovisual Services at p×64 Kbits," ITU-T Recommendation H.261, Mar. 1993.

[2] International Telecommunication Union, "Video Codec for Audiovisual Services at p×64 Kbits," ITU-T Draft H.263, May 1996.

[3] ISO/IEC JTC/SC29/WG11, "ISO/IEC CD N11172: Information Technology," *MPEG-1 Committee Draft*, Dec. 1991.

[4] ISO/IEC JTC/SC29/WG11, "ISO/IEC CD N13818: Information," *MPEG-2 Committee Draft*, Dec. 1993.

[5] ISO/IEC JTC1/SC29/WG11, "MPEG-4 Video Verification Model Version 8.0," *MPEG97/N1796, Stockholm*, July 1997.

[6] Eric Chan and Sethuranman Panchanathan, "Motion Estimation Architecture for Video Compression," *IEEE Trans. Consumer Electron.*, vol. 39, no. 3, pp. 292-297, Aug. 1993.

[7] Viet L. Do and Kenneth Y. Yun, "A Low-power VLSI Architecture for Full-Search Block-Matching motion estimation," *IEEE Trans. on CSVT*, vol. 8, no. 4, pp. 393-398, Aug. 1998.

[8] R. Li, B. Zeng and M. L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation," *IEEE Trans. on CSVT*, vol. 4, no. 4, pp. 438-442, Aug. 1994.

[9] Lai-Man Po and Wing-Chung Ma, "A novel Four-Step Search Algorithm for Fast Block Motion Estimation," *IEEE Trans. on CSVT*, vol. 6, no. 3, pp. 313-317, June 1996.

[10] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath and Ashraf Ali Kassim "A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation," *IEEE Trans. on CSVT*, vol. 8, no. 4, pp. 369-377, Aug. 1998.

[11] Lijun Luo, Cairong Zou, Xiqi Gao, Member, IEEE, Zhenya He, Fellow, IEEE, "A New Prediction Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Trans. on CSVT*, vol. 43, no. 1, pp. 56-61, Feb. 1997.

[12] Jae Yeal Nam, Jae Soo Seo, Jin Suk Kwak, Myoung Ho Lee, and Jae Gark Choi, "A New Fast Search Algorithm for Block Matching Motion Estimation," *International Workshop on Advanced Image Technology*, pp. 159-164, Jan. 2000.

[13] Liang-Wei, Jhing-Fa Wang, Jau-Yien Lee, and Jung-Dar Shie, "Dynamic Search-Window Adjustment and Interlaced Search for Block-Matching Algorithm," *IEEE Trans. on CSVT*, vol. 3, no. 1, pp. 85-87, Feb. 1993.

[14] Alexis M. Tourapis, Oscar C. Au and Ming L. Liou, "Fast Motion Estimation using Circular Zonal Search," *Proc of SPIE VCIP*, vol. 3653, pp. 1496-

1504, Jan. 1999.
[15] M. Ghanbari, "The Cross-Search Algorithm for Motion Estimation," *IEEE Trans. on Communications*, vol. 38, no. 7, pp. 950-953, July 1990.
[16] Jer Min Jou, Pei-Yin Chen and Jian-Ming Sun, "The Gray Prediction Search Algorithm for Block Motion Estimation," *IEEE Trans. on CSVT*, vol. 9, no. 6, pp. 843-848, Sep. 1999.
[17] Huan-Sheng Vang and R. M. Mersereau, "Fast Algorithms for the Estimation of Motion Vectors," *IEEE Trans. on Image Processing*, vol. 8, no. 3, pp. 435-438, Mar. 1999.
[18] Ram Srinivasan and K. R. Rao, "Predictive Coding Based on Efficient Motion Estimation," *IEEE Trans. on Communications*, vol. Com-33, no. 8, pp. 888-896 Aug. 1985.
[19] Y. Baek, H. S. Oh and H. K. Lee, "An Efficient Block-Matching Criterion for Motion Estimation and Its VLSI Implementation," *IEEE Trans. Consumer Electron.*, vol. 42, pp. 885-892, Nov. 1996.

저 자 소 개

남 재 열

1983년 2월 : 경북대학교 전자공학과 졸업 (공학사)
1985년 2월 : 경북대학교 대학원 전자공학과 졸업 (공학석사)
1991년 5월 : University of Texas at Arlington 전기공학과 졸업 (공학박사)
1985년 5월 ~ 1987년 7월 : 한국전자통신연구소 연구원
1991년 9월 ~ 1895년 2월 : 한국전자통신연구소 선임연구원
1995년 3월 ~ 현재 : 계명대학교 컴퓨터·전자공학부 조교수
주관심분야 : 영상신호압축, 디지털 방송, 인터넷방송, 영상검색

서 재 수

1998년 2월 : 계명대학교 컴퓨터·전자공학부 졸업 (공학사)
2000년 2월 : 계명대학교 대학원 컴퓨터공학과 졸업 (공학석사)
2000년 3월 ~ 현재 : (주)웰컴정보시스템
주관심분야 : 영상처리, 컴퓨터 그래픽스

이 명 호

1983년 2월 : 숭실대학교 공과대학 전자공학과 졸업 (공학사)
1985년 2월 : 숭실대학교 대학원 전자공학과 졸업 (공학석사)
1996년 3월 : 일본 오사카대학 통신공학과 졸업 (공학박사)
주관심분야 : 영상부호화, 멀티미디어 통신, 컴퓨터 그래픽스, 디지털 방송

곽 진 석

1992년 2월 : 홍익대학교 공과대학 전자공학과 졸업 (공학사)
1994년 2월 : 홍익대학교 대학원 전자공학과 졸업 (공학석사)
1994년 2월 ~ 현재 : 한국전자통신연구원 선임연구원
주관심분야 : 영상부호화, VLSI 알고리즘 및 아키텍처

송 근 원

1993년 2월 : 경북대학교 전자공학과 졸업(공학사)
1995년 2월 : 경북대학교 대학원 전자공학과 졸업 (공학석사)
1998년 2월 : 경북대학교 대학원 전자공학과 종업(공학박사)
1999년3월 ~현재 : 위덕대학교 전자공학과 전임강사
주관심분야 : 영상 압축, 영상 인식, 디지털 신호처리.