일반논문-07-12-5-11

고속 블록 정합 움직임 추정 기법 기반의 향상된 십자 다이아몬드 탐색

김 정 준[a)‡], 전 광 길[a)], 정 제 창[a)]

ENHANCED CROSS-DIAMOND SEARCH BASED FAST BLOCK MATCHING MOTION ESTIMATION ALGORITHM

Jungjun Kim[a)‡], Gwanggil Jeon[a)], and Jechang Jeong[a)]

요 약

본 논문에서는 새로운 고속 움직임 추정 알고리듬을 제공한다. 이 알고리듬의 이름은 향상된 십자 다이아몬드 탐색이며, 이것은 다이아몬드 탐색을 기초로 한다. 비록 다이아몬드 알고리듬이 가장 널리 알려진 고속 탐색 알고리듬이지만 몇몇의 시퀀스에 대해서 객관적인이고 주관적인 화질이 떨어지고 불필요한 후보 블록에 대한 블록 정합을 수행함을 알 수 있다. 그래서 고속 움직임 추정을 하기 위해서 우리가 제안한 향상된 십자 다이아몬드 알고리듬에서는 첫 번째 단계에서 작은 십자형 탐색을 하고 그 다음 단계에서는 다이아몬드 탐색을 사용한다. 실험결과 향상된 십자 다이아몬드 탐색은 많이 알려진 다른 고속 움직임 추정 알고리듬보다 화질을 향상시킬 뿐만 아니라 탐색 속도 또한 향상됨을 알 수 있다.

Abstract

A new fast motion estimation algorithm is presented in this paper. The algorithm, named Enhanced Cross-Diamond Search (ECDS), is based on the Diamond Search (DS) algorithm. The DS algorithm, even though faster than the most well-known algorithms, was found not to be very robust in terms of objective and subjective qualities for several sequences and the algorithm searches unnecessary candidate blocks. We propose a novel ECDS algorithm using a small cross search as the initial step, and large/small DS patterns as subsequent steps for fast block motion estimation. Experimental results show that the ECDS is much more robust, provides a faster searching speed, and smaller distortions than other popular fast block-matching algorithms.

Keyword : Motion Estimation, Motion Vector, Block Matching Algorithm, Diamond Search, SAD(Sum of Absolute Difference)

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 보상은 부호기의 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

a) 한양대학교 전자통신컴퓨터공학과
Department of Electronics and Computer Engineering, Hanyang University
‡ 교신저자 : 김정준(79jungjun@naver.com)
※ 본 논문은 IWAIT2007학회에 발표된 논문임. 본 연구는 서울시 산학연 협력 사업으로 구축된 서울 미래형콘텐츠컨버전스 클러스터 지원으로 수행되었습니다.

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 16×16 크기의 매크로블록(MB) 또는 임의의 크기 블록으로 나누어 블록 정합 알고리듬(Block Matching Algorithm)을 사용하여 블록단위로 현재 블록을 기준으로 하는 정해진 탐색영역(Search Window)안에서 정합블록을 찾아낸 다음 찾은 블록 위치를 나타내는 움직임 벡터와 현재 블록과의 차이값을 부호화하는 것을 말한다. 일반적으로 영상 코덱에서 사용 되는 전역 탐색 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 최적의 정합 블록을 찾아내는 방법들이 개발되었다. 가장 대표적인 탐색 기법인 다이아몬드 탐색 기법과 같은 경우 다른 종래의 고속 블록 정합 기법들보다 좋은 화질을 가질 수 있지만 초기 탐색에서 방향성을 고려하지 않고 현재 블록과 후보 블록을 비교를 함으로서 하나의 움직임 벡터를 찾기 위해 비교해야 할 탐색점의 수가 많아진다는 것을 알 수 있다.

그래서 본 논문에서 다이아몬드 탐색을 기초로 하여 새로운 고속 움직임 추정 기법을 제안함으로서 다른 고속 움직임 추정 기법과 비교하여 좋은 화질을 가지면서 더 적은 탐색 점수로 움직임 추정을 가능하게 한다. 본 논문의 구성은 다음과 같다. 2장에서 움직임 추정 기법의 개념에 대해 설명하고 3장에서는 다이아몬드 탐색 기법과 적응적 다이아몬드 탐색 기법들에 대해서 소개한다. 그리고 4장에서는 다이아몬드 탐색을 기반으로 한 향상된 십자형 다이아몬드 탐색 방법에 대하여 설명한다. 5장에서 기존 기법들과 성능을 비교 평가한 실험결과를 보여주고 마지막으로 6장에서 결론을 맺는다.

II. 움직임 추정 예측 방법

1. 움직임 추정(Motion Estimation)

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. 블록 정합 방법(BLOCK MATCHING ALGORITHM)

화면간의 시간 축 중복 데이터의 제거를 목적으로 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 블록으로 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 움

직임 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 다음의 식(1), (2), (3)에서 보듯이 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 블록 정합 방법을 이용한 움직임 벡터 검출에 대해 나타낸다.

$$MSE(x,y)=\frac{1}{N^2}\sum_{i=1}^{M}\sum_{j=1}^{N}[C(i,j,t)-P(i+x,j+y,t-1)]^2 \quad (1)$$

$$MAD(x,y)=\frac{1}{N^2}\sum_{i=1}^{M}\sum_{j=1}^{N}[C(i,j,t)-P(i+x,j+y,t-1)] \quad (2)$$

$$SAD(x,y)=\sum_{i=1}^{M}\sum_{j=1}^{N}[C(i,j,t)-P(i+x,j+y,t-1)] \quad (3)$$

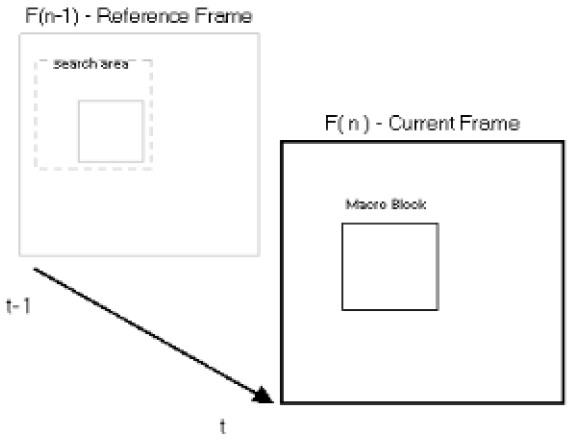

그림 1. 블록 정합 방법
Fig. 1. Block Matching Algorithm

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. 하지만 많은 계산량으로 인하여 실시간 비디오 코딩 응용 분야 및 소프트웨어 구현에 많은 어려움이 있기 때문에 계산량을 줄이고 보다 빠른 움직임 추정을 수행하기 위한 기법들이 많이 연구 되고 있다.

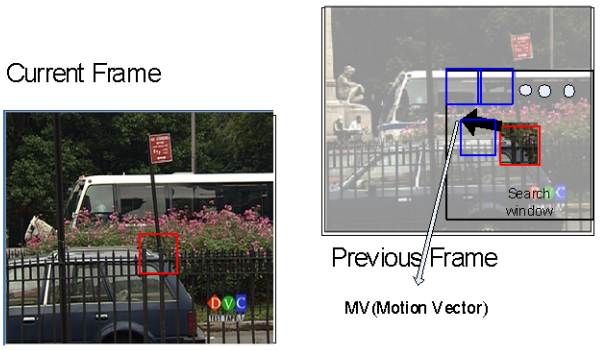

그림 2. 전역 탐색 기법
Fig. 2. Full Search Algorithm

그림 2는 전역 탐색 블록 정합 방법의 예를 나타낸 것이다. 이전 프레임을 중심으로 현재 프레임을 추정하기 위해서는 그림 2와 같이 이전 프레임 내에서 탐색 범위를 위와 같이 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.

위와 같이 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 기법은 탐색 영역내에 있는 모든 탐색점들에 대한 SAD값을 계산하고, 그 중 가장 작은 SAD값을 가지는 탐색점 위치로 향하는 벡터를 움직임 벡터로 추정한다. 이

방법으로 구한 벡터를 이용하여 움직임 추정 및 보상을 수행하면 가장 좋은 화질을 제공할 수 있지만 계산량이 많아 실시간 부호화에 적합하지 않다. 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.

1. 기존의 고속 탐색 알고리듬

각기 다른 형태의 탐색 패턴으로 탐색 영역의 (0,0) 위치를 시작점으로 탐색하는 기존의 고속 움직임 탐색 알고리듬들 아래에서 살펴본다. 이러한 방법들은 전역 탐색 기법에 비해 속도는 향상 되었지만, 객관적 화질(PSNR)을 저하시키는 단점을 가지고 있다.

1.1 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 3와 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.

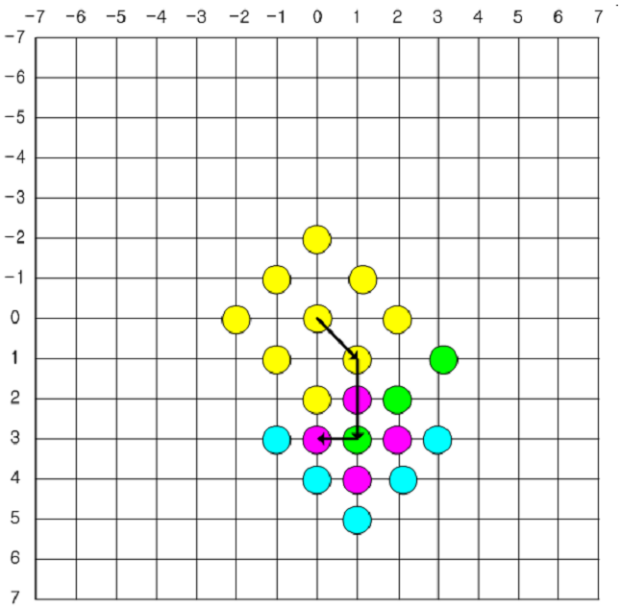

그림 3. 다이아몬드 탐색 기법
Fig. 3. Diamond Search Algorithm

1.2 십자 다이아몬드 탐색 기법(CDS: Cross Diamond Search)[8]

그림 4는 십자 다이아몬드 탐색 기법을 나타낸다. 십자 다이아몬드 탐색 기법은 다이아몬드 탐색 기법을 개선시켜 더욱 성능을 향상 시킬 수 있도록 하는 기법이다.

우선 다이아몬드 탐색 기법의 경우 최소 블록 정합 오차 지점을 찾는데 있어 LDSP 수행 후 SDSP를 한번 더 수행해

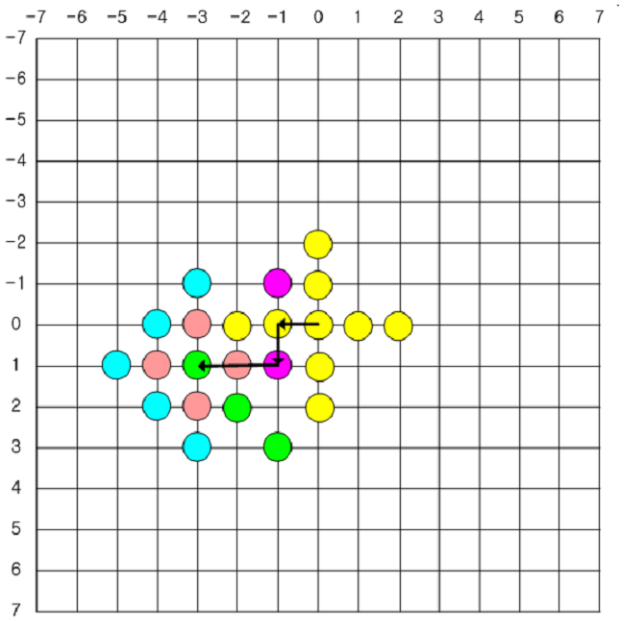

그림 4. 십자 다이아몬드 탐색 기법
Fig. 4. Cross Diamond Search Algorithm

야 하므로 최소 13개의 탐색 지점을 필요로 한다. 그러나 십자 패턴 다이아몬드 탐색 기법의 경우 초기 탐색 지점을 십자 형태의 9개점을 수행함으로써 영상 블록의 움직임이 없는 정지 영상 즉 최소 블록 정합 오차 지점이 원점인 경우 혹은 원점 주변일 경우 밑에서 설명할 움직임 추정 단계의 2단계에서 알 수 있듯이 단 두 지점만을 더 탐색하여 탐색점수를 줄임으로써 실행 시간을 단축시킬 수 있다. 다음은 십자 다이아몬드 탐색 기법의 탐색 단계를 보여준다.

- 1단계) 원점을 중심으로 십자 형태로 8개의 지점에 대한 블록 정합 오차를 구한다. 만약 원점이 최소 블록 정합 오차일 경우 탐색을 종료하게 되고 그렇지 않을 경우 최소 블록 정합 오차를 지니는 점에 대해 직각의 양 방향으로 두 점을 더 확장해서 탐색을 수행하게 된다.
- 2단계) 만약 1단계에서 원점 이외의 점이 선택되어 두 점을 더 확장해서 검사 시 최소 정합 오차의 변화가 없을 경우 탐색을 종료하게 된다. 그러나 만약 변화가 있을 경우 원점에서 현재 지점과 동일한 대각선 방향으로 LDSP 방식으로 3점을 더 추가해서 확장한다.
- 3단계) 만약 2단계에서 LDSP로 확장이 이루어 졌을 경우, 이 후의 탐색 과정은 다이아몬드 탐색 기법과 같이 LDSP와 SDSP를 수행하게 된다.

1.3 새로운 십자 다이아몬드 탐색 기법(NCDS: New Cross Diamond Search)[11]

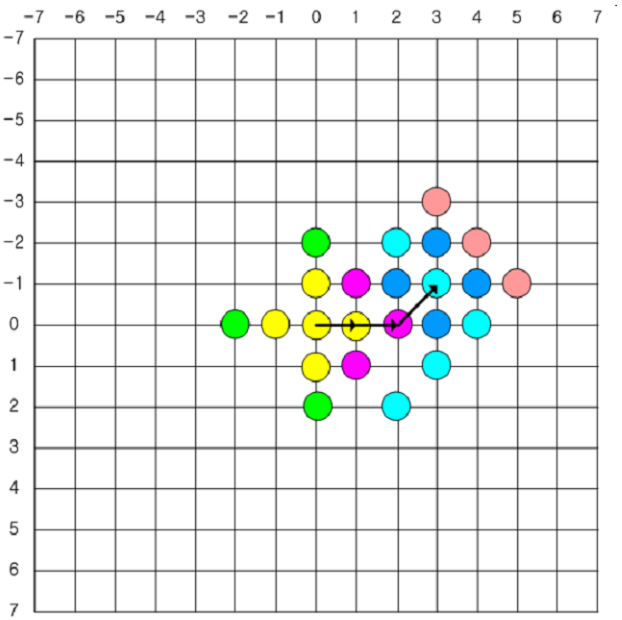

그림 5. 새로운 다이아몬드 탐색기법
Fig. 5. New Cross Diamond Search Algorithm

그림 5은 새로운 십자 다이아몬드 탐색 기법의 움직임 추정 패턴을 나타낸다. 새로운 십자 다이아몬드 탐색 기법은 십자 다이아몬드 탐색 기법을 더욱 개량, 발전시킨 것으로서 초기 탐색을 SDSP로 시작함으로써 십자 다이아몬드 탐색 기법이 갖는 9개의 초기 탐색 점으로 인한 속도 저하를 방지함으로써 특히 움직임이 없거나 적은 영상에 있어서 탁월한 속도 향상을 보인다. 다음은 새로운 십자 다이아몬드 탐색 기법의 움직임 추정 단계를 나타낸다.

- 1단계) 원점을 중심으로 SDSP 5개 지점에 대한 블록

정합 오차를 구한다. 만약 원점이 최소 블록 정합 오차일 경우 탐색을 종료하게 되고 그렇지 않을 경우 2단계로 넘어간다.

- 2단계) 만약 1단계에서 원점 이외의 점이 선택되었을 경우 동일 방향으로 SDSP를 한번 더 수행한다. 만약 이 후에도 변화가 없을 경우는 탐색을 종료 하고 변화가 있을 경우는 3단계 탐색을 수행한다.
- 3단계) 2단계에서 변화가 있을 경우 원점을 중심으로 LDSP의 가장자리의 3점을 더 확장해서 탐색한다. 만약 변화가 있을 경우는 이 후 LDSP 탐색을 수행하고 변화가 없을 경우는 이전 단계의 최소 블록 정합 오차 지점에서 동일 대각 방향 3점과 수평으로 떨어진 가장자리의 한 점에 대한 LDSP 4점을 탐색한다.
- 4단계) 3단계에서 변화가 없을 경우 SDSP 4점을 한번 더 수행한 후 최소 블록 정합 오차 지점을 구하여 탐색을 종료한다. 그렇지 않을 경우는 계속해서 LDSP를 수행한다.

Ⅳ. 제안하는 알고리듬

움직임을 예측하는데 있어서 영상을 일정 크기의 블록들로 분할하고 움직임 벡터를 구하는 고속 블록 정합 기법들은 영상의 대부분의 블록들은 움직임이 없거나 혹은 아주 작다는 특성에 바탕을 두고 있다. 그러므로 대부분 원점(0, 0)을 중심으로 정해진 탐색 범위 이내에서 움직임 벡터를 찾는 과정이 이루어진다. 예를 들어 3단계 탐색 기법(TSS: Three Step Search Algorithm)의 경우 영상의 블록 정합 오차 값이 최고 정합 지점을 중심으로 거리가 멀어질수록 커진다는 가정하에 기반을 두고 움직임 추정을 수행한다. 그러나 이는 탐색 영역의 일부만을 검색을 수행하게 되므로 국부 최소점(local minimum point)에 빠지게 되어 부정확한 움직임 벡터를 산출하게 된다. 이러한 단점을 해결하기 위해서 고안된 새로운 3단계 탐색 기법(NTSS: New Three Step Search Algorithm)은 영상의 대부분의 블록들에 있어서 최소 블록 정합 오차 값이 원점을 중심으로 분포한다는 특성에 기반을 두고 더욱 성능을 향상 시킨 것이라고 볼 수 있다. 또한 십자형 다이아몬드 탐색 기법과 새로운 십자형 다이아몬드 탐색 기법은 움직임 벡터가 원점(0, 0)에 많이 분포되어 있다는 가정을 기초로 하여 다이아몬드 탐색 기법을 탐색점의 수를 줄일 수 있도록 성능을 향상 시킨 것이라고 말할 수 있다.

제안하는 방법 역시 초기 탐색을 수행함에 있어서 같은 원리를 사용하였으며 대표적인 고속 움직임 탐색 기법인 십자형 다이아몬드 탐색 기법과 새로운 십자형 다이아몬드 탐색 기법을 개선시켜 성능을 향상시킨 것이다.

1. 제안된 알고리듬

표 1의 실험 동영상들에 대해 탐색영역의 거리를 ±7로 두고 움직임 벡터를 탐색할 경우, 대부분의 움직임 벡터가 탐색영역의 중심 주위에 분포하는 것을 알 수 있으며 표 2과 같이 움직임 벡터의 분포를 백분율로 나타내 보았을 때 탐색영역의 중심점으로부터 반경 1픽셀 내의 약 62.12%의 가장 많은 분포를 가지고 반경 2픽셀 내에 약 76.95%, 반경 3픽셀 내에는 약 85.59% 분포를 가짐을 알 수 있다.

표 1. 분석을 위한 실험 동영상
Table 1. Image Sequences used for Analysis

Format	Sequence
CIF (352 X 288, 80 frames)	flower foreman hall mother & daughter new children stefan

십자형 다이아몬드 탐색 기법의 경우 초기에 십자 형태의 9개의 점을 수행하고 새로운 십자형 다이아몬드 탐색 기법의 경우 5개의 초기 탐색 점 수행함으로서 움직임 없거나 움직임이 적은 영상 즉 최적의 블록 정합 오차지점이 반경 1픽셀 혹은 반경 2픽셀 내에 분포하고 있는 움직임

벡터 탐색에서는 탁월한 성능을 보인다.

하지만 십자 다이아몬드 탐색 기법과 새로운 십자 다이아몬드 탐색 기법는 반경 3픽셀에도 많은 양의 움직임 벡터가 존재함에도 불구하고 반경 3픽셀 분포하고 있는 움직임 벡터에 대해서는 고려하지 않음으로서 블록 정합 오차 계산에서의 성능 열화를 가져온다는 점을 알 수 있다.

그래서 향상된 십자 다이아몬드 탐색 기법(ECDS: Enhanced Cross Diamond Search Algorithm)은 반경 3픽셀 내에 분포하고 있는 움직임 벡터를 고려하여 블록 정합 오차를 실행함으로서 성능 향상을 가져온다.

표 2. 움직임 벡터의 분포
Table 2. Distribution of Motion vector

2. 제안된 알고리듬 수행 절차(Performance of The ECDS Algorithm)

표 2에 나타나 있는 것처럼 대각선보다는 가로, 세로에 움직임 벡터가 많이 존재한다는 것을 알 수 있으므로 향상된 십자 다이아몬드 탐색 기법은 초기탐색을 SDSP로 시작함으로서 십자 다이아몬드 탐색 기법이 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. 특히 움직임이 없거나 적은 영상에 있어서 다른 알고리듬 보다 탁월한 성능을 발휘함을 볼 수 있다. 다음은 향상된 십자 다이아몬드 탐색 기법의 움직임 추정 단계를 보여준다.

- 1단계) 시작점을 중심으로 한 SCSP(small cross search pattern)의 5개의 점에서 최소 블록 정합 오차 지점을 찾고 만약 그림 6(a)와 같이 중심에서 최소 블록 정합 오차 지점이 발생하면 탐색을 멈추고 아니면 다음 단계로 넘어간다. 그림 6(a)는 움직임 벡터를 (0,0)으로 결정된 경우를 보여준다.
- 2단계) 만약 최소 블록 정합 오차 지점이 (1,0)(혹은 (-1,0), (0,1), (0,-1)) 발생하면 그림 6(b)와 같이 추가된 4개의 점((2,0), (3,0), (1,-1), (1,1))을 탐색하고 최소 블록 정합 오차 지점이 (1,0)(혹은 (-1,0), (0,1), (0-1))에서 발생하면 탐색을 멈추고 아니면 3단계로 넘어간다. 그림 6(b)는 움직

표 2 움직임 벡터 분포
Table 2. Distribution of Motion vector

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
-7	0.03	0.016	0.009	0.012	0.01	0.013	0.014	0.07	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	0.01	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	0.02	0.015	0.021	0.023	0.127	0.155	0.023	0.019	0.018	0.013	0.01	0.007	0.018
-3	0.05	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	0.04	0.022	0.019	0.061
-1	0.125	0.045	0.05	0.062	0.112	0.575	1.064	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	5.706	4.274	1.862	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	1.869	0.655	0.109	0.057	0.045	0.036	0.142
2	0.107	0.049	0.048	0.06	0.056	0.175	0.623	1.471	0.686	0.211	0.051	0.029	0.029	0.03	0.071
3	0.022	0.01	0.013	0.017	0.029	0.114	0.138	0.502	0.143	0.126	0.038	0.017	0.02	0.019	0.058
4	0.012	0.009	0.01	0.013	0.012	0.073	0.089	0.229	0.018	0.067	0.017	0.023	0.011	0.01	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	0.01	0.009	0.009	0.026
6	0.017	0.008	0.01	0.011	0.011	0.02	0.24	0.074	0.04	0.027	0.005	0.008	0.006	0.008	0.026
7	0.039	0.01	0.011	0.009	0.018	0.018	0.055	0.373	0.04	0.047	0.016	0.01	0.014	0.014	0.071

임 벡터를 (1,0)으로 결정된 경우를 보여주며, 그림 6(c)는 움직임 벡터를 (-1,0)으로 결정된 경우를 보여준다.

■ 3단계) 3단계에는 3가지의 경우로 나눌 수 있다.

· 경우 1 : 만약 최소 블록 정합 오차 지점이 (2,0) (혹은 (-2,0), (0,2), (0,-2))에서 발생하면 그림 6(d)와 같이 2개의 점((2,-1), (2,1))을 추가적으로 탐색하고 (2,0)(혹은 (-2,0), (0,2), (0,-2))에서 최소 블록 정합 오차 지점이 나타나면 탐색을 멈추고 그렇지 않으면 4단계로 넘어간다.

· 경우 2 : 만약 최소 블록 정합 오차 지점이 (1,-1)(혹은 (-1,-1), (-1,1), (1,1))에서 발생하면 그림 6(f)와 같이 2개의 점((1,-2), (2,-1))을 추가적으로 탐색하고 (1,-1)(혹은 (-1,-1), (-1,1), (1,1))에서 최소 블록 정합 오차 지점이 발생하면 탐색을 멈추고 아니면 4단계로 넘어간다.

· 경우 3 : 만약 최소 블록 정합 오차 지점이 (3,0) (혹은 (-3,0), (0,3), (0,-3))에서 발생하면 그림 6(i)와 같이 3개의 점을 추가적으로 탐색하고 최소 블록 정합 오차 지점을 결정한 후 4단계로 넘어간다.

■ 4단계) 이전 단계의 최소 블록 정합 오차 지점을 중심으로 한 LDSP(large diamond search pattern) 실행하고 중심에서 최소 블록 정합 오차 지점을 찾으면 5단계로 넘어가고 아니면 4단계를 반복 수행한다.

■ 5단계) 이전 단계의 최소 블록 정합 오차 지점을 중심으로 한 SDSP(small diamond search pattern)을 수행한다.

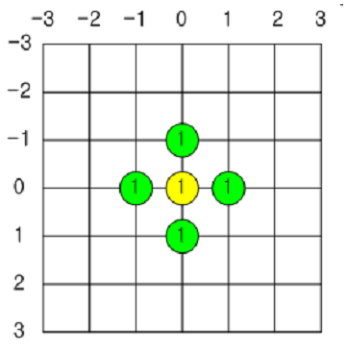

그림 6(a). 1단계에서 탐색을 멈추는 경우 - MV(0,0)
Fig. 6(a). First step stop with MV(0,0)

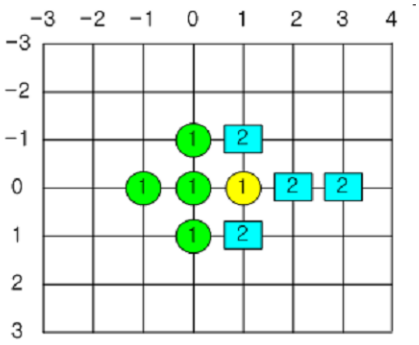

그림 6(b). 2단계에서 탐색을 멈추는 경우 - MV(1,0)
Fig. 6(b). Second step stop with MV(1,0)

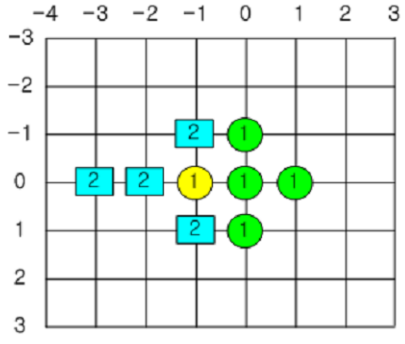

그림 6(c). 2단계의 탐색을 멈추는 경우 - MV(-1,0)
Fig. 6(c). Second step stop with MV(-1,0)

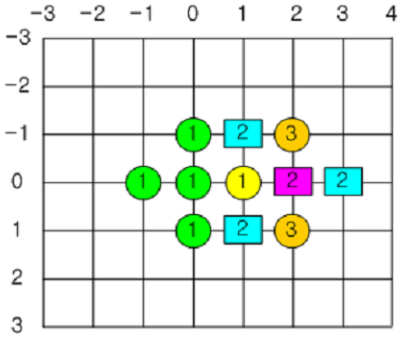

그림 6(d). 3단계의 경우1에서 탐색을 멈추는 경우 - MV(2,0)
Fig. 6(d). Third step stop with MV(2,0)

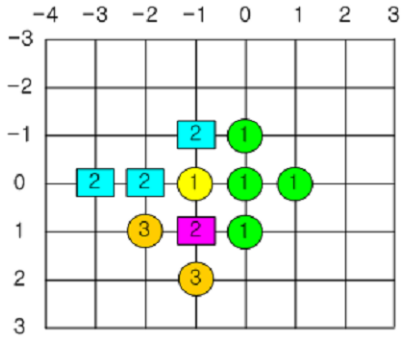

그림 6(g). 3단계의 경우2에서 탐색을 멈추는 경우 - MV(-1,1)
Fig. 6(g). Third step stop with MV(-1,1)

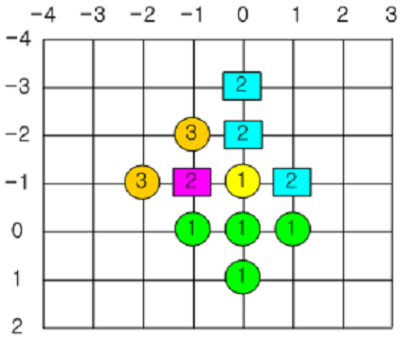

그림 6(h). 3단계의 경우2에서 탐색을 멈추는 경우 - MV(-1,-1)
Fig. 6(h). Third step stop with MV(-1,-1)

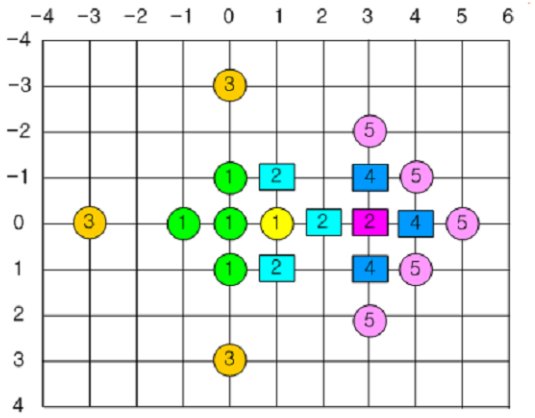

그림 6(i). 움직임 벡터가 (3,0)로 결정되는 경우
Fig. 6(i). Motion Vector(3,0) is found

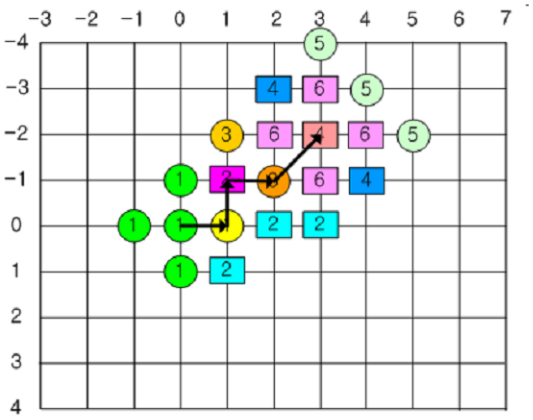

그림 6(j). 움직임 벡터가 (3,-2)로 결정되는 경우
Fig. 6(j). Motion Vector(3,0) is found

IV. 실험 결과 및 분석

실험은 MPEG-4 VM 인코더를 이용하여 이루어졌으며 모든 영상은 IPPP로 부호화 되었다. 비트율 제어(rate control)는 TM5를 적용하였고 frame rate는 30이고 bit rate는 1024[kbps]을 적용하였다. 그리고 CIF(352×288) 영상 "flower," "waterfall," "foreman," "hall monitor," "mother and daughter," "news," "silent," "tempete," "paris," "coastguard," "bridge," "children," "container," "bus," "football," and "Stefan."들에 대해 각각 120프레임씩을 대상으로 실험하였고, 비교 탐색 알고리듬으로는 3단계 탐색 기법, 새로운 3단계 탐색 기법, 4단계 탐색 기법, BBGDS, 육각 탐색 기법(HEXBS: Hexagon-based Search)[13], 다이아몬드 탐색 기법, 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법 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.

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

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

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.

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

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 3, 표 4, 표 5 그리고 표 6에 각각 나타내었다. 표 3, 표 4 그리고 표 5에서는 3단계 탐색 기법, 새로운 3단계 탐색 기법, 4단계 탐색 기법, BBGDS, 육각 탐색 기법(HEXBS: Hexagon-based Search), 다이아몬드 탐색 기법, 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법 그리고 제안한 향상된 십자 다이아몬드 탐색 기법의 평균 PSNR, MSE 그리고 블록 당 평균 SAD를 각각 나타내었고, 표 6에서는 한 블록당 평균 탐색점의 수를 나타내었다.

제안하는 탐색 알고리듬은 표 3, 4 그리고 5에서 볼 수 있듯이 대부분의 영상에 대해서 기존의 고속 탐색 알고리듬과 비슷한 수준의 PSNR, MSE 그리고 SAD를 유지하고 움직임이 많은 몇몇 영상에서는 다이아몬드 탐색기법보다 조금 떨어지지만 눈으로 보는 화질로는 구분할 수 없을 정도다. 그러나 표 6에서 나타나 있는 것처럼 탐색점의 수에 관해서는 3단계 탐색 기법과 비교하였을 때는 평균적으로 블록당 평균 16.79의 탐색점의 수를 감소시키고 다이아몬드 탐색 기법과 비교하였을 때는 7.29의 탐색점의 수를 감소시켰다. 그리고 다이아몬드 탐색 기법의 성능을 향상시

표 3. 기존 방법들과의 평균 PSNR (dB) 비교.
Table 3. Performance Comparison of Average PSNR (dB).

	TSS	FSS	NTSS	BBGDS	HEX	DS	CDS	NCDS	Proposed ECDS
flower	27.99	27.96	27.99	27.99	27.99	27.98	27.98	27.98	27.98
waterfall	30.54	30.54	30.54	30.54	30.55	30.55	30.45	30.53	30.53
foreman	32.06	32.20	32.07	32.05	32.08	32.13	32.11	32.10	32.14
hall	33.82	33.99	33.94	33.83	33.96	33.98	33.86	33.84	33.85
m&d	35.12	35.13	35.03	35.07	35.15	35.18	35.18	35.17	35.18
new	33.83	33.86	33.76	33.80	33.80	33.84	33.81	33.80	33.84
silent	32.88	32.95	32.90	32.86	32.91	32.91	32.90	32.89	32.93
tempete	28.87	28.89	28.84	28.84	28.87	28.88	28.84	28.84	28.85
paris	30.22	30.24	30.16	30.18	30.29	30.22	30.19	30.15	30.19
coastguard	29.74	29.78	29.73	29.74	29.77	29.76	29.74	29.73	29.74
bridge	35.06	35.05	35.06	35.06	35.06	35.06	35.05	35.05	35.06
children	32.21	32.21	32.13	32.15	32.21	32.19	32.14	32.13	32.13
container	32.30	32.32	32.32	32.31	32.28	32.34	32.34	32.31	32.31
stefan	29.54	29.57	29.55	29.55	29.55	29.57	29.56	29.56	29.56
bus	29.22	29.23	29.19	29.20	29.23	29.23	29.21	29.19	29.20
football	33.17	33.02	33.07	33.09	33.07	33.20	33.16	33.16	33.15
Average	31.66	31.68	31.64	31.64	31.67	31.69	31.66	31.65	31.67
Ave-ECDS	-0.01	0.01	-0.03	-0.03	0.00	0.02	-0.01	-0.02	0.00

표 4. 기존 방법들과의 평균 MSE 비교.
Table 4. Performance Comparison of Average MSE.

	TSS	FSS	NTSS	BBGDS	HEX	DS	CDS	NCDS	Proposed ECDS
flower	103.41	103.98	103.38	103.46	103.38	103.56	103.69	103.59	103.62
waterfall	57.36	57.43	57.44	57.51	57.28	57.30	58.66	57.56	57.56
foreman	40.50	39.16	40.38	40.54	40.31	39.79	40.00	40.10	39.72
hall	26.98	25.96	26.26	26.96	26.15	26.01	26.78	26.89	26.82
m&d	20.01	19.98	20.41	20.23	19.86	19.72	19.76	19.78	19.75
new	26.92	26.74	27.40	27.09	27.15	26.86	27.07	27.11	26.85
silent	33.54	32.96	33.39	33.67	33.26	33.25	33.35	33.42	33.15
tempete	84.42	83.94	84.93	84.95	84.36	84.23	84.98	84.99	84.70
paris	61.75	61.50	62.67	62.42	60.79	61.77	62.23	62.82	62.24
coastguard	68.99	68.47	69.20	69.13	68.65	68.65	69.01	69.17	69.02
bridge	20.29	20.30	20.29	20.30	20.30	20.30	20.32	20.32	20.31
children	39.05	39.12	39.82	39.65	39.07	39.29	39.72	39.81	39.80
container	38.27	38.08	38.12	38.25	38.49	37.90	37.99	38.25	38.23
stefan	72.32	71.85	72.21	72.22	72.18	71.84	71.91	71.91	71.91
bus	77.84	77.72	78.38	78.22	77.63	77.61	78.08	78.39	78.21
football	31.35	32.43	32.07	31.91	32.07	31.15	31.42	31.43	31.51
Average	50.19	49.98	50.40	50.41	50.06	49.95	50.31	50.35	50.21
Ave-ECDS	-0.02	-0.23	0.19	0.20	-0.15	-0.26	0.10	0.14	0.00

표 5. 기존 방법들과의 블록 당 평균 SAD 비교.
Table 5. Performance Comparison of Average SAD.

	TSS	FSS	NTSS	BBGDS	HEX	DS	CDS	NCDS	Proposed ECDS
flower	2291.72	2280.60	2251.56	2257.54	2286.54	2260.62	2253.89	2261.24	2266.08
waterfall	1461.18	1462.31	1461.13	1462.50	1461.41	1459.88	1462.98	1463.04	1463.09
foreman	1197.86	1177.16	1184.40	1189.84	1210.10	1182.74	1202.81	1193.25	1189.72
hall	833.33	822.95	825.57	834.83	826.37	823.29	845.93	837.67	836.31
m&d	691.21	691.63	698.25	696.32	690.93	694.66	695.79	692.67	692.34
new	821.18	819.56	824.96	822.75	825.61	819.81	825.16	824.87	820.33
silent	1076.46	1070.32	1073.29	1081.52	1076.41	1071.57	1082.11	1081.71	1078.21
tempete	2053.17	2049.41	2056.84	2061.48	2057.35	2049.99	2047.03	2045.47	2043.48
paris	1471.04	1468.26	1474.64	1476.79	1464.03	1466.77	1468.28	1485.13	1479.57
coastguard	2036.32	2039.57	2026.26	2057.33	2043.79	2027.36	2042.38	2053.73	2057.27
bridge	643.77	644.49	643.49	644.40	644.41	643.92	644.19	644.94	644.95
children	1422.57	1447.30	1427.26	1461.22	1456.61	1444.31	1450.82	1448.43	1446.14
container	1013.43	1011.36	1011.18	1014.09	1016.90	1009.21	1011.50	1013.77	1013.65
stefan	2743.28	2663.74	2748.20	2772.08	2684.27	2648.64	2710.82	2732.02	2657.88
bus	2972.83	3081.71	2979.83	3113.89	3135.15	3052.19	3079.50	3075.53	3073.27
football	2370.62	2446.02	2371.62	2482.63	2450.63	2416.35	2493.30	2513.16	2518.13
Average	1568.75	1573.52	1566.15	1589.33	1583.16	1566.96	1582.28	1585.41	1580.03
Ave-ECDS	-11.28	-6.51	-13.88	9.30	3.13	-13.07	2.25	5.38	0.00

표 6. 기존 방법들과의 블록 당 평균 탐색점의 수 비교.
Table 6. Performance Comparison of Average Search Point.

	TSS	FSS	NTSS	BBGDS	HEX	DS	CDS	NCDS	Proposed ECDS
flower	25.00	20.10	22.19	12.21	12.66	15.75	12.22	10.45	8.87
waterfall	25.00	17.09	17.54	9.30	11.01	13.12	9.21	5.37	5.37
foreman	25.00	19.97	22.14	13.21	12.77	16.17	13.42	10.46	9.31
hall	25.00	17.21	17.51	9.26	11.12	13.22	9.34	5.36	5.28
m&d	25.00	17.65	18.34	9.92	11.35	13.69	9.92	6.20	5.93
new	25.00	17.44	17.93	9.60	11.27	13.50	9.66	5.80	5.64
silent	25.00	17.83	18.65	10.54	11.59	14.14	10.83	6.83	6.56
tempete	25.00	17.53	18.99	10.21	11.33	13.79	10.23	6.53	6.38
paris	25.00	17.86	18.66	10.11	11.48	13.89	10.19	6.54	6.20
coastguard	25.00	20.67	23.39	13.71	13.33	16.97	14.50	13.01	10.58
bridge	25.00	17.08	17.24	9.04	11.04	13.07	9.06	5.01	5.01
children	25.00	17.93	18.90	11.03	11.73	14.37	11.32	7.48	7.11
container	25.00	17.12	17.36	9.09	11.06	13.12	9.09	5.10	5.07
stefan	25.00	20.30	24.43	14.83	13.80	17.82	16.73	13.78	12.19
bus	25.00	22.12	29.01	17.97	15.57	20.91	21.66	17.25	14.94
football	25.00	22.96	29.09	21.40	16.32	23.23	26.91	18.64	16.86
Average	25.00	18.80	20.71	11.96	12.34	15.42	12.77	8.99	8.21
Ave-ECDS	16.79	10.59	12.50	3.75	4.13	7.29	4.56	0.78	0.00

킨 십자 다이아몬드 탐색 기법, 새로운 십자 다이아몬드 탐색 기법과 비교하였을 때에도 많은 탐색점의 수를 감소시키는 것을 볼 수 있다.

V. 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.

제안한 움직임 벡터 추정 기법은 반경 3픽셀 내에 많은 양의 움직임 벡터가 존재한다는 점을 고려하여 시작점을 중심으로 새로운 형태의 움직임 벡터 추정 기법을 제안하여 반경 3픽셀 내에서 움직임 벡터를 고속으로 추정할 수 있도록 하였다.

실험 결과에서 보면 알 수 있듯이 제안된 기법은 이전에 알려진 고속 탐색 기법과 비교하여 대부분 영상에서 비슷한 수준의 PSNR, MSE 그리고 SAD를 유지하면서 최대 16.79까지 평균 탐색점의 수를 크게 줄여 속도를 향상시킬 수 있다.

그리고 만약에 주변 블록의 PSNR, MSE 그리고 SAD등의 평균값을 이용하여 기준값을 정하고 기준값 이하의 값이 나올 경우 블록 정합을 바로 멈추고 움직임 벡터를 결정하는 등의 연구가 이루어진다면 더욱 속도를 향상시킬 수 있을 것이다.

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.

저 자 소 개

김 정 준

- 2006년 2월 : 동아대학교 전기전자컴퓨터공학부 졸업
- 2006년 3월~현재 : 한양대학교 전자통신컴퓨터공학과 석사과정
- 주관심분야 : 영상처리 및 영상압축

전 광 길

- 2003년 2월 : 한양대학교 전자전기컴퓨터공학과 졸업
- 2005년 2월 : 한양대학교 전자통신전파공학과 석사졸업
- 2005년~현재 : 한양대학교 전자통신컴퓨터공학과 박사 과정
- 주관심분야 : 영상처리, 화질개선 및 영상압축

정 제 창

- 1980년 2월 : 서울대학교 전자 공학과 졸업
- 1982년 2월 : KAIST 전기전자 공학과 석사
- 1990년 : 미국 미시간대학 전기 공학과 공학박사
- 1980~1986년 : KBS 기술연구소 연구원(디지털 TV 및 뉴미디어 연구)
- 1991~1995년 : 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
- 주관심분야 : 영상처리 및 영상압축