효율적인 움직임 추정을 위한 고속 블록 정합 알고리듬의 적응적 선택

준회원 김 정 준*, 전 광 길*, 정회원 정 제 창*

Adaptive Selection of Fast Block Matching Algorithms for Efficient Motion Estimation

Jung-Jun Kim*, Gwang-gil Jeon* ***Associate Members***,
Je-chang Jeong* ***Regular Member***

요 약

본 논문에서는 기존의 고속 움직임 추정 방법과 새롭게 제안하는 고속 움직임 추정 방법들을 적응적으로 사용하는 기법을 제공한다. 이 알고리듬의 이름은 AUDC이며, 새롭게 제안하는 고속 움직임 추정 방법은 다이아몬드 탐색와 3단계 탐색 기법을 기초로 한다. 비록 많은 고속 움직임 추정 방법들이 각각 많은 장점을 가지고 있지만 단점도 가지고 있다. 그러므로 한가지의 고속 움직임 추정 방법을 사용하는 것보다는 적응적으로 고속 움직임 추정 기법을 선택하여 사용하는 것이 더욱 효율적이라 할 수 있다. 그래서 제안하는 움직임 추정 기법은 현재 블록에 이웃하는 블록의 Motion Vector의 길이, 탐색점의 수, SAD를 이용하여 다이아몬드 탐색 기법, 십자 3단계 탐색 기법 그리고 개량된 십자 다이아몬드 기법을 적응적으로 사용한다. 실험결과 AUDC는 많이 알려진 다른 고속 움직임 추정 방법보다 화질을 향상시킬 뿐만 아니라 탐색점의 수 또한 향상됨을 알 수 있다.

Key Words : Motion Estimation, Motion Vector, Block Matching Algorithm, Diamond Search, SAD

ABSTRACT

A method that is adaptively selecting among previous fast motion estimation algorithms and a newly proposed fast motion estimation algorithm(UCDS) is presented in this paper. The algorithm named AUDC and a newly proposed fast motion estimation algorithms are based on the Diamond Search(DS) algorithm and Three Step Search(TSS). Although many previous fast motion estimation algorithms have lots of advantages, those have lots of disadvantages. So we thought better adaptive selection of fast motion estimation algorithms than only using one fast motion estimation algorithm. Therefore, we propose AUDC that is using length of the MV, Search Point, SAD of the neighboring block and adaptively selecting among Cross Three Step Search(CTSS), Diamond Search(DS) and Ungraded Cross Diamond Search(UCDS). Experimental results show that the AUDC is much more robust, provides a faster searching speed, and smaller distortions than other popular fast block-matching algorithms.

※ 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2006-005-J04101).
* 한양대학교 전자통신컴퓨터공학과 영상통신 및 신호처리 연구실(kimjj79@ece.hanyang.ac.kr)
논문번호 : KICS2007-09-421 접수일자 : 2007년 9월 14일, 최종논문접수일자 : 2007년 12월 6일

Ⅰ. 서 론

MPEG-1,2,4, H.264와 같은 영상압축표준에서 사용되는 부호화 기술은 크게 움직임 추정 및 보상(Motion Estimation and Compensation: ME/MC), 이산직교변환(DCT)을 이용한 변환 및 양자화, 엔트로피 부호화로 구성되는 기본구조를 가지고 있다. 영상의 부호화 과정에서 움직임 추정 및 보상부호기의 복잡도 즉, 전체 부호화 시간에 가장 큰 영향을 미친다.

움직임 추정 및 보상 기법은 비디오영상이 가지는 특성, 즉 영상을 구성하는 성분간의 높은 공간적, 시간적 중복성을 이용하여 데이터를 압축하는 기술이다. 특히 동영상에서 가장 많은 데이터 중복성을 가지고 있는 시간적 데이터 중복성은 참조 프레임의 데이터를 이용하여 움직임 추정과 움직임 보상을 수행하고 이때 추정된 움직임 벡터(Motion Vector: MV)에 의해서 보상된 영상과 원영상과의 차이를 부호화하여 전송 하는 방법으로 동영상 부호화에서 가장 많은 압축률을 가져온다. 움직임 추정 방법은 크게 화소 순환 알고리듬(Pel Recursive Algorithm: PRA)과 블록 정합 알고리듬(Block Matching Algorithm: BMA) 으로 나누어진다. 현재 많은 비디오 부호화 표준에서는 데이터흐름의 규칙성, 계산의 복잡도, 하드웨어 구현을 고려하여 블록 정합 기법을 움직임 추정 및 보상 부호화에서 널리 사용하고 있다. 다시 정리하면 한 화면을 16×16크기의 매크로블록(MB) 또는 임의의 크기 블록으로 나누어 블록 정합 알고리듬(Block Matching Algorithm)을 사용하여 블록단위로 현재 블록을 기준으로 하는 정해진 탐색 영역(Search Window)안에서 정합블록을 찾아낸 다음 찾은 블록 위치를 나타내는 움직임 벡터와 현재 블록과의 차이 값을 부호화하는 것을 말한다. 일반적으로 영상 코덱에서 사용 되는 Full Search 기법은 최적의 정합블록을 찾기 위해 탐색 영역 내의 모든 위치에서 SAD 값을 비교하기 때문에 이러한 움직임 추정 및 보상과정은 전체 부호화 시간에서 가장 많은 부분은 차지하게 된다. 이런 문제를 개선하기 위하여 복잡도를 최소화하기 위한 다양한 고속알고리듬이 개발되었다.

대표적으로 3단계 탐색기법(TSS:Three Step Search)[5], 새로운 3단계 탐색 기법(NTSS:New Three Step Search)[6], 4단계 탐색기법(FSS:Four Step Search)[7], BBGDS(Block-based Gradient Search)[14], 다이아몬드 탐색 기법(DS:Diamond Search)[12] 등과 같이 다양한 형태의 탐색 패턴을 활용하여 초기탐색 지점을 원점(0,0)으로 시작해서 최적의 정합 블록을 찾아내는 방법들이 개발되었다. 하지만 대부분의 고속 블록 정합 알고리듬들이 현재 블록과 주변 블록의 공간적 상관성을 고려하지 않고 일정한 방법으로 블록 정합 오차를 계산함으로서 최적의 블록 정합 오차 지점을 찾아내기 위한 계산량이 많아진다는 단점을 가지고 있다. 예를 들어 3단계 탐색 기법과 같은 경우 최적의 블록 정합 오차 지점이 원점(0,0) 근처에 존재할 때에는 비효율적이라 할 수 있으며, BBGDS 기법도 최적의 블록 정합 오차 지점이 원점(0,0)에서 멀리 떨어진 곳에 존재할 때에는 비효율적이라 할 수 있다.

그래서 본 논문에서 기존의 고속 움직임 추정 방법과 새롭게 제안하는 고속 움직임 추정 방법들을 적응적으로 선택하여 사용함으로서 다른 고속 움직임 추정 기법과 비교하여 비슷하거나 좋은 화질을 가지면서 더 적은 탐색점의 수로 움직임 추정을 가능하게 한다. 본 논문의 구성은 다음과 같다. Ⅱ장에서 움직임 추정 기법의 개념에 대해 설명하고 Ⅲ장에서는 기존의 고속 움직임 탐색 기법과 각 기법들에 대한 분석결과 대해서 소개한다. 그리고 Ⅳ장에서는 현재 블록과 주변 블록의 공간적 상관성을 이용하여 기존의 움직임 추정 기법과 새롭게 제안하는 고속 움직임 추정 기법들을 적응적으로 선택하여 탐색하는 방법에 대하여 설명한다. Ⅴ장에서 기존 기법들과 성능을 비교 평가한 실험결과를 보여주고 마지막으로 Ⅵ장에서 결론을 맺는다.

Ⅱ. 움직임 추정 예측 방법

2.1 움직임 추정

비디오 영상은 연속한 프레임간의 높은 상관성으로 인하여 많은 시간적 중복성을 지니고 있다. 따라서 연속된 프레임간의 움직임 정보를 찾고, 움직임 정보를 전송한다면 각각의 프레임내의 압축뿐만 아니라, 시간적 중복성을 제거하여 고압축을 실현할 수 있다. 동영상 부호화기에서 움직임 예측, 움직임 보상부분이 바로 이러한 움직임 추정을 수행하는 부분이다. 본 논문에서는 움직임 예측 과정에 있어서의 영상의 이동 변위를 나타내는 움직임 벡터를 탐색하는 방법인 화소 반복법과 블록 정합 방법 중에 가장 일반적으로 쓰이는 방법인 블록 정합 방법에 대해 알아본다.

2.2 블록 정합 방법

화면간의 시간 축 중복 데이터의 제거를 목적으

로 하는 예측 부호화기에서 보다 정확한 화면간 예측을 통해 부호화 효율을 높이기 위해 움직임 보상과 움직임 예측 알고리듬을 사용하며, 보통 매크로 블록당 하나의 움직임 벡터를 사용한다.

블록 정합 방법은 프레임을 일정한 크기의 블록(block)으로 나누고 현재 부호화 하고자 하는 블록과 가장 유사한 블록을 이전 프레임에서 찾아 그 블록으로 현재의 블록을 추정하는 방법이다. M×N 크기의 블록에 대해 프레임 당 최대 움직임 속도를 프레임당 P화소라 하면 상하, 좌우로 P화소만큼 움직임이 가능하므로 데이터 검색영역의 크기는 (2P+M)×(2P+N)이 된다. 이 때, 블록간의 일치 정도를 평가하는 값으로서 계산이 간편하고 하드웨어 구현이 용이한 MSE(Mean Square Error) 방법이나 MAD(Mean Absolute Difference), SAD(Sum of Absolute Difference) 방법이 주로 사용된다. 다음의 그림 1은 블록정합 방법을 이용한 움직임 벡터 검출에 대해 나타낸다.

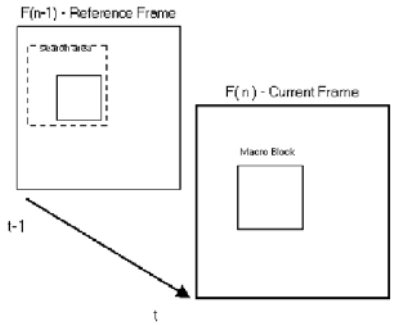


그림 1. 블록 정합 방법

2.3 전역 탐색 기법(Full Search)

블록 움직임 추정 방법 중 탐색 영역 내에 모든 후보 블록과의 차이를 비교하여 가장 유사한 블록을 찾는 전역 탐색 블록 정합 방법은 예측 효율과 정확도 측면에서 가장 많이 사용 되고 있는 기법이다.

이전 프레임을 중심으로 현재 프레임을 추정하기 위해서는 이전 프레임 내에서 탐색 범위를 정하고, 이전 프레임 내 전역을 탐색하여 최소 블록 정합 오차값을 갖는 블록의 좌표를 찾아내어 움직임 벡터로 결정하는 방법이 전역 탐색 블록 정합 방법이라 할 수 있다.

전역 탐색 블록 정합 기법은 탐색 영역 내의 모든 블록에 대해서 정합을 행하는 방식으로, 움직임 벡터를 정확하게 찾을 수 있으나 많은 계산량을 필요로 한다. 이러한 이유로 부분 탐색 방법이나, 부분 정합 등을 이용하여 계산량을 감소시키면서 전역 탐색 블록 정합 방법과 유사한 화질을 갖는 고속 움직임 추정 방법이 많이 연구 되고 있다.

Ⅲ. 고속 움직임 벡터 탐색 방법

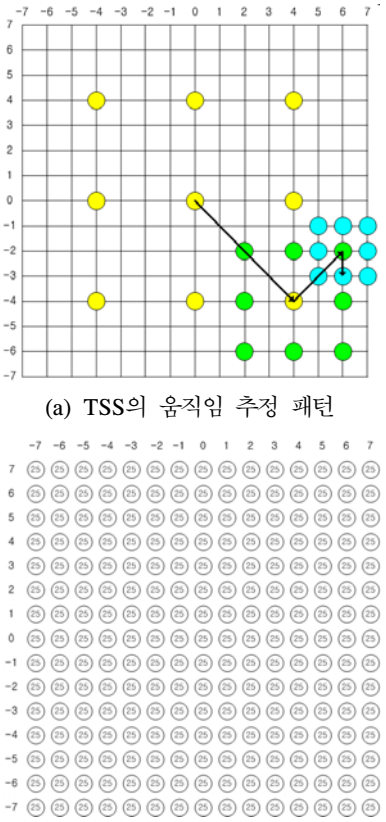
전역 탐색 기법의 단점인 속도문제를 개선하기 위하여 고속 움직임 벡터 탐색 방법 들이 제안되었다. 고속 움직임 벡터 탐색 알고리듬은 다음과 같은 몇 가지 가정을 바탕으로 한다.

첫째, Global Minimum가정이다. 탐색 영역 안의 절대 최소값은 하나뿐이라는 가정이다. 이것은 고속 알고리듬에서 가장 중요한 기초이다.

둘째, Non-Zero Gradient가정이다. 이 가정은 탐색점(Checking Point)이 최소 SAD값을 가지는 점으로 바깥쪽으로 떨어져 있을수록 SAD값이 선형적으로 증가한다고 가정한다.

3.1 기존의 고속 탐색 알고리듬

3.1.1 3단계 탐색 기법(TSS: Three Step Search)



(a) TSS의 움직임 추정 패턴

(b) 각 픽셀의 최소 탐색점의 수

그림 2. 3단계 탐색 기법그림

2(a)는 3단계 탐색 기법의 추정 패턴을 보여준다. 그림에서 보듯이 넓은 탐색 간격의 초기 패턴으로부터 시작해서 탐색 간격을 1/2간격으로 좁히면서 3번의 단계를 거쳐서 움직임 벡터를 결정하게 된다. 다음은 3단계 탐색 기법의 탐색 단계를 보여준다.

1단계) 원점을 중심으로 원점과 함께 4픽셀 떨어진 곳의 8개 점에 대해 정합을 실시하여 최소 블록 정합 오차를 가지는 지점을 결정한다.

2단계) 1단계에서 구한 최소 블록 정합 지점을 중심으로 1단계 간격의 반인 2 픽셀 떨어진 8개의 지점을 1단계에서와 마찬가지로 검사하여 최소 블록 정합 오차 지점을 결정한다.

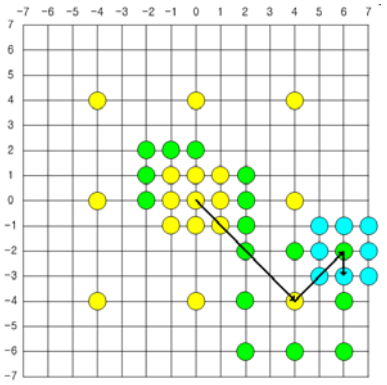
3단계) 마찬가지로 2단계에서 얻은 최소 블록 정합 오차 지점을 중심으로 1픽셀 떨어진 8개의 지점을 검사하여 최소 블록 정합 오차 값을 갖는 지점을 움직임 벡터로 한다.

그림 2(b)는 ±7의 픽셀 탐색 범위에서의 3단계 탐색 기법을 사용할 때 각 픽셀의 최소 탐색점의 수를 보여주며, 모든 픽셀에서 필요한 탐색점의 수는 25임을 알 수 있다. 이것은 움직임이 적은 영상에서는 비효율적이지만 움직임이 많은 영상에서는 효율적임을 보여준다.

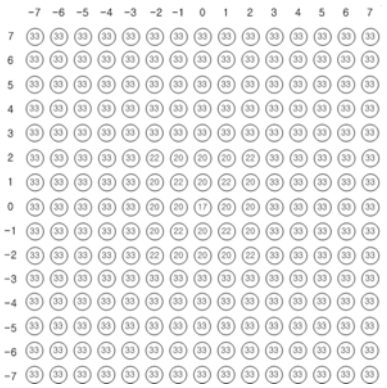
3.1.2 새로운 3단계 탐색 기법(NTSS: New Three Step Search)

새로운 3단계 탐색 기법(NTSS: New Three Step Search)은 원점을 중심으로 한 탐색 패턴(Center Based Search Pattern)으로 이루어진 기법이다. 영상에 있어서 대부분의 블록들은 움직임이 없거나 또한 움직임이 적은 정적인 이미지(Static Image)에서는 움직임 벡터가 원점을 중심으로 집중되어 있다는 사실을 근거로 하여 3단계 탐색 기법에서 원점을 중심으로 8개의 탐색 지점을 추가하여 변형시킨 것으로 볼 수 있다.

그림 3(a)에서 보듯이 첫 단계에서 17개 점의 블록 정합 오차를 구한다. 만약 원점이 최소 블록 정합 오차를 지닐 경우 원점을 움직임 벡터로 지정하고 탐색을 종료한다. 만약 원점과 인접한 8개의 지점이 최소 블록 정합 오차를 지닐 경우에는 3점 혹은 5점을 확장해서 탐색 하게 된다. 만약 원점을 중심으로 대각 방향의 지점이 선택될 경우 동일 방향으로 1픽셀씩 확장하여 5개의 점을 추가로 확장한다. 그리고 만약 직각 방향일 경우는 3점을 확장하게 된다. 그리고 만약 원점과 떨어진 가장자리의 8개의 점이 선택된 경우 3단계 탐색 기법에서와 동일한 방식으로 검색을 수행한다.



(a) NTSS의 움직임 추정 패턴



(b) 각 픽셀의 최소 탐색점의 수

그림 3. 새로운 3단계 탐색 기법

그리고 그림 3(b)는 ±7의 픽셀 탐색 범위에서의 새로운 3단계 탐색 기법 각 픽셀의 최소 탐색점의 수를 보여주며 새로운 3단계 탐색 기법은 3단계 탐색 기법에 비해 움직임이 적은 영상에서는 효율적이지만 움직임이 많은 영상에서는 3단계 탐색 기법보다 비효율적임을 알 수 있다.

3.1.3 4단계 탐색 기법(FSS: Four Step Search)

그림 4(a)에 타나 있는 4단계 탐색 기법(FSS: Four Step Search)은 3단계 탐색 기법에서 흔히 발생하는 국부 최소점(local minimum point)에 빠지는 단점을 보다 개선시킨 방법으로 볼 수 있다. 다음은 4단계 탐색 기법의 탐색 방법에 대해 설명한다.

1단계) 원점을 중심으로 직각, 대각 방향으로 2픽셀 떨어진 8개의 탐색점을 검사하고 원점이 최소 블록 정합 오차를 지닐 경우 4단계로 가고 그렇지 않을 경우 2단계 탐색을 시작한다.

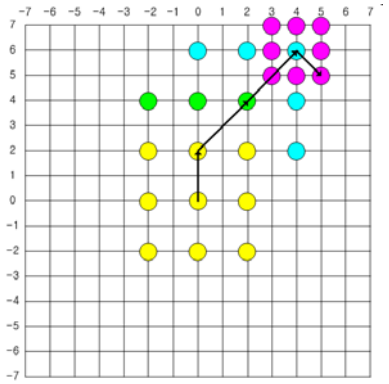
2단계) 1단계에서 원점 이 외의 값이 최소 블록 정합 오차를 지닐 경우 대각선 방향일 경우는 5점, 그리고 직각 방향일 경우는 3점을 더 확장해서 검사하게 된다.

만약 최소 블록 정합 오차 지점의 변화가 없을 경우 3단계로 넘어가고 그렇지 않을 경우 4단계로 넘어간다.

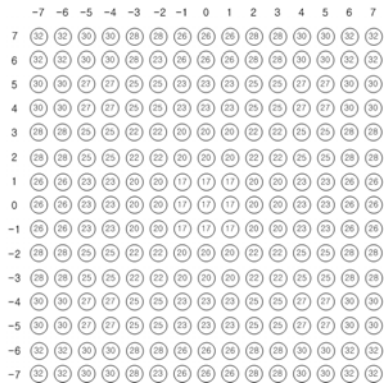
3단계) 2단계에서와 동일한 방식으로 3점 혹은 5점의 최소 블록 정합 오차를 계산한다. 그리고 선택된 최소 블록 정합 오차 지점에 대해 4단계 탐색을 수행한다.

4단계) 3단계에서 얻어진 최소 블록 정합 오차 지점을 중심으로 1픽셀씩 떨어진 대각, 직각 방향의 8개의 점을 더 탐색하고 최소 블록 정합 오차 지점의 이동 변위를 움직임 벡터로 결정하고 탐색을 마친다.

그리고 그림 4(b)는 ±7의 픽셀 탐색 범위에서의 4단계 탐색 기법 각 픽셀의 최소 탐색점의 수를 보여주며 4단계 탐색 기법 또한 새로운 3단계 탐색 기법과 마찬가지로 3단계 탐색 기법에 비해 움직임이 적은 영상에서는 효율적이지만 움직임이 많은 영상에서는 새로운 3단계 탐색 기법보다 비효율적임을 알 수 있다.



(a) FSS의 움직임 추정 패턴



(b) 각 픽셀의 최소 탐색점의 수

그림 4. 4단계 탐색 기법

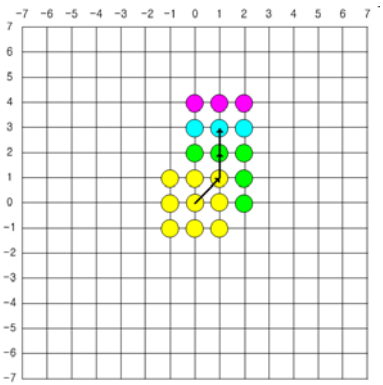
3.1.4 BBGDS (BBGDS: Block-based Gradient Search)

그림 5(a)에서는 BBGDS(Block Based Gradient Descent Search)를 나타낸다. BBGDS기법의 경우 3단계 탐색 기법, 4단계 탐색 기법보다 영상의 움직임 특성이 원점에 근접한 성질을 이용한 기법이다. 다음은 BBGDS 기법의 움직임 추정 단계를 나타낸다.

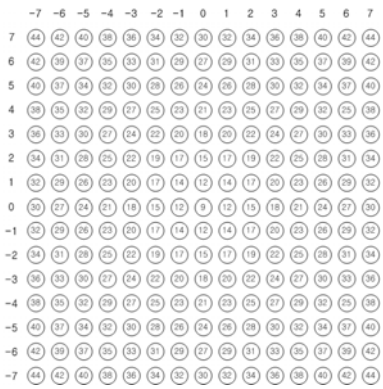
1단계) 원점을 중심으로 9개의 점을 탐색하고 원점이 최소 블록 정합 오차를 가질 경우 탐색을 종료한다. 원점이 아닐 경우 2단계 탐색을 수행한다.

2단계) 만약 원점에서 대각 방향의 지점이 최소 블록 정합 오차 지점일 경우 같은 방향으로 1픽셀씩 확장해서 5개의 점을 탐색하고 직각 방향일 경우 3점을 더 확장해서 탐색을 수행한다. 그리고 최소값의 변화가 없을 때까지 2단계를 반복 실행한다.

그리고 그림 5(b)는 ±7의 픽셀 탐색 범위에서의 BBGDS 탐색 기법 각 픽셀의 최소 탐색점의 수를 보여주며 BBGDS가 움직임이 느린 영상에서는 효율적이지만 움직임이 빠른 영상에서는 비효율적임을 보여준다.



(a) BBGDS의 움직임 추정 패턴



(b) 각 픽셀의 최소 탐색점의 수

그림 5. BBGDS 탐색 기법

3.1.5 다이아몬드 탐색 기법(DS: Diamond Search)

다이아몬드 탐색 기법은 움직임 벡터의 분포가 탐색 영역의 중심인 원점 부근에 치우쳐 존재한다는 가정을 이용한 기법으로 영상의 움직임이 크고 작은 영상 모두에 있어 기존의 기법들에 비해서 화질과 속도 면에 있어서 가장 좋은 성능을 보인다. 다이아몬드 탐색 기법은 그림 6(a)와 같은 탐색 패턴을 이용하여 다음에서 설명하는 단계를 수행하여 움직임 벡터를 추정한다.

1단계) 탐색 영역의 원점을 중심으로 직각, 대각 방향으로 LDSP(Large Diamond Search Pattern)로 8개의 점들을 배치하고 각각의 점들에 대해 블록 정합을 수행하여, 최소 블록 정합 오차 지점을 찾는다. 만약 최소 블록 정합 오차를 가진 점의 위치가 원점일 경우 3단계로 넘어가고 원점이 아닐 경우 2단계로 넘어간다.

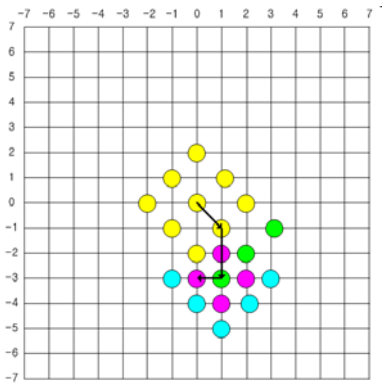
2단계) 1단계에서의 최소 블록 정합 오차 지점을 중심으로 하여 직각 방향의 지점인 경우 5개의 점을 더 확장해서 검사하게 된다. 그리고 만약 대각 방향의 지점인 경우는 3개의 점을 더 확장해서 검사하게 된다. 그리고 최소 블록 정합 오차의 변화가 있을 경우 계속 해서 2단계를 수행하고 최소 블록 정합 오차가 변화가 없을 경우 3단계로 넘어간다.

3단계) 최소 블록 정합 오차 지점을 중심으로 하여 SDSP(Small Diamond Search Pattern)을 수행한 후 최소 블록 정합 오차 지점을 구하여 움직임 벡터를 정한다.

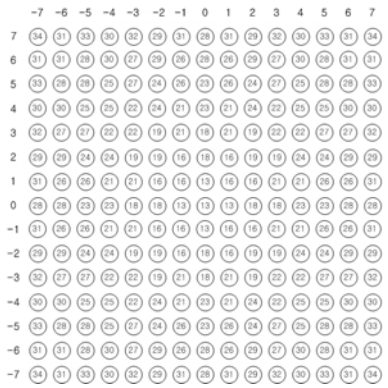
그리고 그림 6(b)는 ±7의 픽셀 탐색 범위에서의 다이아몬드 탐색 기법은 각 픽셀의 최소 탐색점의 수를 보여주며 다이아몬드 탐색 기법은 움직임이 느린 영상에서 BBGDS보다 움직임이 빠른 영상에서는 3단계 탐색 기법보다 비효율적이지만 느리지도 빠르지도 않은 영상에서는 다른 고속 탐색 알고리듬보다 성능이 뛰어남을 알 수 있다.

3.2 픽셀당 최소 탐색점의 수를 고려한 분석

그림 7(a)는 ±7의 픽셀 탐색 범위에서의 3단계 탐색 기법, 새로운 3단계 탐색 기법, 4단계 탐색 기

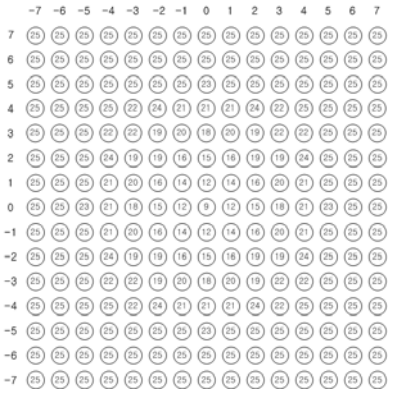


(a) DS의 움직임 추정 패턴

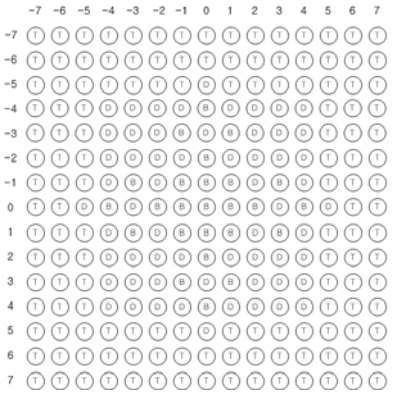


(b) 각 픽셀의 최소 탐색점의 수

그림 6. 다이아몬드 단계 탐색 기법



(a) 각 픽셀의 최소 탐색점의 수.



(b) 각 픽셀에서의 고속 탐색 기법

그림 7. 여러 고속 탐색 기법들을 고려한 분석

법, BBGDS 그리고 다이아몬드 탐색 기법 모두 고려해 보았을 때, 픽셀당 최소의 탐색점의 수를 보여주며 그림 7(b)는 각 픽셀에서 최소 탐색점의 수을 가지게 하는 고속 탐색 알고리듬의 종류를 보여준다. 그림 7(a), (b)에서는 픽셀당 최소 탐색점의 수를 고려해 볼 때 움직임이 적거나 없는 영상 혹은 블록에서는 BBGDS가 효율적이고 MV(Motion Vector)의 길이가 3와 4 그리고 5중에 MV가 결정되는 영상이나 블록에 전반적으로 다이아몬드 탐색 기법이 유리함을 논증하다. 그리고 3단계 탐색 기법이 움직임이 많은 영상이나 블록에서는 효율적임을 알 수 있다. 한 프레임을 고려해 볼 때 움직임이 적은 블록, 많은 블록, MV의 길이가 3 ~ 5인 블록은 각기 존재한다. 따라서 한 프레임을 움직임 추정할 때에는 현재 블록의 움직임의 느리고 빠름 정도를 예측하여 BBGDS, 다이아몬드 탐색 기법 그리고 3단계 탐색 기법 고속 탐색 알고리듬을 선택하여 사용하는 것이 효율적임을 알 수 있다.

Ⅳ. 제안하는 알고리듬

제안하는 알고리듬은 한 프레임에서 각각 블록의 움직임 추정할 때에는 현재 블록의 움직임의 느리고 빠름 정도를 미리 예측하여 효율적인 고속 탐색 알고리듬을 적응적으로 선택하여 사용하는 것이다. 현재 블록이 움직임이 적은 경우라고 판단하면 BBGDS 탐색 기법을 대신해 새롭게 제안하는 향상된 십자형 다이아몬드 탐색 기법(UCDS)을 이용하고 움직임이 많은 경우라고 판단되면 십자형 3단계 탐색 기법(CTSS)을 이용한다. 그리고 나머지의 경우 다이아몬드 탐색 기법을 이용한다.

4.1 The UCDS(Upgraded Cross Diamond Search)

픽셀당 최소 탐색점의 수를 고려한 분석에서 볼 수 있듯이 BBGDS는 느린 영상이나 블록에서 탐색점의 수가 줄어드는 경향을 볼 수 있다. 하지만 다이아몬드 탐색 기법을 개선시켜 성능을 향상 시킨 십자형 다이아몬드 탐색 기법(CDS: Cross Diamond Search Algorjthm)[8]이나 새로운 십자형 다이아몬드 탐색 기법(NCDS: New Cross Diamond Search Algorithm)[11]이 움직임이 느린 영상이나 블록에 더욱 유용하다. 왜냐하면 CDS와 NCDS는 다이아몬드 탐색 기법의 경우 LDSP 수행 후 SDSP를 한번 더 수행해 최소 13개의 탐색 지점을 필요로 한다는 점을 착안하여 CDS는 초기 탐색 지점을 십자 형태의 9개점을 수행하고 NCDS는 초기 탐색을 원점을 중심으로 SDSP 5개 지점을 수행하여 탐색점의 수를 줄이기 때문이다. 그러나 실험 동영상들에 대해 탐색영역의 거리를 ±7로 두고 움직임 벡터의 분포를 조사해 보았을 때, 표 2와 같이 대부분의 움직임 벡터가 탐색영역의 중심 주위에 분포하고 탐색영역의 중심점으로부터 반경 1픽셀 내의 약 62%의 가장 많은 분포를 가지고 반경 2픽셀 내에 약 76%, 반경 3픽셀 내에는 약 84% 분포를 가짐을 알 수 있다. 십자형 다이아몬드 탐색 기법(CDS: Cross Diamond Search Algorjthm)의 경우 초기에 십자형태의 9개의 점을 수행하고 새로운 십자형 다이아몬드 탐색 기법(NCDS: New Cross Diamond Search Algorjthm)의 경우 5개의 초기 탐색 점 수행함으로서 움직임 없거나 움직임이 적은 영상 즉 최적의 블록 정합 오차지점이 반경 1픽셀 내에 분포하고 있는 움직임 벡터 탐색에서는 탁월한 성능을 보인다. 하지만 십자형 다이아몬드 탐색 기법과 새로운 십자형 다이아몬드 탐색 기법은 반경 2, 3 픽셀에도 많은 양의 움직임 벡터가 존재함에도 불구하고 반경 2, 3픽셀 분포하고 있는 움직임 벡터에 대해서는 고려하지 않음으로서 블록 정합 오차 계산에서의 성능 열화를 가져온다는 점을 알 수 있다. 그래서 제안된 알고리듬에서는 향상된 십자형 다이아몬드 탐색 기법(UCDS: Upgraded Cross Diamond Search Algorithm)이라는 새로운 탐색 기법을 사용하여 반경 2, 3픽셀 내에 분포하고 있는 움직임 벡터를 고려하여 블록 정합 오차를 실행함으로서 성능 향상을 가져올 것이다.

표 1. 분석을 위한 실험 동영상

Format	Sequence
CIF (352 X 288)	flower, foreman, hall, mother & daughter, news, football, coastguard

4.2 Performance of The UCDS Algorithm

UCDS는 초기탐색을 원점(0,0)을 중심으로 SDSP를 수행함으로서 십자 다이아몬드 탐색 기법은 갖는 9개의 초기 탐색 점으로 인한 속도 저하를 방지하고 SDSP 수행 후 납작한 다이아몬드 형태로 4개의 점을 추가함으로서 새로운 십자 다이아몬드 탐색 기법보다 빨리 블록 정합 오차가 최소인 지점을 찾을 수 있다.

표 2. 움직임 벡터 분포

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
-7	0.030	0.016	0.009	0.012	0.010	0.013	0.014	0.070	0.014	0.011	0.011	0.009	0.007	0.014	0.026
-6	0.015	0.011	0.008	0.008	0.006	0.011	0.012	0.053	0.014	0.008	0.010	0.003	0.007	0.006	0.014
-5	0.019	0.012	0.011	0.015	0.009	0.014	0.016	0.126	0.014	0.014	0.014	0.011	0.009	0.007	0.017
-4	0.032	0.013	0.020	0.015	0.021	0.023	0.127	0.155	0.023	0.019	0.018	0.013	0.010	0.007	0.018
-3	0.050	0.024	0.027	0.022	0.027	0.078	0.155	0.549	0.136	0.086	0.024	0.017	0.014	0.016	0.037
-2	0.061	0.028	0.027	0.022	0.052	0.241	0.487	1.554	0.507	0.133	0.054	0.040	0.022	0.019	0.061
-1	0.125	0.045	0.050	0.062	0.112	0.575	1.864	3.623	1.431	0.607	0.169	0.136	0.101	0.084	0.196
0	0.731	0.258	0.408	0.412	1.042	1.727	3.785	45.724	5.306	4.274	1.662	1.519	1.275	0.247	0.779
1	0.227	0.077	0.087	0.104	0.138	0.539	1.439	3.284	1.169	0.605	0.109	0.057	0.045	0.036	0.142
2	0.107	0.049	0.048	0.060	0.056	0.175	0.623	1.471	0.586	0.171	0.051	0.029	0.029	0.030	0.071
3	0.022	0.010	0.013	0.017	0.029	0.114	0.138	0.502	0.143	0.126	0.038	0.017	0.020	0.019	0.058
4	0.012	0.009	0.010	0.013	0.012	0.073	0.089	0.229	0.018	0.067	0.017	0.023	0.011	0.010	0.034
5	0.016	0.008	0.011	0.013	0.011	0.014	0.031	0.141	0.019	0.034	0.008	0.010	0.009	0.009	0.026
6	0.017	0.008	0.010	0.011	0.011	0.020	0.240	0.074	0.040	0.027	0.005	0.008	0.006	0.008	0.026
7	0.039	0.010	0.011	0.009	0.018	0.018	0.055	0.373	0.040	0.047	0.016	0.010	0.014	0.014	0.071

특히 움직임이 없거나 적은 영상에 있어서 다른 알고리듬 보다 탁월한 성능을 발휘함을 볼 수 있다. 다음은 UCDS의 움직임 추정 단계를 보여준다.

1단계) 원점(0,0)을 중심으로 한 SCSP(small cross search pattern)의 5개의 점에서 최적의 블록을 찾고 만약 그림 8와 같이 중심에서 최적의 블록이 발생하면 탐색을 멈추고 아니면 다음 단계로 넘어간다.

2단계) 만약 최적의 블록이 (1,0)(혹은 (-1,0), (0,1), (0,-1)) 발생하면 그림 9과 같이 추가된 4개의 점((2,0), (3,0), (1,-1), (1,1))을 탐색하고 최적의 블록이 (1,0)(혹은(-1,0), (0,1), (0-1))에서 발생하면 탐색을 멈추고 아니면 3단계로 넘어간다.

3단계) 3단계에는 3가지의 경우로 나눌 수 있다.

경우 1 : 만약 최적의 블록이 (2,0)(혹은 (-2,0), (0,2), (0,-2))에서 발생하면 그림 10과 같이 2개의 점((2,-1), (2,1))을 추가적으로 탐색하고 (2,0)(혹은 (-2,0), (0,2), (0,-2))에서 최적의 블록이 나타나면 탐색을 멈추고 그렇지 않으면 4단계로 넘어간다.

경우 2 : 만약 최적의 블록이 (1,-1)(혹은 (-1,-1), (-1,1), (1,1))에서 발생하면 그림 11와 같이 2개의 점((1,-2), (2,-1))을 추가적으로 탐색하고 (1,-1)(혹은 (-1,-1), (-1,1), (1,1))에서 최적의 블록이 발생하면 탐색을 멈추고 아니면 4단계로 넘어간다.

경우 3 : 만약 최적의 블록이 (3,0)(혹은 (-3,0), (0,3), (0,-3))에서 발생하면 4단계로 넘어간다.

4단계) 이전 단계의 최적의 블록점을 중심으로 한 LDSP(large diamond search pattern) 실행하고 중심에서 최적의 블록점을 찾으면 5단계로 넘어가고 아니면 4단계를 반복 수행한다.

5단계) 이전 단계의 최적의 블록점을 중심으로 한 SDSP(small diamond search pattern)을 수행함.

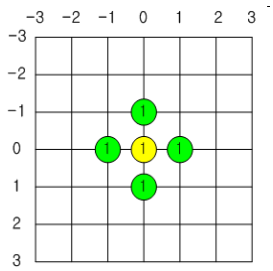


그림 8. 1단계에서 탐색을 멈추는 경우 - MV(0,0)

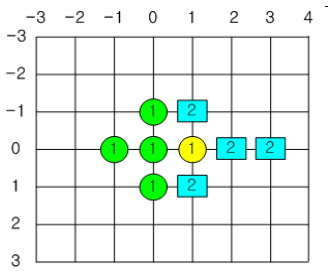


그림 9. 2단계에서 탐색을 멈추는 경우 - MV(1,0)

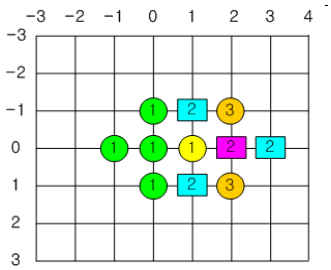


그림 10. 3단계 경우1에서 탐색을 멈추는 경우 - MV(2,0)

그림 12과 13는 전 단계를 수행한 예를 나타내고 있다. 그림 12는 1단계에서 원점(0,0)을 중심으로 4개의 점을 비교하여 (1,0)이 선택되고 2단계에서 4개의 점을 추가해 블록 정합 오차를 계산한 후 (3,0)이 선택되면 3단계의 경우 3임으로 4단계로 넘어가서 LDSP 수행하고 최적의 블록점이 중심에서 나타나면 5단계의 SDSP를 수행하여 (3,0)을 결정한다. 그리고 그림 13는 마찬가지 방법으로 1단계에서 원점(0,0)을 중심으로 4개의 점을 비교하여 (1,0)을 선택하고 2단계에서 4개의 점을 추가해 비교하여 (1,-1)결정한 후 2개((1,-2),(2,-1))의 점을 추가해 비교하는 3단계의 경우 2 수행해 (2,-1)를 선택하고 최적의 블록점을 중심으로 LDSP를 수행하는 4단계를 한 다음 5단계의 SDSP를 수행하여 최종적으로 (3,-2)를 결정되는 과정을 보여주고 있다.

실제로 MPEG-4 VM 인코더에서 CIF 영상 100 프레임을 움직임이 적은 영상과 많은 영상을 선택하여 실험한 결과 표 3를 보면 움직임이 작은 영상에서는 UCDS의 PSNR가 BBGDS, 십자형 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법과 비슷하게 유지되면서 UCDS의 블록당 탐색점의 수(Search Point : SP)가 다른 알고리듬보다 적음을 알 수 있다. 하지만 움직임이 많은 영상에서는 다이아몬드 탐색 기법 비하여 PSNR는 떨어짐을 알 수 있다. 그러므로 원점을 중심으로 하여 일정한 영역에 MV가 존재하는 움직임이 적은 영상을 탐색할 경우 UCDS를 사용하는 것이 유리하다는 결론을 얻을 수 있다.

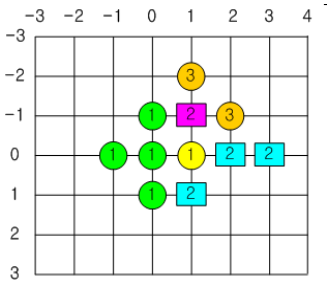


그림 11. 3단계 경우2에서 탐색을 멈추는 경우 - MV(1,-1)

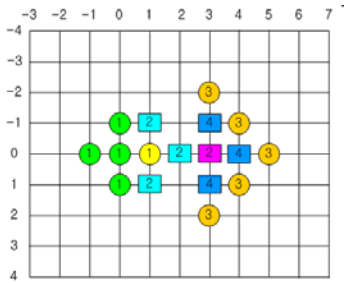


그림 12. MV(3,0)가 결정되는 경우

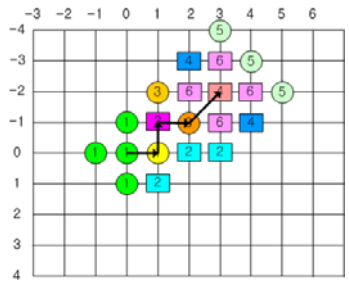


그림 13. MV(3,-2)가 결정되는 경우

표 3. CIF 영상 100 프레임의 PSNR과 Search Point

	DS		BBGDS		CDS		NCDS		UCDS	
	PSNR	SP	PSNR	SP	PSNR	SP	PSNR	SP	PSNR	SP
akiyo	36.08	13.17	36.08	9.26	36.08	9.20	36.07	5.32	36.08	5.22
waterfall	30.51	13.11	30.51	9.25	30.51	9.14	30.50	5.33	30.50	5.31
football	33.31	23.50	33.27	21.61	33.27	27.26	33.26	18.81	33.27	18.74
bus	29.30	21.02	29.25	18.20	29.25	21.77	29.24	17.93	29.24	17.04

표 4. CIF 영상 100 프레임의 PSNR과 Search Point

		football	bus	news	coastguard	foreman	silent	stefan	flower
TSS	PSNR	33.29	29.28	33.90	29.71	31.97	32.85	29.56	28.07
	SP	25	25	25	25	25	25	25	25
CTSS	PSNR	33.29	29.27	33.90	29.71	31.98	32.86	29.55	28.07
	SP	21.44	21.53	21.26	21.47	21.39	21.31	21.29	21.13

4.3 The CTSS(Cross Three Step Search)

표 4와 같이 3단계 탐색 기법과 CTSS를 비교해 보면 PSNR은 비슷하게 나오지만 탐색점의 수(SP)는 상대적으로 줄여듬을 알 수 있다. 그래서 제안된 알고리듬에서는 3단계 탐색 기법을 대신해 CTSS를 사용하겠다.

CTSS는 3단계 탐색 기법과 마찬가지로 초기탐색을 원점(0,0)을 중심으로 8개 점에 대해 정합을 수행하고 탐색 간격을 1/2간격으로 좁히면서 2단계, 3단계에서는 십자 형태로 4개의 점을 우선 정합을 실시한 후 최소 블록 정합 오차 지점을 따라 적응적으로 2개의 점을 추가적으로 정합을 실시한다. 다음은 CTSS의 움직임 추정 단계를 보여준다.

1단계) 원점을 중심으로 원점과 함께 4픽셀 떨어진 곳의 8개 점에 대해 정합을 실시하여 최소 블록 정합 오차를 가지는 지점을 결정한다.

2단계) 그림과 같이 1단계에서 구한 최소 블록 정합 지점을 중심으로 1단계 간격의 반인 2 픽셀 떨어진 십자 형태로 4개 점을 우선 정합 실시하고 정합 지점이 변화가 없을 경우 그림 14와 같이 3단계로 넘어가고 정합 지점이 변화가 있을 경우 그림 15와 같이 적응적으로 2개 점을 정합 실시하고 3단계로 넘어간다.

3단계) 마찬가지로 2단계에서 얻은 최소 블록 정합 오차 지점을 중심으로 1픽셀 떨어진 십자 형태로 4개 점을 우선 정합 실시하고 정합 지점이 변화가 없을 경우 그림 14와 같이 움직임 벡터를 결정하고 정합 지점이 변화가 있을 경우 그림 15와 같이 적응적으로 추가적으로 2개 점을 정합 실시하고 움직임 벡터를 결정한다.

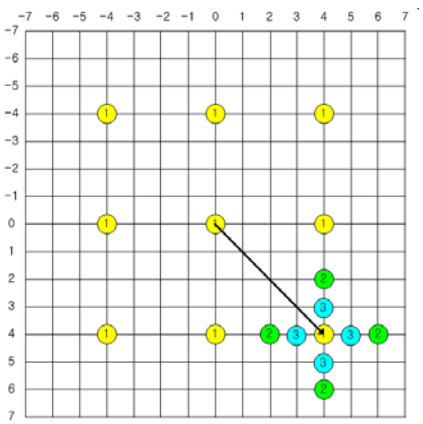


그림 14. MV(4,4)가 결정되는 경우

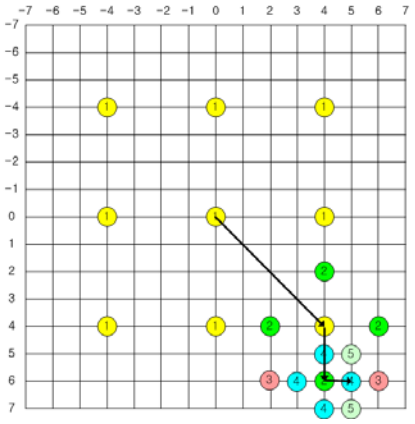


그림 15. MV(5,6)가 결정되는 경우

제안된 알고리듬에서는 계산 복잡도를 낮추고 보다 정확한 최소 블록 정합 오차 지점을 적은 탐색점의 수로 찾아내기 위해, 영상이 가지는 특성상 현재 블록과 이웃하는 블록의 공간적 상관성이 높다는 점을 이용한다.

4.4 현재 블록의 움직임 정도 예측 방법

현재 블록의 움직임 추정하기 위한 주변 블록에서 이미 구해진 움직임 벡터의 길이와 탐색점의 수 그리고 SAD값을 이용하여 현재 블록의 움직임의 정도 판단하여 적응적으로 알고리듬을 사용한다. 다시 말해서 그림 16에서와 같이 주변 블록(left, above, above right)에서 이미 결정된 최적의 블록 정합 오차 지점, 탐색점의 수, SAD 값을 이용해 다음 식(1),(2),(3),(4)에 적용시켜 어떤 알고리듬을 선택할지의 판단기준이 되는 MV(Motion Vector)의 길이, 탐색점의 수, SAD 값을 결정한다. 하지만 현재블록이 가장 왼쪽, 위에 위치하면 판단 기준들을 가장 최소값으로 설정하고 현재블록이 가장 위에 위치할 경우 1개의 주변블록(left)만을 이용하고 가장 아래 오른쪽에 위치할 경우 2개의 주변블록(left, above)만을 이용한다.

	MV_{above}	$MV_{above\ right}$
MV_{left}	$MV_{current}$	

그림 16. 현재블록과 주변블록

$$MV_{max} = \max\ (MV_x, MV_y) \quad (1)$$

$$MV_{length} = (\text{int})\{ averge (MV_{left}, MV_{above}, MV_{abvoe-right}) \} \quad (2)$$

$$SP_{current} = (\text{int})\{ averge (SP_{left}, SP_{above}, SP_{abvoe-right}) \} \quad (3)$$

$$SAD_{current} = (\text{int})\{ averge (SAD_{left}, SAD_{above}, SAD_{abvoe-right}) \} \quad (4)$$

그리고 3개의 판단 기준을 각각 2개의 Threshold을 이용해 Small, Medium, Large를 판별한다. MV의 길이(MV-len)에 대한 2개의 Threshold는 그림 8에서 보여지는 것과 같이 MV의 길이가 2이하에서는 BBGDS, 6이상에서는 TSS, 그리고 3에서 5사이에서는 DS가 좋음을 알 수 있다. 그러므로 Threshold를 T1=2, T=6로 설정하고 표 5에서 같이 Small, Medium, Large를 결정한다.

탐색점의 수(Search-Point)에 대한 2개의 Threshold는 그림 3.6에서 보여지는 것과 같이 최소 탐색점의 수는 17이하 BBGDS, 25이상에서는 3단계 탐색 기법, 그리고 18에서 24사이에서는 다이아몬드 탐색 기법이 유리함을 알 수 있다. 하지만 본 알고리듬에서는 움직임 벡터를 빨리 찾기 위해 변형된 알고리듬들을 사용함으로 Threshold를 T1=10, T=20로 설정하고 표 6에서와 같이 Small, Medium, Large를 결정한다.

MPEG-4 VM 인코더에서 평균 SAD 값을 대한 2개의 Threshold를 T1=1100, T2=2200할 때 좋은 결과를 얻을 수 있었다. 그러므로 Threshold를 T1=1100, T2=2200로 설정하고 표 7에서 같이 Small, Medium, Large를 결정한다.

표 5. MV-len 판별 방법

MV-len ≤ T1	Small(S)
T1 < MV-len < T2	Medium(M)
MV-len ≥ T2	Large(L)

표 6. Search-Point 판별 방법

Search-Point ≤ T1	Small(S)
T1 < Search-Point < T2	Medium(M)
Search-Point ≥ T2	Large(L)

표 7. SAD 판별 방법

SAD ≤ T1	Small(S)
T1 < SAD < T2	Medium(M)
SAD ≥ T2	Large(L)

MV-len, SP, SAD 3가지의 판별 기준에 따라 현재 블록을 분류할 경우 표 8처럼 27가지의 경우가 나타나는데 3가지 경우로 나눌 수 있다. 그리고 제안된 알고리듬에서는 표 9에서와 같이 3가지 경우에 대하여 적응적으로 사용할 알고리듬을 결정한다.

표 8. 판별된 결과의 분류

SS이 두 개 이상이 나올 경우		SSS, SSM, SMS, MSS, SSL, SLS, LSS
MM이 두 개 이상이 나올 경우	하나가 S인 경우	SMM, MSM, MMS
	하나가 M인 경우	MMM
	하나가 L인 경우	MML, MLM, LMM
S, M, L이 혼합되어 나올 경우		SML, SLM, MSL, MLS, LSM, LSM,
LL이 두 개 이상이 나올 경우		SLL, LSL, LLS, MLL, LML, LLM, LLL

표 9. 판별된 결과로부터 적용될 알고리듬

SS이 두 개 이상이 나올 경우	UCDS(Upgraded Cross Diamod Search)
MM이 두 개 이상이고 하나가 S인 경우	
MM이 두 개 이상이고 하나가 M혹은 L일 경우	DS(Diamond Search)
S, M, L이 혼합되어 나올 경우	
LL이 두 개 이상이 나올 경우	CTSS(Cross Three Step Search)

여기서 만약 현재 블록이 프레임의 첫 번째일 경우 UCDS를 실행한다. 왜냐하면 가장자리는 배경일 가능성이 많기 때문에 움직임이 적은 부분이라고 말 할 수 있다. 그 이후에 블록들은 주변블록들의 움직임 벡터, 탐색점의 수, SAD를 이용하여 움직임 정도를 판단하여 UCDS, DS 그리고 CTSS 알고리듬 중에 가장 적절한 알고리듬을 선택하여 움직임 추정을 수행한다.

Ⅴ. 실험 결과 및 분석

제안된 알고리듬의 성능을 평가하기 위하여 MPEG-4 VM 인코더를 이용하여 이루어졌다. 모든 영상은 IPPP를 부호화 되었고, 비트율 제어(rate control)는 TM5를 적용하였다. 그리고 실험에 사용한 영상은 CIF(352×288) 영상 "football," "akiyo," "bus," "news," "tempete," "waterfall," "bridge," "coastguard," "container," "foreman," "hallway," "mobile," "paris," "silent," "Stefan," "table," 그리고 "flower"들에 대해 각각 100프레임씩을 대상으로 실험하였고, 비교 탐색 알고리듬으로는 3단계 탐색기법, 새로운 3단계 탐색 기법, 4단계 탐색기법, BBGDS, 다이아몬드 탐색 기법, 육각 탐색 기법 (HEXBS: Hexagon-based Search)[13], 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법 그리고 제안된 알고리듬을 사용하였다. CDS, NCDS 그리고 제안된 알고리듬을 사용하였다. 움직임 예측에 사용한 매크로 블록의 크기는 16×16 픽셀이며, 탐색 영역의 변위 ±7을 적용하여 실험을 수행하였다. 성능 비교 평가 함수로는 영상 화질의 품질을 평가하기 위해 평균 제곱 오차(MSE: mean squared error)를 이용한 PSNR(peak signal-to-noise ratio)를 이용하였으며, 정합 오차 측정 함수로는 절대값 오차의 합(SAD: sum of absolute difference)을 이용하였다. 또한 제안하는 알고리듬의 성능 향상을 측정하기 위해 블록 당 탐색점의 수를 기존 알고리듬들과 비교하였다. 탐색 알고리듬의 성능을 비교평가하기 위해 사용한 함수인 MSE와 PSNR은 각각 식(1),(2)와 같다.

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

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \quad (2)$$

식 (1)과 (2)에서 M과 N은 각각 영상의 가로와 세로의 크기를 나타내며, $O_t(x,y)$는 원영상의 화면을 나타내고, $E_t(x,y)$는 움직임 예측 화면을 나타낸다.

블록의 정합오차를 측정하기 위한 함수 SAD는 식(3)과 같다.

$$SAD = \sum_{k=1}^{MB}\sum_{l=1}^{MB}\left|I_t(k,l) - I_{t-1}(k+i, l+j)\right| \quad (3)$$

식 (3)에서 MB는 매크로 블록의 가로와 세로의 크기를 나타내며, $I_t(k,l)$은 현재 프레임을 나타내고, $I_{t-1}(k+i,l+j)$은 이전 프레임을 나타낸다.

실험영상에 대한 실험 결과는 표 10, 표 11 그리고 표 12에 각각 나타내었다. 표 10과 표 11에서는 3단계 탐색기법, 새로운 3단계 탐색 기법, 4단계 탐색기법, BBGDS, 다이아몬드 탐색 기법, 육각 탐색 기법, 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법 그리고 제안한 알고리듬의 평균 PSNR과 MSE를 각각 나타내었고, 표 12에서는 한 블록당 평균 탐색점의 수를 나타내었다.

제안하는 탐색 알고리듬은 표 10과 11에서 볼 수 있듯이 대부분의 영상에 대해서 기존의 고속 탐색 알고리듬과 비슷한 수준의 PSNR과 MSE를 유지하며 특히 다이아몬드 탐색 기법의 성능을 향상시킨 십자형 다이아몬드 탐색 기법, 새로운 십자형 다이아몬드 탐색 기법과 비교하였을 때, "football," "bus," 그리고 "Stefan"과 같이 움직임이 많은 영상에서는 오히려 PSNR과 MSE가 더 좋음을 알 수 있다. 표 12에 나타나 있는 것처럼 탐색점의 수에 관해서는 3단계 탐색 기법과 비교하였을 때는 평균적으로 블록당 16.35의 탐색점의 수를 감소시키고 다이아몬드 탐색 기법과 비교하였을 때는 6.92의 탐색점의 수를 감소시켰다. 그리고 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법과 비교하였을 때에도 평균 4.27에서 0.85까지 탐색점의 수를 감소시키는 것을 볼 수 있다.

Ⅵ. 결 론

본 논문에서는 영상의 압축 효율을 높이는 움직임 추정 기법에 있어서 기존에 제안된 블록 정합 방법만큼 좋은 화질을 유지하면서 보다 고속으로 전송하기 위한 움직임 벡터 추정을 제안하였다.

표 10. 기존 방법들과의 평균 PSNR (dB) 비교

	TSS	DS	NTSS	BBGDS	FSS	HEX	CDS	NCDS	A-UDC
football	33.29	33.31	33.28	33.27	33.28	33.28	33.27	33.26	
akiyo	36.04	36.08	36.09	36.03	36.09	36.01	36.08	36.06	36.08
bus	29.28	29.30	29.24	29.25	29.28	29.28	29.25	29.24	29.28
news	33.90	33.93	33.83	33.88	33.95	33.86	33.88	33.87	33.92
tempete	28.76	28.78	28.74	28.74	28.79	28.76	28.74	28.74	28.76
waterfall	30.51	30.51	30.51	30.50	30.51	30.52	30.50	30.50	30.50
brigdge	35.00	35.01	35.01	35.00	35.00	35.00	35.01	35.00	35.00
coastguard	29.71	29.74	29.71	29.71	29.75	29.74	29.72	29.71	29.72
container	32.28	32.32	32.30	32.28	32.30	32.26	32.31	32.28	32.30
forman	31.97	32.02	31.98	31.95	32.13	31.99	31.97	31.96	32.01
highway	34.89	34.91	34.90	34.92	34.93	34.94	34.90	34.91	34.91
mobile	27.30	27.27	27.25	27.27	27.28	27.26	27.27	27.27	27.28
paris	30.18	30.19	30.13	30.14	30.21	30.23	30.15	30.10	30.17
slient	32.85	32.89	32.86	32.83	32.92	32.89	32.86	32.86	32.89
stefan	29.56	29.61	29.58	29.57	29.60	29.58	29.58	29.57	29.61
table	31.31	31.33	31.26	31.28	31.29	31.27	31.30	31.29	31.29
flower	28.07	28.06	28.07	28.07	28.05	28.08	28.06	28.06	28.07
Average	31.46	31.49	31.45	31.45	31.49	31.47	31.46	31.45	31.47
Ave-proposed	-0.01	0.02	-0.02	-0.02	0.02	0.00	-0.01	-0.02	0.00

표 11. 기존 방법들과의 평균 MSE 비교

	TSS	DS	NTSS	BBGDS	FSS	HEX	CDS	NCDS	A-UDC
football	30.48	30.37	30.54	30.60	30.54	30.58	30.61	30.69	30.44
akiyo	16.18	16.02	16.01	16.22	15.99	16.28	16.02	16.09	16.02
bus	76.98	76.37	77.55	77.34	76.81	76.76	77.26	77.52	76.77
news	26.48	26.32	26.89	26.60	26.18	26.72	26.61	26.66	26.38
tempete	86.52	86.03	86.88	86.93	85.88	86.43	86.91	86.90	86.50
waterfall	57.83	57.76	57.82	57.91	57.88	57.72	57.89	57.95	57.92
brigdge	20.54	20.52	20.54	20.54	20.56	20.55	20.53	20.56	20.56
coastguard	69.45	69.06	69.59	69.55	68.90	69.00	69.30	69.55	69.42
container	38.51	38.13	38.33	38.46	38.33	38.67	38.23	38.50	38.32
forman	41.28	40.83	41.21	41.50	39.86	41.15	41.32	41.38	40.90
highway	21.09	21.01	21.04	20.95	20.92	20.84	21.02	21.00	21.00
mobile	121.17	121.84	122.57	122.05	121.75	122.21	121.93	122.00	121.72
paris	62.36	62.25	63.16	62.98	62.02	61.61	62.88	63.61	62.57
slient	33.71	33.41	33.63	33.92	33.17	33.44	33.68	33.69	33.43
stefan	71.94	71.06	71.66	71.74	71.26	71.61	71.67	71.82	71.13
table	48.14	47.87	48.67	48.39	48.28	48.54	48.22	48.35	48.36
flower	101.47	101.53	101.31	101.41	101.89	101.26	101.62	101.59	101.45
Average	54.35	54.14	54.55	54.53	54.13	54.32	54.45	54.58	54.29
Ave-proposed	0.06	-0.15	0.26	0.24	-0.16	0.03	0.16	0.29	0.00

제안한 움직임 벡터 추정 기법은 현재 블록과 주변 블록의 공간적 상관성을 높다는 것을 착안하여 주변블록들의 움직임 벡터, 탐색점의 수, SAD를 이용하여 움직임 정도를 판단하여 UCDS, DS 그리고 CTSS 알고리듬 중에 가장 적절한 알고리듬을 선택하여 움직임 추정을 수행함으로서 움직임 벡터

표 12. 기존 방법들과의 블록 당 평균 탐색점의 수 비교

	TSS	DS	NTSS	BBGDS	FSS	HEX	CDS	NCDS	A-UDC
football	25.00	23.50	29.26	21.61	23.08	16.44	27.26	18.81	17.80
akiyo	25.00	13.17	17.40	9.26	17.16	11.08	9.20	5.32	5.22
bus	25.00	21.02	29.63	18.20	22.27	15.65	21.77	17.93	16.94
news	25.00	13.34	17.64	9.42	17.32	11.19	9.35	5.61	5.47
tempete	25.00	13.73	18.80	10.15	17.53	11.33	10.14	6.45	6.37
waterfall	25.00	13.11	17.45	9.25	17.09	11.01	9.14	5.33	5.31
brigdge	25.00	13.06	17.23	9.04	17.07	11.04	9.02	5.01	5.01
coastguard	25.00	17.03	23.04	13.84	20.74	13.35	14.45	13.15	10.61
container	25.00	13.11	17.34	9.10	17.12	11.06	9.04	5.10	5.07
forman	25.00	16.34	22.42	13.41	20.23	12.89	13.52	10.80	9.49
highway	25.00	14.20	19.29	10.35	18.15	11.66	10.45	6.70	6.24
mobile	25.00	14.33	21.38	11.13	18.00	11.58	10.76	7.80	7.88
paris	25.00	14.03	18.94	10.28	18.02	11.57	10.34	6.80	6.40
slient	25.00	14.19	18.67	10.60	17.83	11.61	10.87	6.88	6.62
stefan	25.00	17.63	24.42	14.63	20.22	13.74	16.35	13.80	12.68
table	25.00	17.22	23.65	14.42	20.94	13.46	15.91	12.85	11.20
flower	25.00	15.65	22.04	12.08	19.98	12.59	12.01	10.17	8.69
Average	25.00	15.57	21.09	12.16	18.98	12.43	12.92	9.32	8.65
Ave-proposed	16.35	6.92	12.44	3.41	10.33	3.78	4.27	0.85	0.00

를 고속으로 추정할 수 있도록 하였다.

실험 결과에서 보면 알 수 있듯이 제안된 기법은 이전에 알려진 고속 탐색 기법과 비교하여 대부분 영상에서 비슷한 수준의 PSNR과 MSE를 유지하면서 최대 16.35까지 평균 탐색점의 수를 크게 줄여 속도를 향상시킬 수 있다.

참 고 문 헌

〔1〕 J. D. Robbins and A. N. Netravali, "Recursive motion compensation: a review," *Image Sequence Processing And Dynamic Sequence Analysis*, pp76-103, Springer Verlag, 1983.

〔2〕 J. R. Jain and A. K. Jain, "Displacement measurement and its application in inter frame image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799-1808, Dec. 1984.

〔3〕 K. P. Horn and B. G. Schunck, "Determining Optical flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.

〔4〕 G. Sorwar, M. Mushed, and L. Dooley, "Block-based true motion estimation using distance dependent thresholded search," *in Proc. ISCA Comp. Appl. In Indus. And Eng.*, pp. 45-48, 2001.

〔5〕 T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *in Proc. Nat. Telecommon. Conf., New Orleans*, LA. Nov.-Dec. 1981, pp.G5.3.1-G5.3.5.

〔6〕 R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technology*, vol. 4, pp. 438-443, Aug. 1994.

〔7〕 L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuit Syst. Video Technology*, vol. 6, pp.313-317, June 1996

〔8〕 C. H. Cheung, and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technology*, vol. 12, no. 12, Dec. 2002.

〔9〕 J. B. Xu, L. M. Po, and C. K. Cheung, "A new prediction model search algorithm for fast block motion estimation," *in Proc. ICIP*, 1997, vol.3, pp.610-613.

〔10〕 B. Liu and Zaccarin, "A new fast algorithms for the estimation of block motion vectors,"

IEEE Trans. CSVT, vol.3, no.2, pp. 148-157, April 1993.

〔11〕 C. W. Lam, L. M. Po, and C. H. Cheung, "A new cross-diamond search algorithm for fast block matching motion estimation," *in Proc. Neural Network and Signal Processing 2003*, Dec. 2003.

〔12〕 J. Y. Tham, S. Ranganath, M. Ranganath, and A. A Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technology*, vol. 8, no. 4, Aug. 1998.

〔13〕 C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technology,* vol. 12, no. 5, pp. 349-355, May 2002.

〔14〕 L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technology*, vol. 6, pp. 313-317, June 1996.

김 정 준 (Jung-Jun Kim) 준회원

2006년 2월 동아대학교 전기전자컴퓨터공학부 졸업

2006년~현재 한양대학교 전자통신컴퓨터공학과 석사과정

<관심분야> 영상처리, 영상압축

전 광 길 (Gwang-gil Jeon) 준회원

2003년 2월 한양대학교 전자전기컴퓨터공학과 졸업

2005년 2월 한양대학교 전자통신전파공학과 석사졸업

2005년~현재 한양대학교 전자통신컴퓨터공학과 박사 과정

<관심분야> 영상처리, 화질개선 및 영상압축

정 제 창 (Je-chang Jeong) 정회원

1980년 2월 서울대학교 전자 공학과 졸업

1982년 2월 KAIST 전기전자 공학과 (석사)

1990년 미국 미시간대학 전기 공학과 (공학박사)

1980~1986 KBS 기술연구소 연구원(디지털 TV 및 뉴미디어 연구)

1991~1995 삼성전자 멀티미디어 연구소 (MPEG, HDTV, 멀티미디어 연구)

1995~현재 한양대학교 전자통신컴퓨터공학과 교수 (영상통신 및 신호처리 연구실)

1998년 11월 27일 과학기술자상 수상

1998년 12월 31일 정보통신부장관상 표창

<관심분야> 영상처리, 영상압축