韓國컴퓨터情報學會 論文誌
第16卷 第12號, 2011. 12.

2011-16-12-2-2

부화소 단위의 빠른 움직임 예측을 위한 개선된 탐색 알고리즘

김 대 곤* 유 철 중**

An Enhanced Search Algorithm for Fast Motion Estimation using Sub-Pixel

Dae-Gon Kim* Cheol-Jung Yoo**

요 약

움직임 예측은 동영상 인코딩 과정에서 가장 많은 연산량을 차지하는 매우 중요한 처리과정이다. H.264/AVC에서는 움직임 예측을 할 때 정수 화소 단위 뿐만 아니라 부화소 단위까지 예측을 실시함으로써 영상의 압축률을 매우 높일 수 있다. 하지만 이로 인해 계산량이 더욱 증가되는 문제점이 있다. 일반적으로 움직임 예측시 각 블록들 간의 절대 변환 오차(SATD : Sum of Absolute Transform Difference)는 최소점을 기준으로 포물선 형태를 가지는 특성이 있다. 본 논문에서는 이러한 특성을 이용하여 움직임 예측 과정에서 필요한 탐색점을 줄이는 방법을 제안하였다. 제안한 방법은 연산 처리 시간을 감소시킴으로써 계산의 복잡도를 줄일 수 있었다. 제안한 기법을 적용한 결과, 기존의 고속 움직임 예측 알고리즘들과 비교하여 화질 저하는 적은데 비해, 인코딩시 움직임 예측 과정에서 약 20%정도 처리속도를 감소시키는 결과를 얻었다.

▸ Keyword : H.264, 움직임 예측, 예측 움직임 벡터

Abstract

Motion estimation (ME) is regarded as an important component in a video encoding process, because it consumes a large computation complexity. H.264/AVC requires additional computation overheads for fractional search and interpolation. This causes a problem that computational complexity is increased. In Motion estimation, SATD(Sum of Transform Difference) has the characteristics of a parabolic based on the minimum point. In this paper, we propose new prediction algorithm to reduce search point in motion estimation by sub-pixel interpolation

• 제1저자 : 김대곤 • 교신저자 : 유철중
• 투고일 : 2010. 12. 25, 심사일 : 2011. 01. 08, 게재확정일 : 2011. 07. 22.
* 전북대학교 IT응용시스템공학과(Dept. of IT Applied System Engineering, Chonbuk National University)
** 전북대학교 소프트웨어공학과(Dept. of Software Engineering, Chonbuk National University)
※ 이 논문은 2007년 한국정보과학회 한국컴퓨터종합학술대회에 발표한 “가변블록에서의 움직임 특성을 이용한 부화소 단위 고속 움직임 예측 방법” 논문[12]을 확장한 것임
※ 본 연구는 한국과학재단 「기초연구지원사업(특정기초)-과제번호 R01-2004-000-10730-0」에 의해 지원되었습니다.

characteristics. The proposed algorithm reduces the time of encoding process by decreasing computational complexity. Experimental results show that the proposed method reduces 20% of the computation complexity of motion estimation, while the degradation in video quality is negligible.

▸ Keyword : H.264, motion estimation, predictive motion vector

I. 서 론

H.264/AVC 부호화에서 움직임 예측의 계산 복잡도가 막대하기 때문에, 움직임 예측을 빠르게 하는 것은 효율적 H.264/AVC 부호화기 설계의 가장 중요한 쟁점이다. 비디오 영상은 연속된 프레임 간의 높은 상관성으로 인하여 높은 시간적 중복성을 가지고 있다. 이렇게 연속된 프레임 간의 움직임 정보만을 전송함으로써 이전 프레임과의 시간적 중복성을 제거할 수 있어 높은 데이터 압축률을 가진다. 더욱이 디지털 TV의 급속한 보급과 DMB 등의 다양한 크기와 종류의 디스플레이 장치가 대중화된 환경에서 이를 활용하기 위해 영상을 확대하거나 축소하기 위한 기술에 대한 연구가 필요하게 되었다. 특히, 아날로그 영상과는 달리 디지털 영상에서는 원영상을 확대/축소하는 과정에서 변환된 영상의 화소 값 추정과정이 필요하게 되었다[1].

H.264/AVC 동영상 압축 표준에서는 프레임 간의 상관관계를 이용하여 프레임 간의 시간적 중복성을 제거하기 위해 프레임의 화소값을 이용한 움직임 예측(Motion Estimation)과 움직임 보상(Motion Compensation) 기법을 사용하고 있다. 움직임 예측 및 보상 기법은 동영상 압축과정에서 일반적으로 처리 과정에 필요한 전체 연산량의 60% 이상을 차지하고 있는 중요한 과정이다[2]. 두 기법 중 움직임 예측 기법은 크게 화소 순환 알고리즘(Pel Recursive Algorithm: PRA)과 블록 정합 알고리즘(Block Matching Algorithm: BMA)으로 나누어진다. 현재 대부분의 비디오 부호화 표준에서는 계산의 복잡도 및 하드웨어 구현을 고려하여 블록 정합 알고리즘을 널리 사용하고 있다.

블록 정합 알고리즘은 한 블록내의 모든 화소들이 같은 방향으로 평행 이동한다는 것을 가정하여 움직임 추정을 하는 방법으로서, 전역 탐색 블록 정합 알고리즘(Full Search Block Matching Algorithm)은 최적의 정합 블록을 찾기 위해 탐색 영역 내의 모든 지점에서 SAD 값을 비교하기 때문에 최적의 움직임 벡터를 구할 수 있지만 계산량이 많아 전체 부호화 시간의 가장 많은 부분을 차지하게 된다. 이와 같은 단점을 줄이기 위해 탐색점의 수를 줄임으로써 움직임 벡터를 보다 고속으로 추정할 수 있는 방법들이 연구되었다.

하지만, 대부분의 고속 블록 정합 알고리즘들은 현재 블록과 주변 블록의 공간적 상관성을 고려하지 않고 획일적인 방법으로 블록 정합 오차(Block Matching Error: BME)를 계산함으로써 최적의 BMA 지점을 찾아내기 위해 더 많은 계산량을 필요로 하는 단점을 가진다. 더욱이 영상 내 객체들의 실제 움직임 벡터가 샘플링에 사용되는 격자 단위의 정수배 단위로만 발생하지는 않기 때문에 정화소 단위에서 움직임 벡터를 구할 경우 정확한 움직임 벡터가 아닌 경우도 발생한다. 따라서 부화소 단위의 움직임 예측을 통한 움직임 벡터는 압축률을 증가시킬 수 있다.

객체의 움직임 추출에 있어서 움직임 예측(Motion Estimation:ME) 알고리즘이란 일반적으로 동영상 비디오에서 각 객체 혹은 처리 단위 블록이 시간상 앞뒤 프레임에서 어느 위치로 움직였는지를 추정하는 것을 말하는 것으로, MPEG 등의 동영상 압축과 화소 디모자이킹(Pixel Demosaicking), 프레임 보간(Frame Interpolation)등의 비디오 영상 처리에서 광범위하게 이용된다. 움직임 예측의 방법에는 여러 가지가 있으나, 최근의 H.264/AVC 압축표준에서는 매크로 블록(Macro Block) 단위로 현재 압축하고자 하는 매크로 블록과, 앞뒤 여러 장의 프레임의 주변 매크로 블록과의 각 픽셀간의 명도(Luminance)차이를 최소로 하는 블록을 검색하여 움직임 벡터(Motion Vector)를 결정하게 된다. 본 논문에서는 프레임을 16×16의 블록으로 나누어 움직임 예측(ME) 알고리즘을 적용한다.

일반적으로 정화소 움직임 예측은 넓은 탐색 영역으로 인하여 부화소 움직임 예측 연산보다 수십배에 달하는 많은 연산량을 필요로 한다. 따라서 정화소 움직임 예측의 연산량을 감소시키기 위한 많은 연구들이 진행되어져 왔으며, 그 결과로 현재 일반적으로 4~5개 이하의 정화소만으로도 정화소 단위의 움직임 예측이 가능한 탐색 알고리즘[3~5]이 제안되었다. 하지만 이러한 연구의 결과에서는 16개 이상의 부화소 지점을 탐색해야 하는 부화소 단위의 움직임 예측에 필요한 연산량은 감소하지 않아 상대적으로 정화소에 비해 큰 비중을 차지하게 되었다. 따라서 최근에는 부화소 단위의 움직임 예측시 필요한 연산량을 줄일 수 있는 빠른 부화소 단위의 고속 탐색 알고리즘에 대한 연구가 이루어지고 있다.

본 논문에서는 가변 움직임 블록을 위한 기존의 방법에 비

해 부화소 단위에서 움직임 예측 시 탐색점의 수를 줄임으로써 연산 시간을 감소시킬 수 있는 낮은 계산복잡도의 빠른 개선된 탐색 알고리즘을 제안하고자 한다. 이를 위해 2장에서는 부화소 단위에서의 고속 움직임 예측에 대한 기존 연구방법들에 대해 간략히 소개하고, 3장에서는 이러한 고속 움직임 탐색 기법보다 개선된 탐색 알고리즘을 제안한다. 또한 4장에서는 제안한 방법과 기존의 방법들과의 수행 시간 및 성능을 비교 분석한 결과를 통해 제안한 방법에 대한 장단점을 기술하고, 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

II. 부화소 움직임 예측 특성 및 기존 탐색 알고리즘

움직임 벡터를 추정하는 알고리즘으로 본 논문에서는 정합의 척도인 평균 절대 오차(Mean Absolute Difference: MAD)를 기준으로 지정된 값 이상의 MAD를 가진 블록의 움직임 벡터를 추정한다. H.264/AVC에서는 보다 정확한 움직임 예측을 위하여 정화소 단위에서 움직임 예측을 수행하여 BMA가 최소가 되는 지점을 중심으로 부화소 움직임 예측을 수행하여 최소값을 갖는 지점을 찾는다.

2.1 부화소 움직임 벡터 예측

움직임 예측 연산을 실시할 때 부화소 단위 탐색 연산은 정화소 단위 탐색 연산을 수행하였을 때보다 해당 블록의 정확한 움직임을 구함으로써 복원되어지는 영상의 화질이 더욱 좋아지게 된다. 하지만 추가적인 부화소 단위의 보간 및 탐색 과정으로 인한 연산량이 증가되는 단점을 가지게 된다. 또한 항상 부화소 단위의 움직임 예측값이 정화소 단위의 움직임 예측 값보다 더 정확한 움직임 벡터값을 찾았다는 것을 의미하지는 않는다.

많은 고속 탐색 알고리즘은 현재 블록의 움직임 벡터가 주변 블록들의 움직임 벡터와 유사하다는 가정하에 초기 탐색 지점으로 예측된 움직임 벡터(PMV : Predicted Motion Vector)를 사용한다. 여기서 예측 움직임 벡터(PMV)는 정화소와 부화소 움직임 벡터가 모두 포함되어 있으므로 연산에 필요한 부화소 움직임 벡터만을 따로 분리시켜야 한다.

식 1)은 움직임 예측에 필요한 부화소 움직임 벡터를 구하고 있다. H.264/AVC에서는 1/4 부화소 단위까지 탐색을 하므로 일반적으로 β는 4가 된다.

$$pred\_frac\_mv = \left(pred\_mv - \int\_mv\right) \% \beta \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

※ pred_mv : 정화소와 부화소 움직임 벡터
int_mv : 정화소 움직임 벡터
β : 부화소 움직임 벡터의 정확도

위 식 1)을 바탕으로 부화소 움직임 추정 연산을 수행하게 되면 대부분의 영상에서 부화소 예측 움직임 벡터(PMV)와 실제 부화소 움직임 벡터(MV)가 일치할 경우가 70%를 넘었지만 실험영상 "Foreman"과 같은 영상의 경우는 상대적으로 영상 내부 블록의 움직임이 비교적 많기 때문에 예측 움직임 벡터(PMV) 값과 실제 움직임 벡터(MV) 값이 일치한 경우가 적었다. 그렇지만 부화소 예측 움직임 벡터를 중심으로 1/2 화소 내에 부화소 움직임 벡터가 있을 경우는 모든 영상에서 대략 90%가 넘었다. 따라서 부화소 움직임 예측시 주위 ±1 픽셀 이내에서 해당 지점을 보다 빠르게 찾을 수 있는 탐색 패턴이 사용되어야 한다[6].

H.264에서는 일반적으로 1/2화소와 1/4화소 단위에서의 움직임 예측은 [그림 1]과 같이 각 방향의 이전 화소 단위에서 구해진 최소 오차 점을 기준으로 하여 상하좌우 대각선의 8점을 탐색하여 각 화소 단위에서의 움직임 벡터를 찾는다. 결과적으로 정수 화소에서만 움직임 예측을 하는 경우보다 1/2화소에서 8개, 1/4화소에서 8개 등 총 16개의 SAD 연산이 필요하다는 단점이 있다. 이러한 추가적인 연산을 줄이기 위한 대표적인 고속 알고리즘으로는 2차원 로그 탐색(Two Dimensional Logarithm Search: 2-D LOG) 알고리즘과 3단계 탐색(Three Step Search: TSS) 알고리즘, 최근접 이웃 탐색(Nearest-Neighbors Search) 알고리즘, 육각 패턴 탐색(HEXBS) 알고리즘, 개선된 크로스-납작한 육각 탐색 패턴(Enhanced Cross-Flat Hexagon Search Pattern) 알고리즘[7] 등이 있다.

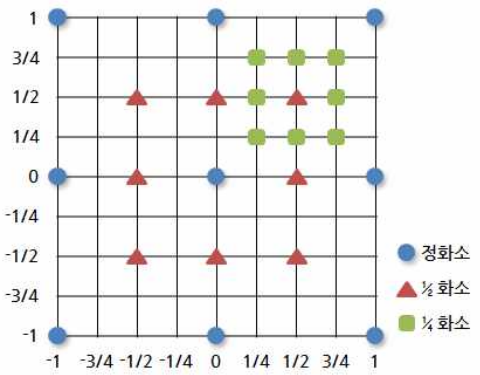

그림 1. 정화소와 부화소의 위치
Fig. 1. Location of Integer and Sub-pixel

2.1.1 NNS(Nearest-Neighbors Search) 알고리즘

NNS 알고리즘은 기존의 고속 알고리즘들의 단점인 PSNR (Peak Signal to Noise Ratio) 저하 현상을 크게 향상시킨 알고리즘이다. 계산량은 TSS 방법과 거의 유사하지만 PSNR 값은 기존의 고속 알고리즘보다 0.5dB~1dB 정도 높다[8].

<표 1>은 NNS 알고리즘에서 주위 블록의 코딩 모드에 따른 PMV 및 탐색 경로(Search Path)를 나타내고 있다. NNS 알고리즘은 주위 블록들의 코딩 모드에 따라 2가지 모드로 나뉘게 된다. NNS 알고리즘의 첫 번째 모드는 현재 블록의 왼쪽과 위쪽 블록인 MB0과 MB1이 인트라 모드(INTRA mode)인 경우로서 TSS(Three Step Search) 방법과 동일한 과정을 수행한다. 두 번째 모드는 주변 블록으로부터 예측 움직임 벡터(PMV)를 구하고 이 벡터를 중심으로 NN(Nearest-Neighbors) 탐색을 하는 경우이다.

표 1. NNS 탐색 경로
Table 1. NNS Search Path

코딩 모드			Predicted Motion Vector	탐색경로
MB0	MB1	MB2		
I	I	×	(0,0)	Three-Step
I	P	×	MV1	Nearest-Neighbors
P	I	×	MV0	Nearest-Neighbors
P	P	I	Mean(MV0, MV1)	Nearest-Neighbors
P	P	P	Median(MV0, MV1, MV2)	Nearest-Neighbors

주) 1. I:INTRA, P:INTER, ×:don't care
2. MB0~MB2의 위치는 [그림 4] 참조
3. MV0는 MB0의 MV를 의미함

[그림 2]는 NNS 방법의 두 번째 모드를 나타내고 있다. 두 번째 모드의 경우 예측 움직임 벡터(PMV)를 중심으로 다이아몬드 탐색을 한다. 이때 다이아몬드 탐색점 중에서 가운데 탐색점이 최소 SAD 값을 가진다면 탐색을 중지하고 그 점을 움직임 벡터로 설정한다.

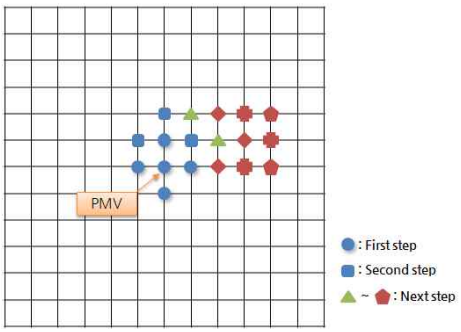

그림 2. NNS(두번째 모드)
Fig. 2. NNS(Second mode)

그러나 예측 움직임 벡터(PMV)가 현재 블록의 움직임 벡터와 항상 유사하다고는 할 수 없다. 예를 들면, 영상에서 장면의 전환이 발생했거나 해당 블록의 위치가 물체의 경계면일 때, 또는 물체의 출현 및 소멸이 발생했을 때에는 예측 움직임 벡터(PMV)는 현재 블록의 움직임 벡터와 크게 다를 수도 있다. 즉, 유사성이 전혀 없을 수도 있다. 따라서 예측 움직임 벡터(PMV)와 현재 블록의 움직임 벡터 간의 유사성, 다시 말해 예측 움직임 벡터(PMV)의 신뢰도를 판단하는 규칙을 어떻게 정하느냐가 중요한 문제가 된다. 예측 움직임 벡터(PMV)의 신뢰도를 판단하기 위해서 NNS 방법에서는 MBi (i=0,1,2) 각각의 코딩 모드로 판단을 하였다.

PMV의 신뢰도 판정은 우선 PMV에서의 SAD 값 SAD(PMV)를 구하여 NNS 방법 1단계에서 9개 탐색점에 대한 SAD값 SAD(NNS-1) 과 비교한다. SAD(PMV) 가 SAD(NNS-1) 보다 더 작거나 같으면 PMV는 신뢰도가 높다고 판단하고, 그렇지 않다면 PMV는 신뢰도가 없는 것으로 판단한다. 이렇게 구해진 PMV의 신뢰도에 따라 NNS 알고리즘은 2가지 탐색경로를 갖는다. PMV가 신뢰도가 높으면 PMV 주변 영역에서 세부 검색을 수행하고, 그렇지 않으면 일반적인 NNS 탐색의 나머지 과정을 수행한다.

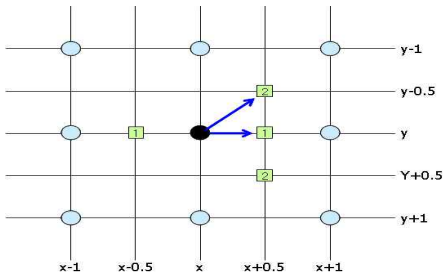

그림 3. 2SS(Two-Step Search) 방법의 예
Fig. 3. Example of 2SS(Two-Step Search) Method

2.1.2 2SS(Two-Step Search) 알고리즘

2SS(Two-Step Search) 알고리즘은 정수 화소 검색 이후 수평 방향 2개의 위치 중 최소 SAD를 갖는 위치를 수직 탐색 중심으로 설정한 후 수직 방향 2개의 위치에 대해서 최소의 SAD를 갖는 위치를 순차적으로 탐색하는 기법이다[9].

[그림 3]은 전형적인 2SS 알고리즘을 보여주고 있다. 2SS 알고리즘은 정수배 화소 단위에서 가장 작은 SAD값을 알고 있고, 수평과 수직 방향으로의 1/2화소 단위에서 움직임 예측을 한다. 따라서 대부분의 경우 1/2화소 단위의 4개 점에서 움직임 예측을 한다. 만약 정수배 단위에서의 찾아진 화소점이 프레임의 상단, 하단 또는 좌우의 끝에 위치할 경우 탐색점의 수는 줄어들게 된다.

단계 1: 정수배 화소 단위에서 찾은 SAD가 가장 작은 화소 위치인 (x, y)를 중심으로 탐색을 시작한다. 먼저 수평 위치의 각 화소 (x±0.5, y), (x, y) 에서의 SAD값들을 계산하여 비교한다. 이중 가장 작은 SAD값을 갖는 화소를 수직 방향 탐색의 중심인 (p, q)로 정한다.

단계 2: 수직 위치의 각 화소 (p, q±0.5), (p, q) 에서의 SAD값들을 계산하여 가장 작은 SAD값을 갖는 화소점이 1/2단위에서의 움직임 벡터가 된다.

그러나 2SS 알고리즘에서는 움직임 벡터를 구하기 위해 반-전역 탐색 알고리즘(Half-Full Search Algorithm)을 사용하여 움직임 벡터의 후보를 선택한다. 따라서 전역탐색 알고리즘보다는 움직임 벡터를 예측하는 시간이 짧지만 TSS 방식이나 NTSS 방식보다는 거의 같거나 느려지게 된다.

III. 부화소 단위의 빠른 움직임 예측 알고리즘

3.1 개 요

H.264/AVC 압축표준에서는 움직임 벡터 예측시 1/4화소 단위의 정확도를 갖기 위해 부화소(Sub-pixel) 움직임 탐색을 추가로 실시한다. 일반적으로 부화소 단위 움직임 탐색을 위해 H.264에서는 6-tap FIR 필터를 이용하여 참조 영상을 1/2화소 단위로 보간한 후, 얻어진 1/2위치의 화소값들을 다시 선형 보간법을 사용하여 1/4 위치의 화소를 구한다. 이를 통해 정수 화소 위치의 움직임 벡터를 중심으로 주변에 위치한 각 부화소 가운데 최소 SATD 값을 갖는 위치 지점을 찾아 실제 움직임 벡터와 가장 흡사한 최종 움직임 벡터를 확정할 수 있게 된다.

본 논문에서 제안하는 방법은 6-tap FIR 필터로 1/2화소를 보간하는 H.264/AVC 표준에서 양 옆의 정수 화소로부터 가장 큰 영향을 받는 특성을 이용하여 SATD가 가장 작은 화소와 SATD가 두 번째로 작은 화소 사이의 1/2화소와 1/4화소에서의 움직임 예측 알고리즘을 제안하였다. 제안한 알고리즘에서는 부화소 단위 움직임 탐색전에 미리 예측 움직임 벡터(PMV)를 구한다. 그 이후, 앞에서 구한 예측 움직임 벡터를 이용하여 보상된 위치의 블록에 대한 SATD값과 기존 연구의 2SS 알고리즘의 첫 번째 탐색점의 SATD 값에 따라 최소값을 갖는 곳을 탐색 초기 지점으로 정한다. 이렇게 탐색 초기점이 정해지면 2SS 알고리즘과 다르게 1/2과 1/4 화소 단위에서의 움직임 예측시 탐색점을 4개에서 3개로 줄여 움직임 예측을 실시한다. 이로써 기존 4개의 탐색점에서 실시하는 움직임 예측 방법보다 부화소 단위의 연산시간을 상당히 줄일 수 있을 뿐만 아니라 빠르면서도 낮은 복잡도를 가지게 된다.

3.2 예측 움직임 벡터(PMV) 계산

MPEG-1/2/4나 H.263/264와 같은 표준에서 다양한 움직임 벡터 예측 방식이 제안되어 왔다. 예측의 정확성을 높이고 움직임 예측의 복잡도를 줄이기 위해 공간적·시간적으로 이웃한 블록들의 MV를 이용한다[10]. [그림 4]는 움직임 벡터 값을 구하고자 하는 매크로블록과 연산시 참조하게 되는 주위의 매크로블록의 위치를 나타내고 있다. 여기서 MB0, MB1, MB2는 각각 좌측, 상단, 우측상단에 위치한 매크로블록을 나타낸다.

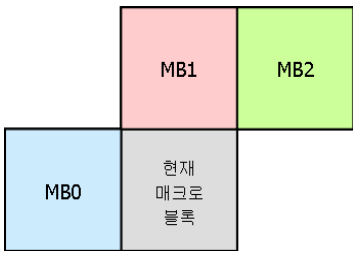

그림 4. 참조할 주변 매크로블록
Fig. 4. Surrounding Macroblock to Reference

각각의 MBi (i=0,1,2)에 대한 예측 움직임 벡터(PMV)을 구하는 방법으로는 현재 영상을 비교적 움직임이 적은 영상이라 가정하고 식 2)를 통해 중간값을 계산하여 이 값을 사용하는 것이 가장 일반적인 방법으로 알려져 있다.

$$PMV = median(MV_0, MV_1, MV_2) \quad \cdots\cdots (2)$$

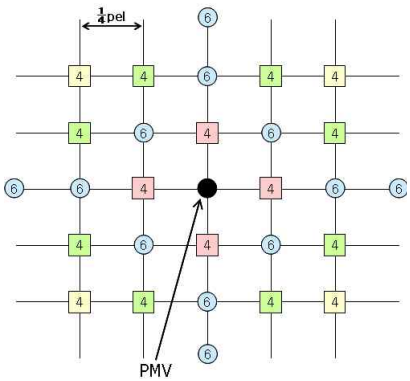

그림 5. PMV의 주변 각 움직임 벡터의 부호 길이
Fig. 5. Encoding Length of Each Motion Vector on PMV's Surrounding

[그림 5]는 예측 움직임 벡터(PMV)를 중심으로 각 움직임 벡터의 부호 길이를 나타낸 것이다. 즉, DMV(Differential Motion Vector)를 부호화하는데 필요한 비트수를 각 화소 위치마다 나타낸 것이다. DMV는 PMV(Predictive Motion Vector)와 현 블록의 움직임 벡터에 차이 값을 나타낸다. 일반적으로, 기하학적인 확률 모델과 잘 부합하는 EG(Exponential Golomb) 부호는 DMV를 부호화하는데 효율적이므로, 이러한 부호의 길이로부터 DMV가 발생할 확률을 계산할 수 있다. 따라서 구하고자 하는 영상의 움직임 벡터와 주변 블록들의 움직임 벡터의 중간값 간의 차분치를 구하여 이를 차분 부호화함으로써 엔트로피 부화화의 효율을 더욱 높일 수 있게 된다.

만약, 움직임 벡터가 예측 움직임 벡터(PMV)와 동일한 경우에는 부호 길이는 1비트가 되고 이 지점을 중심으로 블록 거리가 1인 동, 서, 남, 북의 지점에 위치할 경우에는 4비트가 된다. 따라서 이러한 부호 길이가 나타내는 통계적 특성을 구하면, 움직임 벡터가 예측 움직임 벡터와 동일할 확률은 대략 50%정도이며, 예측 움직임 벡터를 포함하여 작은 다이아몬드 검색(SDS; Small Diamond Search) 위치에 움직임 벡터가 존재할 확률이 거의 75%에 이른다. 따라서 부화소 단위에서 움직임 벡터에 대한 검색에 SDS 방법을 사용한다면, 부호화 효율에 대한 성능 저하는 최소로 하면서 적은 연산량만으로도 고속 움직임 벡터 탐색이 가능하게 된다[11].

3.3 N2SS(New Two-Step Search) 알고리즘

부화소 PMV를 초기 탐색점으로 부화소 움직임 예측을 수행하면 탐색 중심점 근처에서 움직임 벡터가 발견될 확률이 높아 고속 탐색 알고리즘에 매우 적합하다. [그림 3]에서와 같이 부화소 움직임 벡터는 부화소 PMV 주변 [-2, +2]안에 대부분 존재하지만 해당 범위안의 모든 부화소 지점을 탐색하게 되면 오히려 탐색점의 수가 많아진다. 따라서 부화소 PMV 주변 [-2, +2] 영역을 효과적으로 탐색하기 위한 방법이 필요하다.

표 2. 정/부 화소 움직임 벡터 분석 (단위: %)
Table 2. Analyzing the integer/sub-pixel MV (unit: %)

Sequence Type	Integer-pixel		Half-pixel				Quarter-pixel			
	(0,0)	(0,0) (0,1) (1,0)	(0,0)	8 point	(0,1) (1,0)	(0,0) (0,1) (1,0)	(0,0)	8 point	(0,1) (1,0)	(0,0) (0,1) (1,0)
Bike	33.25	67.62	45.64	54.36	34.39	80.02	50.61	49.39	34.08	84.69
Car phone	50.80	86.15	47.86	52.14	37.23	85.09	39.50	60.50	40.82	80.32
Flower garden	17.67	94.71	32.12	67.88	60.54	92.66	40.40	59.60	46.96	87.37
Foreman	35.98	74.96	43.38	56.62	39.51	82.89	40.19	59.81	40.26	80.44
Missa	82.29	97.90	73.88	26.12	19.90	93.78	65.16	34.84	25.46	90.61
Mother & daughter	86.80	96.58	72.99	27.01	21.14	94.13	54.07	45.93	33.49	87.56
Silent	86.15	95.38	82.24	17.76	13.18	95.41	69.64	30.36	21.80	91.44
Susie	53.08	83.17	48.12	51.88	34.33	82.45	42.79	57.21	34.77	77.56
Table tennis	30.75	67.38	47.55	52.45	37.92	85.47	40.48	59.52	41.31	81.80

주) 참고문헌 [10]에서 수행한 실험 데이터의 일부임

<표 2>를 보면 정수화소에서 움직임 벡터가 (0,0)인 경우가 전체의 약 17~87%를 차지하고 수평 혹은 수직이 0인 경우를 포함하면 약 67~97%에 이름을 알 수 있다. 그리고, 1/2 화소에서 움직임 벡터가 (0,0)인 경우는 약 32~82%, 수평 혹은 수직이 0인 경우를 포함하면 약 80~95%에 해당함을 알 수 있다. 또한, 1/4 화소에서 움직임 벡터가 (0,0)인 경우는 약 39~69%, 수평 혹은 수직이 0인 경우를 포함하면 약 77~91%에 해당함을 알 수 있다. 이러한 결과들로부터 움직임 벡터가 (0,0)인 경우와 수평/수직 방향인 경우를 찾아냄으로써 움직임 예측의 효율화를 기대해 볼 수 있음을 알 수 있다.

따라서 본 알고리즘은 기존 2SS 알고리즘에서 필요한 탐색점의 수를 줄이는 개선된 탐색 알고리즘[12]으로서 이러한 특성을 이용하여 SATD가 가장 작은 지점과 SATD가 두 번째로 작은 지점 사이의 부화소에서의 움직임 예측을 실시한다. 즉, 기존의 방법이 1/2화소와 1/4화소 단위에서 각각 8개씩의 점을 탐색하던 것을 평균 3개의 탐색점으로 줄임으로써 움직임 예측 단계에서의 연산량을 줄이고 처리 시간을 빠르게 하는 장점을 가지게 된다. 따라서 (x, y)점에서 SATD값이 최소라고 할 때, 1/2화소 단위의 움직임 벡터는 다음과 같이 과정을 통해 구할 수 있다.

단계 1 : 정수배 화소 단위에서 SATD값이 가장 작은 점과 예측 움직임 벡터(PMV)를 이용하여 보상된 위치의 SATD값을 비교하여 두 값 중 최소값을 갖는 곳을 탐색 초기 지점으로 정한다.

단계 2 : 단계 1의 탐색 초기 지점을 중심으로 SATD값의 분포 특성을 이용하여 최소점의 ±1 내에 있는 이웃 화소의 SATD값을 구하여 2번째로 작은 SATD값을 가지는 점을 찾는다.

단계 3 : 단계 2에서 찾아낸 두 번째로 작은 점의 방향과 같은 방향의 부화소들 중 가장 주변에 위치한 3개의 화소들에 대해 움직임 예측을 한다.

i) : (x, y-1)의 수직 위치가 두 번째로 작은 SATD를 갖는 경우, (x, y-0.5), (x±0.5, y-0.5)의 세 점의 부화소 단위 SATD값을 계산한다. (x, y+1)이 두 번째로 작은 SATD값을 갖는 경우에도 위와 비슷한 방법을 적용한다.

ii) : (x+1, y)의 수평 위치가 두 번째로 작은 SATD를 갖는 경우, (x+0.5, y±0.5), (x+0.5, y)의 세 점의 부화소 단위 SATD값을 계산한다. (x-1, y)이 두 번째로 작은 SATD값을 갖는 경우에도 위와 비슷한 방법을 적용한다.

iii) : (x+1, y+1)과 같은 대각선 위치의 점이 두 번째로 작은 SATD를 갖는 경우, 대각선 방향의 부화소 (x+0.5, y-0.5), (x-0.5, y+0.5)와 최저점 주변의 두 점 (x, y-0.5), (x+0.5, y)등 총 4개의 위치에서 SATD를 계산하여 움직임 예측을 한다. 다른 대각선 방향일 경우에도 위와 비슷한 방법을 적용한다.

[그림 6]은 제안한 방법의 순서도로서 주요 과정을 보여주고 있다.

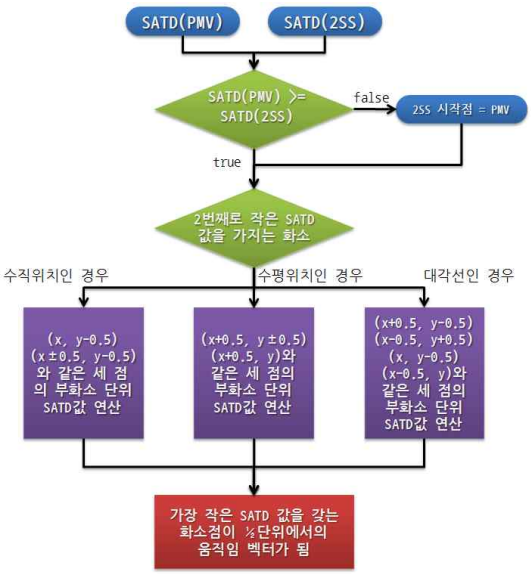

그림 6. 제안한 방법의 순서도
Fig. 6. Flow-Chart of Proposed Algorithm

IV. 실험 및 결과 분석

4.1 실험 환경

제안한 알고리즘의 성능을 평가하기 위하여 본 논문에서는 H.264/AVC 참조 소프트웨어 JM 10.2[13]를 사용하였고, 테스트 방법은 JVT에서 권고하는 시뮬레이션 조건[14]을 따르도록 하였다. 또한 기존에 제안된 방식과 본 논문에서 제안한 알고리즘의 성능을 분석하였다. <표 3>은 H.264 표준과 객관적인 실험 결과 비교를 위해 제안된 실험조건이다.

표 3. 실험 조건
Table 3. Test Condition

영상 / 조건 특성	news (qcif)	container (qcif)	foreman (qcif)	silent (qcif)	akiyo (qcif)	football (qcif)
Frame rate(Hz)	10	10	10	15	15	15
Total frames	100	100	100	150	150	150
QP	28, 32, 36, 40					
Coding options used	R-D optimization, Hadamard transform, IPPP structure, CAVLC, no error tools					
H.264 Codec	JM 10.2 encoder, Baseline Profile					

본 논문에서는 제안한 알고리즘의 성능을 시험하기 위하여 각각 다양한 특징을 갖는 영상을 선정하여 각 실험 영상을 각각 5번씩 부호화를 수행한 후 평균을 측정하였다.

정수 화소 단위에서는 JM 10.2에서 구현된 고속 움직임 탐색 알고리즘을 그대로 사용하였고, 부화소 단위에서는 탐색점의 수를 줄이는 본 논문에서 제안한 방법과 기존의 움직임 탐색 알고리즘을 비교한 뒤 그 성능 평가 결과를 <표 4>에 정리하였다.

표 4. 성능 평가 비교
Table 4. Comparing the Performance Evaluation

영상	구분	JM10.2	2SS	DS	Proposed
News	비트율 (kbps)	95.7	95.8	95.4	95.5 (-0.2)
	PSNR (dB)	34.6	33.90	33.93	34.3 (-0.3)
	복호화 시간 (Sec)	43.9	38.8	37.1	37.2 (84.7%)
	부화소 연산 처리시간 (Sec)	9.32	6.15	6.34	6.22 (66.7%)
Silent	비트율 (kbps)	56.2	56.3	55.9	56.4 (+0.2)
	PSNR (dB)	32.2	32.85	32.86	32.3 (+0.1)
	복호화 시간 (Sec)	38.7	32.8	31.4	31.9 (82.4%)
	부화소 연산 처리시간 (Sec)	7.82	5.22	5.34	5.10 (65.2%)
Container	비트율 (kbps)	48.7	48.5	48.6	48.6 (-0.1)
	PSNR (dB)	31.8	32.28	32.32	32.0 (+0.2)
	복호화 시간 (Sec)	39.1	31.3	31.5	31.2 (79.8%)
	부화소 연산 처리시간 (Sec)	8.03	5.61	5.58	5.58 (69.5%)
Foreman	비트율 (kbps)	49.5	49.8	49.5	50.1 (+0.6)
	PSNR (dB)	35.1	31.97	32.02	34.7 (-0.4)
	복호화 시간 (Sec)	40.2	36.9	37.1	36.5 (90.8%)
	부화소 연산 처리시간 (Sec)	8.98	7.84	8.01	7.82 (87.1%)
Akiyo	비트율 (kbps)	43.9	43.5	43.6	43.8 (-0.1)
	PSNR (dB)	38.7	38.04	39.08	38.8 (+0.1)
	복호화 시간 (Sec)	41.7	35.5	34.9	34.6 (83.0%)
	부화소 연산 처리시간 (Sec)	8.73	5.62	6.15	5.86 (67.1%)
Football	비트율 (kbps)	49.5	50.2	49.9	50.2 (+0.7)
	PSNR (dB)	33.7	33.29	33.31	32.9 (-0.8)
	복호화 시간 (Sec)	43.2	39.8	39.3	39.1 (90.5%)
	부화소 연산 처리시간 (Sec)	9.41	8.74	8.16	8.12 (86.3%)

제안된 방법과 기존의 방법과의 동영상 화질 비교를 위한 측정 단위를 위한 MSE와 PSNR의 식은 다음 식 3), 4)와 같다.

$$MSE = \left(\frac{1}{M \times N}\right)\sum_{x=1}^{M}\sum_{y=1}^{N}\left[O_t(x,y) - E_t(x,y)\right]^2 \quad \cdots\cdots (3)$$

$$PSNR = 10\log_{10}\left(\frac{(2^n-1)^2}{MSE}\right) \quad \cdots\cdots (4)$$

※ (2n-1) : The maximum pixel value of the image,
n : Bits of the image

실험 결과, 제안한 알고리즘은 기존의 알고리즘과 비교하여 화질을 판단할 수 있는 PSNR이 평균적으로 -0.4dB정도 감소하였지만 거의 같은 성능으로 볼 수 있다. 비트량은 기존의 탐색 방법보다 0.2%정도 증가하였으나 이 또한 동일한 성능으로 볼 수 있다. 연산시간은 기존의 2단계 탐색 알고리즘 방법(2SS)에 비해 인코더 처리과정에서 약 20%정도 감소하였다. 또한, 부화소 단위에서의 연산처리 시간을 비교해 보면 기존의 방법보다 약 26%정도 연산처리 시간의 감소가 있음을 알 수 있었다.

<표 4>에서 Silent, Container 영상들의 부호화 시간 감소량이나 비트 감소량은 다른 영상들에 비해 매우 작다. 이들 두 영상은 단순한 움직임을 갖고 있기 때문에 근원적으로 움직임 벡터의 발생 빈도가 매우 낮기 때문이다. 반면에 News, Akiyo, Foreman, Football 영상은 제안한 방법으로 비트량 변화량이 평균 10%이상 줄어들었으며 부호화 시간은 감소도 평균 32% 이상이었다. 이는 본 논문에서 제안된 고속 움직임 예측 알고리즘은 움직임이 크고 난잡한 영역이 많은 영상에 매우 효율적임을 나타낸다.

V. 결론 및 향후연구

본 논문에서는 기존의 방법에 비해 PSNR 손실을 최대한 줄이면서 부화소 단위에서 움직임 예측 시 탐색점의 수를 줄임으로써 연산 시간을 감소시킬 수 있는 개선된 부화소 탐색 알고리즘을 제안하였다. 제안된 알고리즘은 처음 단계로는 예측 움직임 벡터의 특성을 이용하여 부화소 단위에서 작은 SATD값을 이용하여 움직임 벡터를 예측함으로써 최적의 움직임 벡터를 찾는다. 그리고 SATD값이 작은 화소점의 방향에 위치한 화소점들만 탐색을 함으로써 기존 2SS 알고리즘과 달리 탐색점을 4개에서 3개로 줄였다. 제안한 알고리즘으로 다양한 움직임을 갖는 테스트 영상을 가지고 실험한 결과

2SS방법과 널리 알려진 거의 비슷한 성능을 가지면서 인코더 처리과정에서 기존의 2단계 탐색 알고리즘에 비해 일부 연산 시간이 약 20% 정도 감소하는 경우가 나타났고, 부 화소 단위의 움직임 예측에 있어서는 기존의 알고리즘들에 비해 30% 이상의 연산시간 감소를 확인하였다.

이러한 결과를 바탕으로 본 논문에서 제안한 탐색 알고리즘은 고화질의 영상을 필요치 않는 이동형 단말기 및 멀티미디어 기기 상에서 H.264를 이용한 영상 재생시 움직임 예측의 효율성 개선에 더욱 효과가 있으리라 본다. 끝으로 ,현재 휴대용 이동기기에 포함되어짐으로써 다양한 형태의 멀티미디어 기기의 활용과 사용자의 관심이 많아지고 있으므로, 향후에는 해당 이동기기에 최적화된 알고리즘으로 높은 화질과 압축 효율이 가능한 연구가 계속 이루어져야 할 것이다.

참고문헌

[1] Yong-Kwang Kwon, “Hybrid Interpolation using Intra Prediction Information of H.264/AVC,” Journal of the Korea Society of Computer and Information, vol. 13, no. 7, pp. 83-90, July 2008.

[2] Y. J. Wang, C. E. Cheng, and T. S. Chang, "A fast fractional pel motion estimation algorithm for H.264/MPEG-4 AVC," IEEE Int. Symp. On Circuits and Systems, pp. 3974-3977, 2006

[3] X. Q. Banh and Y. P. Tan, "Adaptive Dual-Cross Search Algorithm for Block-Matching Motion Estimation," IEEE Transactions on Consumer Electronics, vol. 50, no. 2, pp. 766-775, 2004

[4] X, Yi and N. Ling, "Improved partial distortion search algorithm for rapid block motion estimation via dual-halfway-stop," IEE Int. Conf. Acoustics, Speech and Signal Processing, pp. 917-920, 2005

[5] Kwang Woo Lee, Myung Hoon Sunwoo, “Motion -based Fast Fractional Motion Estimation Scheme for H.264/AVC” Journal of IEEK, vol. 45, no. 3, pp. 64-79, May. 2008.

[6] Seong Hyeon Jo and Jong Hwa Lee, “A Center Biased Cross-Diamond Search Algorithm for Fast Fractional-pel Motion Estimation,” Journal of IEEK, vol. 46, no. 2, pp. 78-84, Feb. 2009.

[7] Hyeon-Woo Nam, "A Fast Block Matching Motion Estimation Algorithm by using an Enhanced Cross-Flat Hexagon Search Pattern," Journal of the Korea Society of Computer and Information, vol. 13, no. 7, pp. 99-108, July 2008.

[8] M. Gallant, G. Côté and F. Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding", IEEE Trans. Image Processing, vol. 8, no. 12, Dec. 1999.

[9] Bo Zhou and Jian Chen, “A Fast Two-step Search Algorithm for Half-pixel Motion Estimation,” IEEE, vol. 2, pp. 611-614, May. 2003.

[10] Alexis M. Tourapis, Oscar C. Au, and Ming L. Liou, "Highly Efficent Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation," IEEE Transactions Circuits Systems for Video Technolo gy, vol. 12, no. 10, pp. 934-947, Oct. 2002.

[11] Chun-Ho Cheung and Lai-Man Po, "A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation," IEEE Transactions Circuits Systems for Video Technology, vol. 12, no. 12, pp. 1168-1177, Dec. 2002.

[12] D. G. Kim, S. J. Kim, C. J. Yoo, and O. B. Chang, "A Fast Sub-pixel Motion Estimation Algorithm Using Motion Characteristics of Variable Block Sizes," Proceeding of Korea Computer Congress 2007, pp. 560-565, Jun. 2007.

[13] H.264/AVC Reference Software Version JM10.2, http://iphome.hhi.de/suehring/tml/download/old_jm/jm102.zip

[14] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), 2003.

저 자 소 개

김 대 곤

2006 : 전북대학교 컴퓨터공학과 졸업 (공학사)

2008 : 전북대학교 대학원 컴퓨터공학과 졸업(공학석사)

현 재 : 전북대학교 대학원 IT응용시스템공학과 박사수료

관심분야 : 소프트웨어공학, 안티 포렌식 디지털 포렌식, 멀티미디어 처리

Email : bluve0214@jbnu.ac.kr

유 철 중

1982 : 전북대학교 전산통계학과 졸업 (이학사)

1985 : 전남대학교 대학원 계산통계학과 졸업(이학석사)

1994 : 전북대학교 대학원 전산통계학과 졸업(이학박사)

1997~현 재 : 전북대학교 공과대학 소프트웨어공학과 교수

관심분야 : 소프트웨어 개발 프로세스, 소프트웨어 품질, 소프트웨어 테스팅, 컴포넌트 기반 소프트웨어, 소프트웨어 메트릭스, 소프트웨어 에이전트, 지능형 로봇 소프트웨어, 임베디드 소프트웨어 및 테스팅, 멀티미디어 소프트웨어, GIS, 교육공학, 인지과학 등임.

Email : cjyoo@jbnu.ac.kr