논문-98-3-1-10

# 변형된 비용 함수를 이용한 움직임 추정 기법

조 한 욱*, 서 정 욱*, 정 제 창*

# Motion Estimation Using Modified Cost Functions

H. W. Cho*, J. W. Suh*, and J. Jeong*

요 약

최근 HDTV나 화상회의 시스템, VOD(video on demand) 서비스등에서 쓰이는 영상 신호의 부호화가 주요한 관심사가 되고 있다. 동영상 압축 알고리즘에서 움직임 추정기법은 매우 중요한 역할을 담당하는 반면, 수행시간이나 하드웨어 구현에 어려움이 많아 이를 개선하기 위한 많은 알고리즘들이 개발되어 왔다. 본 논문에서는 적절한 화소 분류를 통해 우수한 화질과 적은 계산량, 간단한 하드웨어 구조를 가지는 효율적인 움직임 추정기법을 제안한다. 기존의 1-비트 화소 분류 방법에서 변형된 새로운 비용 함수를 이용한 2-비트, 3-비트 화소 분류 방법과 2차 비용함수를 이용한 화소 분류 방법을 제안하였다. 또한 여러 고속 움직임 추정 알고리즘과도 쉽게 연결하여 사용할 수 있으며 우수한 성능을 나타내는 것을 모의 실험을 통해 보였다.

Abstract

The coding of video sequences has been the focus of research in recent years. High Definition TV(HDTV), video conferencing and video on demand(VOD) are some of the well known applications. In the moving picture compression, motion estimation algorithm plays a very important role, but due to its high computational complexity, there has been many approaches to overcome this difficulty. We propose a new block matching criterion that uses modified cost functions. This new block matching criterion classifies pixels into 2-levels, 4-levels and 8-levels with a suitable thresholding method. We also show that our new proposed algorithms can easily be combined with many other fast motion estimation algorithms with good performance.

## I. 서 론

최근의 디지털 기술의 발달로 인해 멀티미디어 통신, 화상회의, HDTV, VOD등의 여러 가지 멀티미디어 서비스가 대중화됨에 따라 동영상 데이터의 효율적인 압축 기술의 개발이 중요한 문제로 대두되고 있다. 동영상 압축 기술 중에서도 움직임 추정(motion estimation)기법은 동영상 압축 알고리즘에서 매우 중요한 역할을 담당하는 반면, 계산량이 많아 수행시간이나 하드웨어 구현시 어려움이 많으므로, 이를 개선하기 위한 많은 알고리즘들이 개발되어 왔다[1][5]. 이 알고리즘들이 주로 탐색 회수를 줄임으로써 계산량을 줄여 화질 면에서는 만족할 만한 결과를 보이지 못한 반면 본 논문에서는 적절한 임계치를 이용한 화소 분류를 통해 우수한 화질과 적은 계산량, 그리고 간단한 하드웨어 구조를 가지는 효율적인 블록 정합 기준을 이용한 새로운 움직임 추정 기법을 제안한다.

효율적인 동영상 데이터의 압축은 시간적 중복성과 공간적 중복성을 제거함으로써 이루어진다. 이중에서 시간적 중복성을 제거하는 방법은 연속적인 영상 신호에서 현재 프레임의 화소들이 이전 프레임에 비해 어느 정도 움직였는지를 벡터로 나타낸 움직임 벡터를 추정하여 전체 영상의 전송대신 움직임 벡터를 전송함으로써 전송 데이

* 한양대학교 전자통신공학과
Dept. of Electronic Communications Engineering, Hanyang University

터량을 줄이는 방법이다. 움직임 추정 방법은 크게 화소 순환 알고리즘(pel recursive algorithm)과 블록 정합 알고리즘(block matching algorithm)으로 나눌 수 있는데, 화소 순환 알고리즘에 비해 블록 정합 알고리즘이 수행 시간이 적게 소요되며 하드웨어 구현이 용이하기 때문에 대부분의 움직임 보상 부호화에 널리 이용되고 있다. 블록 정합 알고리즘 중에서도 전체 프레임을 작은 단위 블록으로 나눈 후 각 단위 블록별로 전체 영역에서 움직임 벡터를 추정하는 전역 탐색 방법(full search block matching algorithm)이 계산량은 많으나 우수한 성능으로 인해 널리 사용되고 있다. 제안하는 움직임 벡터 추정 방법은 기존의 전역 탐색 방법에서처럼 현재 프레임과 이전 프레임에서 대응되는 블록간의 화소들의 차(difference)를 구한 후 기존의 블록 정합 기준으로 움직임 벡터 결정 기준인 평균 제곱 오차(mean square error)나 평균 절대 오차(mean absolute)를 쓰는 대신 적절한 임계치(threshold)를 정해 화소들을 분류한 후 각 구간별로 대표값을 주어 그 대표값들의 합이 최소가 되는 블록을 움직임 벡터로 추정하는 방법이다[6]. 실험은 구간을 각각 두 구간, 네 구간, 여덟 구간으로 나누어 실험하였다. 그 결과는 계산량과 하드웨어 구현에 있어 기존의 전역 탐색 방법에 비해 많은 개선이 있었으며, 화질면에 있어서도 평균 절대 오차(mean absolute error)를 이용한 방법보다 월등한 성능을 보였고, 평균 제곱 오차(mean square error)를 이용한 방법에 거의 근접하는 성능을 보였다.

논문의 구성은 다음과 같다. 먼저 1장에서는 서론을, 2장에서는 일반적인 움직임 추정 기법과 블록 정합 알고리즘에 대해서 설명하고, 3장에서는 제안하는 방법인 화소 분류를 통한 새로운 블록 정합 기준, 즉 1-비트, 2-비트, 3비트 화소 분류 방법, 그리고 2차 비용 함수를 이용한 화소 분류 방법에 대해서 설명하고 또한 적절한 임계치를 찾는 알고리즘에 대해서 설명한다. 4장에서는 제안하는 새로운 블록 정합 기준과 다른 고속 알고리즘과의 결합에 대해서 설명하고 5장에서는 위의 여러 가지 알고리즘들에 대한 다양한 실험 결과와 그 결과에 대한 고찰을 하였으며, 마지막으로 6장에서는 이상의 결과를 종합하여 결론을 지었다.

## II. 블록 정합 알고리즘

움직임 추정 및 보상 부호화에 있어서 가장 중요한 것은 이동 정보를 갖고 있는 움직임 벡터를 정확히 검출하는 것이며, 그 방법에 따라 블록 정합 알고리즘(block matching algorithm)과 화소 순환 알고리즘(pel recursive algorithm)으로 크게 나눌 수 있다. 블록 정합 알고리즘은 각 블록 단위로 움직임을 찾아 그 움직임 벡터를 그 블록의 모든 화소에 적용하는 방법이며, 화소 순환 알고리즘은 각 화소 단위의 움직임 벡터를 이전 화소들의 데이터를 사용하여 계산한 후 적용하는 방법이다. 화소 순환 알고리즘에 비해 블록 정합 알고리즘이 수행 시간이 적게 소요되며 하드웨어 구현이 용이하기 때문에 대부분의 움직임 보상 부호화에 널리 이용되고 있다.

그림 1은 블록 정합 알고리즘 중 가장 일반적인 방법인 전역 탐색 방법(full search method)으로 움직임 벡터를 찾는 과정을 나타낸 그림이다.

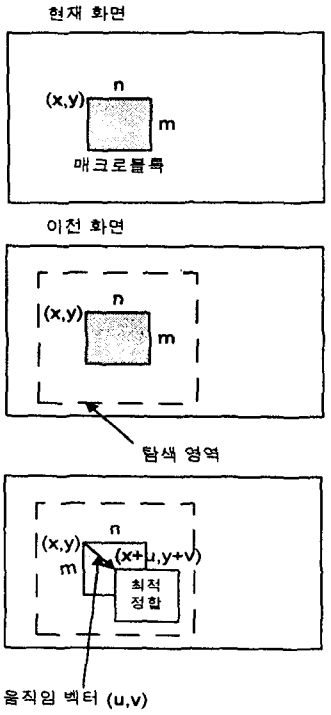



그림 1. 블록 정합 알고리즘으로 움직임 벡터를 찾는 과정
Fig. 1. Motion estimation using block matching algorithm

인접한 프레임 사이에서 가능한 최대 움직임이 x, y 방향에서 각각 화소단위로 p 이고, 움직임을 탐색해야할 블록의 크기가 m × n 이라면, 탐색영역의 크기는 (2p + m) × (2p + n)이 된다. 움직임 벡터를 찾고자 하는 현재 프레임의 블록과 같은 위치로부터 (2p + m) × (2p + n) 크기의 탐색영역을 이전 프레임에서 설정한다. 현재 프레임의 탐색 블록과 탐색영역내의 블록들과의 유사도를 화소단위로 이동해 가며 구한다. 유사도가 가장 높은 블록, 즉 정합 에러(matching error)가 가장 작은 블록의 위치를 구한다. 이렇게 구한 블록과 현재 탐색 블록과의 위치 차이가 바로 움직임 벡터이다. 정합 에러를 계산하는 함수들로 주로 사용하는 블록 정합 기준들에는 평균 제곱 오차(mean squared error)나 평균 절대 오차(mean absolute error)등이 있다.

$$MSE(i,j)=\frac{1}{mn}\sum_{x=1}^{m}\sum_{y=1}^{n}[I(x,y,t)-I(x+i,y+j,t-\tau)]^2 \quad (1)$$

$$MAE(i,j)=\frac{1}{mn}\sum_{x=1}^{m}\sum_{y=1}^{n}|I(x,y,t)-I(x+i,y+j,t-\tau)| \quad (2)$$

이러한 블록 정합 기준들을 써서 전체 탐색 영역내에서 가능한 모든 움직임을 탐색하는 방식이 전역 탐색 방법이다. 모든 움직임을 탐색하기 때문에 블록 정합 알고리즘으로 얻을 수 있는 가장 정확한 움직임을 찾을 수 있다. 전역 탐색 방법의 한 블록당 탐색 횟수는 프레임간의 최대 움직임을 p 라고 했을 때 $(2p+1)^2$이 되며 한 프레임에서는 (영상의 블록의 개수) × $(2p+1)^2$이 되므로 엄청난 양의 계산량이 필요하다. 이처럼 전역 탐색 방법은 정확한 움직임 벡터를 찾을 수 있지만 많은 계산량 때문에 실시간 시스템 구현에 문제점이 많다. 따라서 계산량을 줄이면서 정확한 움직임 벡터를 찾기 위한 여러 가지 고속 알고리즘들의 연구가 활발히 진행되고 있다.[1][6]

## III. 제안하는 새로운 블록 정합 기준

### 1. 기존의 1-비트 화소 분류 방법

전역 탐색 방법에서 블록 정합 기준으로 주로 사용하는 평균 제곱 오차(MSE)나 평균 절대 오차(MAE)는 정확한 움직임 벡터를 찾을 수 있다는 장점에도 불구하고 많은 계산량 때문에 실시간 시스템 구현에 어려움이 있다. 따라서 기존의 블록 정합 기준보다 적은 계산량과 간단한 하드웨어 구현을 가능하게 해주는 새로운 블록 정합 기준이 1-비트 화소 분류 방법이다.[1] 이 방법에서는 기존의 전역 탐색 방법과 마찬가지의 방법으로 탐색을 행한 후 정합 에러 계산시 새로운 블록 정합 기준을 사용함으로써 효율적인 움직임 추정을 행한다. 제안된 방법에서는 먼저 현재 탐색 블록과 이전 프레임의 블록의 화소의 차이값들을 정합 화소(matching pel)와 비정합 화소(mismatching pel)로 나눈다.

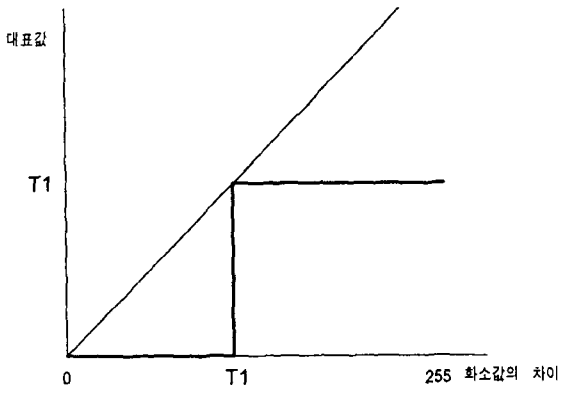



그림 2. 1-비트 화소 분류 방법
Fig. 2. 1-bit pixel classification method

$$T(x,y,i,j)=\begin{cases}1, & \text{if } |I(x,y,t)-I(x+i,y+j,t-\tau)|\le t\\ 0, & otherwise\end{cases} \quad (3)$$

즉, 적당한 임계치(threshold)를 정한 후 그 화소에서의 차이값이 임계치보다 크면 정합 화소로, 임계치보다 작으면 비정합 화소로 결정하는 것이다. 그 후 정합 화소들의 개수의 합을 구해 그 합이 가장 많은 블록의 위치를 움직임 벡터로 결정한다. T(x, y, i, j)는 화소 차이값에 대한 이진 비용 함수이며, 정합 화소인지 비정합 화소인지에 따라 각각 1과 0값을 가진다.

$$G(i,j)=\sum_x\sum_y T(x,y,i,j) \quad (4)$$
$$(\ i=0,\ \pm1\ \cdots\ \pm p\ \&\ j=0,\ \pm1\ \cdots\ \pm p\ )$$

G(i, j)의 값은 현재 블록과 x, y 방향으로 각각 i, j만큼 이동된 이전 프레임의 블록간의 정합에서 정합 화소들의 개수이다.

$$G_m(d_h, d_v) = \max\{G(i,j)\} \quad (5)$$

따라서 i와 j를 변화시켜가며 탐색해서 가장 큰 값을 갖는 G(i,j)에서의 i, j값이 움직임 벡터의 좌표가 된다. 하드웨어 구조면에서 보면 정합 에러 계산에 평균 절대 오차(MAE)를 이용하는 경우, m x n 크기의 블록 크기와 화소당 8비트의 신호일 때 ( 8 + $\log_2$ m x n )비트의 가산기가 필요한 반면 화소 분류 방법의 경우 임계치 비교기(threshold comparator)와 $\log_2$ (m x n) 비트의 카운터만 있으면 되기 때문에 상당히 간단한 하드웨어 구조를 가지게 된다.

### 2. 제안하는 방법 1 : 2-비트 화소 분류법

기존의 1-비트 화소 분류 방법은 간단한 하드웨어 구조가 가능하다는 장점이 있지만 화질 면에서 기대한 만큼의 성능을 보여 주지 못하고 있다. 즉 평균 제곱 오차뿐만 아니라 평균 절대 오차 방법에 비해서도 상당히 떨어지는 성능을 보이고 있다. 이에 본 논문에서는 기존의 방법에서 쓰인 0과 1로 나누는 이진 비용 함수 대신 비교하는 비용 함수(cost function)를 평균 제곱 오차(mean squared error)에 근사화 시킬것인지 평균 절대 오차(mean absolute error)에 근사화 시킬것인지에 대한 것과 화소 분류에 사용하는 임계치(threshold)를 4단계 또는 8단계로 나눌것인가에 따라 제안하는 방법 1, 2, 3 으로 나누어 제안한다. 그림 3 에서 보는 것과 같이 현재 탐색 블록과 이전 프레임의 화소들의 차이값들을 세 개의 임계치를 정해 4개의 구간으로 분류한 후 그 구간의 임계치를 그 구간에 속하는 화소들의 대표값으로 부여한다.

그림에서 보는 바와 같이 비용 함수를 이용해 화소의

차이값들을 분류하게 되면 상당수의 화소 차이값들이 첫 번째 구간에 속하게 되어 대표값으로 0을 부여받게 되어 그만큼 계산량을 줄일 수 있게 된다.

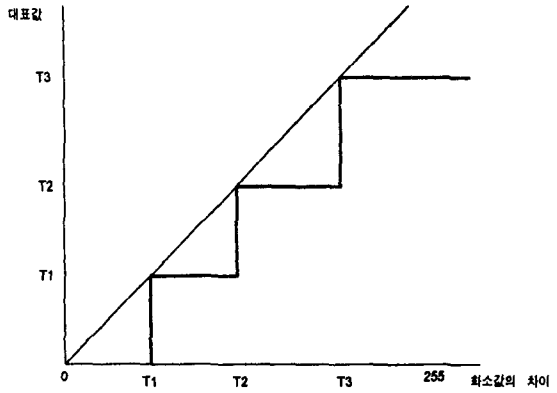



그림 3. 2-비트 화소 분류 방법
*Fig. 3. 2-bit pixel classification method*

각각의 임계치를 $T_1$, $T_2$, $T_3$라 하면 각 화소의 대표값은 다음과 같다.

$$\begin{aligned}
T(x,y,i,j) &= 0, \quad \text{if } 0 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_1 \\
T(x,y,i,j) &= T_1, \text{ if } T_1 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_2 \\
T(x,y,i,j) &= T_2, \text{ if } T_2 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_3 \\
T(x,y,i,j) &= T_3, \text{ if } T_3 \le |I(x,y,t) - I(x+i,y+j,t-\tau)|
\end{aligned} \tag{6}$$

그 후 정합 화소들의 대표값들의 합을 구해 그 합이 가장 작은 블록의 위치를 움직임 벡터로 결정한다.

$$G(i,j) = \sum_x \sum_y T(x,y,i,j) \tag{7}$$
$$(\text{I}=0, \pm 1 \cdots \pm \text{p} \ \& \ \text{j}=0, \pm 1 \cdots \pm \text{p})$$

G(i,j)의 값은 현재 블록과 x,y 방향으로 각각 i, j만큼 이동된 이전 프레임의 블록간의 정합에서 정합 화소들의 개수이다.

$$G_m(d_h, d_v) = \min\{G(i,j)\} \tag{8}$$

### 3. 제안하는 방법 2 : 3-비트 화소 분류법

3 비트 화소 분류 방법은 2-비트 화소 분류 방법과 마찬가지로 현재 탐색 블록과 이전 프레임의 화소들의 차이값들을 일곱 개의 임계치를 정해 여덟 개의 구간으로 분류한 후 그 구간의 임계치를 그 구간에 속하는 화소들의 대표값으로 부여한 후 정합 화소들의 대표값들의 합을 구해 그 합이 가장 작은 블록의 위치를 움직임 벡터로 결정한다.

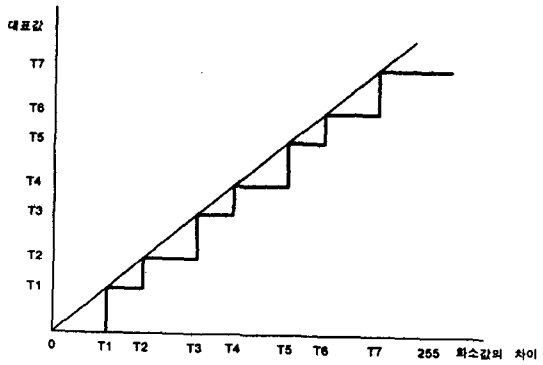



그림 4. 3-비트 화소 분류 방법
Fig. 4. 3-bit pixel classification method

$$\begin{aligned}
T(x,y,i,j) &= 0, \quad \text{if } 0 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_1 \\
T(x,y,i,j) &= T_1, \text{ if } T_1 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_2 \\
T(x,y,i,j) &= T_2, \text{ if } T_2 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_3 \\
T(x,y,i,j) &= T_3, \text{ if } T_2 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_4 \\
T(x,y,i,j) &= T_4, \text{ if } T_4 < |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_5 \\
T(x,y,i,j) &= T_5, \text{ if } T_5 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_6 \\
T(x,y,i,j) &= T_6, \text{ if } T_6 \le |I(x,y,t) - I(x+i,y+j,t-\tau)| \le T_7 \\
T(x,y,i,j) &= T_7, \text{ if } T_7 \le |I(x,y,t) - I(x+i,y+j,t-\tau)|
\end{aligned} \tag{9}$$

$$G_m(d_h, d_v) = \min\{\sum_x \sum_y T(x,y,i,j)\} \tag{10}$$

3-비트 화소 분류 방법은 2-비트 화소 분류 방법에 비해 하드웨어 구조의 복잡도는 다소 커지지만 화질면에서 좀더 나은 성능을 보여주고 있다. 그러나 더 이상의 구간으로 화소를 분류하는 것은 복잡도의 증가에 비해 화질의 향상이 그다지 효율적이지 못함을 실험을 통해 알 수 있

었다.

### 4. 제안하는 방법 3 : 2차 함수로 모델링한 비용 함수를 이용한 화소 분류 방법

제안하는 방법 1과 제안하는 방법 2는 평균 절대 오차(MAE)에서 사용하는 화소들의 차이값의 절대값을 사용한 방법인데 비해 제안하는 방법 3에서는 평균 제곱 오차(MSE)에서 사용하는 화소들의 차이값의 제곱을 이용해서 화소들을 분류하였다. 이 경우 계산량의 증가는 있지만 평균 제곱 오차에 근사화하는 비용 함수를 사용하는 만큼 화질에서, 특히 PSNR 측면에서는 월등한 성능을 보이고 있다.

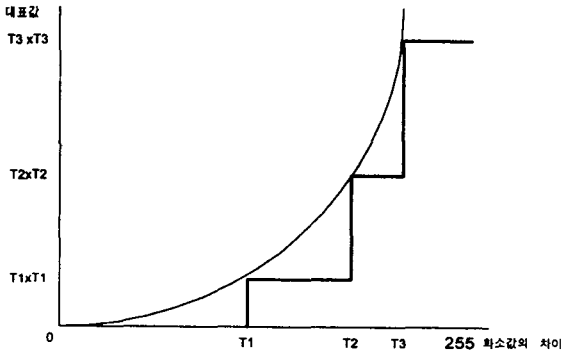



그림 5. 2차 비용 함수를 이용한 화소 분류 방법
Fig. 5. Pixel classification method using the second order cost function

$$
\begin{aligned}
T(x,y,i,j) = \;& 0, \text{ if } 0 \le [I(x,y,t) \\
& - I(x+i,y+j,t-\tau)]^2 \le T_1 \\
T(x,y,i,j) = \;& T_1^2, \text{ if } T_1 < [I(x,y,t) \\
& - I(x+i,y+j,t-\tau)]^2 \le T_2 \\
T(x,y,i,j) = \;& T_2^2, \text{ if } T_2 < [I(x,y,t) \\
& - I(x+i,y+j,t-\tau)]^2 \le T_3 \\
T(x,y,i,j) = \;& T_3^2, \text{ if } T_3 < [I(x,y,t) \\
& - I(x+i,y+j,t-\tau)]^2
\end{aligned}
\tag{11}
$$

위 식은 제안하는 방법 3 으로 화소들을 2-비트로 분류한 예이다. 그 다음 제안하는 방법 2와 마찬가지 방법으로 움직임 벡터를 구한다. 즉, 움직임 벡터의 좌표는 다음과 같다.

$$G_m(d_h, d_v) = \min\{\sum_x \sum_y T(x,y,i,j)\} \tag{12}$$

### 5. 임계치 결정

본 논문에서 제안하는 화소 분류 방법에 있어서 적절한 임계치를 결정하여 화소들을 잘 분류하는 것은 매우 중요한 요소이다. 이것은 곧 임계치의 결정이 복호된 화질에 미치는 영향이 매우 크다는 것을 의미하는 것이다. 본 논문에서는 임계치를 결정하는 방법으로 최적 양자화 레벨 결정 방법을 사용하였다. 임계치를 주어 화소들에 대표값을 부여하는 과정이 양자화 알고리즘과 유사하기 때문이다. 최적의 임계치를 찾기 위해, 평균 제곱 오차(mean square error)를 최소화하는 관점에서 최적의 양자화를 행하는 로이드-맥스(Lloyd-Max) 최적 양자화 알고리즘을 사용하였다.

현재 블록과 이전 블록의 화소의 차이값들을 입력치로 받아 들여 로이드-맥스 알고리즘을 수행하면 평균 제곱 오차를 최소화하는 reconstruction level과 decision level들이 결정되게 되는데 이 decision level(di)들을 바로 화소 분류의 임계치(Ti)로 사용하였다. 제안하는 방법에서 최적의 임계치를 찾기 위한 로이드-맥스 알고리즘중 PDF (probability density function)는 특정한 PDF를 사용하지 않고 SNR을 최대화하는 최적의 reconstruction level과 decision level들을 각 프레임별로 반복해서 계산함으로써 최적의 양자화 임계치를 구하였다. 이런 방법의 경우 계산량이 많아지고 수행 시간이 늘어나는 단점이 있지만 정확한 임계치를 구하지 않는 경우 performance의 차이가 많이 나는 경향이 있다. 향후 좀더 일반적인 임계치를 구하는 방법이 연구되어야 할 것이다. 그림 6은 2-비트 화소 분류 방법의 임계치 결정 예이다.

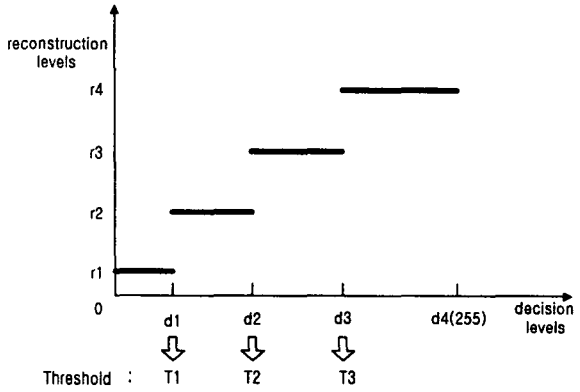



그림 6. 최적 양자화 방법으로 임계치 결정
Fig. 6. Threshold decision using the optimal quantization method

## Ⅳ. 제안한 알고리즘과 다른 고속 알고리즘과의 결합

제안하는 새로운 블록 정합 기준은 기존의 여러 가지 고속 알고리즘과 결합하여 좀더 빠르고 간단한 하드웨어 구조를 가지는 움직임 추정을 할 수 있다. 여러 가지 고속 알고리즘 중에서 대표적인 고속 알고리즘인 3단계 탐색 방법(three step search)과 화소 부표본화 방법(pixel subsampling method)을 제안하는 알고리즘과 결합시켜

실험해 보았다.

### 1. 3단계 탐색 방법과 제안하는 방법과의 결합

3단계 탐색 방법은 탐색 간격을 단계별로 줄여가며 탐색하는 방식이다.[12] 먼저 탐색 간격을 4로 하여 중심을 포함한 9개의 위치에서 정합 에러를 구한다. 에러가 가장 작은 위치를 중심으로 하여 이번에는 탐색 간격을 2로 하여 9개의 위치에서 정합 에러를 구한다. 마지막으로 역시 에러가 가장 작은 위치에서 탐색 간격을 1로 줄여 주위의 9개의 위치에서 정합 에러를 구한 후 그 중 가장 작은 에러를 갖는 위치를 움직임 벡터로 결정한다.

3단계 탐색 방법 중 각 단계별로 정합 에러를 구하는 과정에서 블록 정합 기준으로 평균 제곱 오차나 평균 절대 오차를 쓰는 대신 제안하는 새로운 블록 정합 기준인 화소 분류 방법을 사용하였다. 하드웨어 부담이나 계산량의 감소에도 불구하고 화질은 대체로 우수한 편이었다.

제안하는 방법 2 즉, 2-비트 화소 분류 방법만 3단계 탐색 방법과 결합하여 실험해 보았으나, 다른 두 가지의 방법과의 결합에서도 좋은 성능을 나타낼 것이 예상된다.

### 2. 화소 부표본화 방법과 제안하는 방법과의 결합

화소 부표본화 방법과 2 비트 화소 분류 방법을 결합시켜 실험하였다. 화소 부표본화 방법은 움직임 추정시 복잡도(complexity)를 줄이기 위해 블록 정합시 사용되는 화소들의 개수를 줄이는 방법이다.[15] 움직임 추정의 정확도를 감소시키지 않기 위해서 그림 3-5와 같은 방법으로 화소 부표본화를 행한다. 그림 7은 8 × 8 블록에서 각각의 화소들을 a, b, c, d 네 가지의 패턴으로 부표본화(subsampling)한 것이다. 패턴 A를 모든 a 화소들로 구성된 부표본화 패턴이라 하고, 마찬가지로 B, C, D 패턴을 각각 모든 b, c, d 화소들로 이루어진 부표본화 패턴이라 한 후 정합 에러 계산시 이 네 가지의 패턴 중 한 가지만으로 계산을 수행함으로써 전역 탐색 방법보다 1/4의 계산량 감소를 가져올 수 있다. 패턴 A, B, C, D로 블록 정합 에러를 구한 후 네 패턴에서 가장 정합 에러가 작은 움직임 벡터를 각각 하나씩 추출한 후 그 네 개의 벡터에 대해서는 전역 탐색 방법으로 정합 에러를 계산하여 그중 가장 에러가 작은 것을 최종적인 움직임 벡터로 결정한다.

화소 부표본화 방법 중 패턴 A, B, C, D 별로 정합 에러가 가장 작은 움직임 벡터를 각각 하나씩 추출하는 과정에서 제안하는 화소 분류 방법을 썼으며, 그 네 개의 벡터중 하나를 전역 탐색 방법으로 찾는 과정에서는 평균 절대 오차(MAE)를 사용하였다. 실험 결과는 프레임별로 대체로 평균 절대 오차(MAE)를 이용한 전역 탐색 방법과 비슷하거나 오히려 좋을 때도 있어 제안한 방법이 우수한 성능을 나타냄을 알 수 있었다.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |

그림 7. 화소 부표본화 방법
Fig. 7. Pixel subsampling method

## V. 실험 및 고찰

표 1. Football 영상의 평균 PSNR
Table 1. Average PSNR of the Football sequence

| Football 영상(1~40 프레임) | 평균 PSNR(dB) |
|---|---|
| MSE | 23.0565 |
| MAE | 22.8512 |
| 1-비트 화소 분류 방법 | 22.3305 |
| 2-비트 화소 분류 방법 | 22.8923 |
| 3-비트 화소 분류 방법 | 22.9345 |
| 3단계 탐색 방법 | 22.4293 |
| 3단계 + 2-비트 화소 분류 방법 | 22.3597 |
| 화소 부표본화 방법 | 22.8337 |
| 화소 부표본화 + 2-비트 화소 분류 방법 | 22.8347 |

표 2. Flower garden 영상의 평균 PSNR
Table 2. Average PSNR of the Flower garden sequence

| Flower garden 영상(1~40 프레임) | 평균 PSNR(dB) |
|---|---|
| MSE | 23.7794 |
| MAE | 23.6593 |
| 1-비트 화소 분류 방법 | 23.3527 |
| 2-비트 화소 분류 방법 | 23.6695 |
| 3-비트 화소 분류 방법 | 23.7022 |
| 3단계 탐색 방법 | 22.2720 |
| 3단계 + 2-비트 화소 분류 방법 | 22.0251 |
| 화소 부표본화 방법 | 23.6287 |
| 화소 부표본화 + 2-비트 화소 분류 방법 | 23.6050 |

실험에 사용한 영상은 Football영상과 Flower garden영상, 그리고 Table tennis영상이다. Football영상은 움직임이 크고 많이 발생하며, Flower garden영상은 복잡한 텍스춰(texture)가 많은 영상이고, Table tennis영상은 움직임이 비교적 적은 영상이다. 화면의 크기는 모두 352 × 240 크기의 SIF(Source Input Format)화면이며, 1프레임부터 40프레임까지의 화면을 대상으로 실험하였다.

표 3. Table tennis 영상의 평균 PSNR
Table 3. Average PSNR of the Table tennis sequence

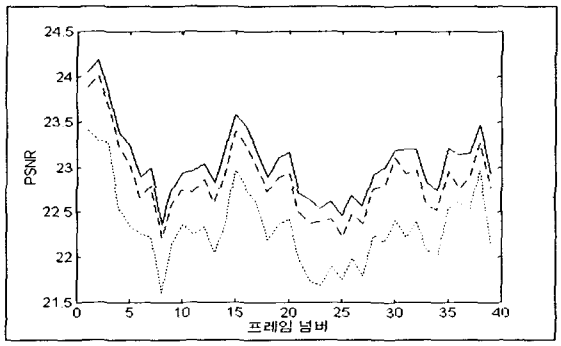
| Table tennis 영상(1~40 프레임) | 평균 PSNR(dB) |
|---|---|
| MSE | 29.8459 |
| MAE | 29.6776 |
| 1-비트 화소 분류 방법 | 29.2133 |
| 2-비트 화소 분류 방법 | 29.6991 |
| 3-비트 화소 분류 방법 | 29.7480 |
| 3단계 탐색 방법 | 28.6873 |
| 3단계 + 2-비트 화소 분류 방법 | 28.6494 |
| 화소 부표본화 방법 | 29.6634 |
| 화소 부표본화 + 2-비트 화소 분류 방법 | 29.5937 |

(a)

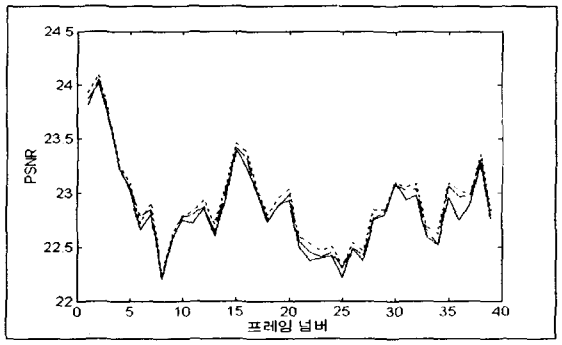
(b)

(c)

그림 8. 실험에 사용된 영상들 (a) Football (b) Flower garden (c) Table tennis
Fig. 8. Sequences used in the experiments (a) Football (b) Flower garden (c) Table tennis




(a) MSE ——— , MAE - - - - - , 1 비트 화소 분류 ·····




(b) MAE ——, 2 비트 화소 분류 - - - -, 3 비트 화소분류 ·····

그림 9. Football 영상의 프레임별 PSNR 변화 (a) MSE, MAE, 1 비트 화소 분류 (b) MAE, 2 비트 화소 분류, 3 비트 화소 분류
Fig. 9. PSNR results of the Football sequence (a) MSE, MAE, 1 bit pixed classification (b) MAE, 2 bit pixel classification, 3 bit pixel classification

그림 8은 실험에 사용한 영상들의 특정 프레임을 보여주고 있다. 탐색하는 블록의 크기는 16 × 16 매크로 블록이며, 탐색 영역의 범위는 x, y 방면 모두 -16~15 화소 단위로 하였다. 화질의 측정은 PSNR(Peek Signal to

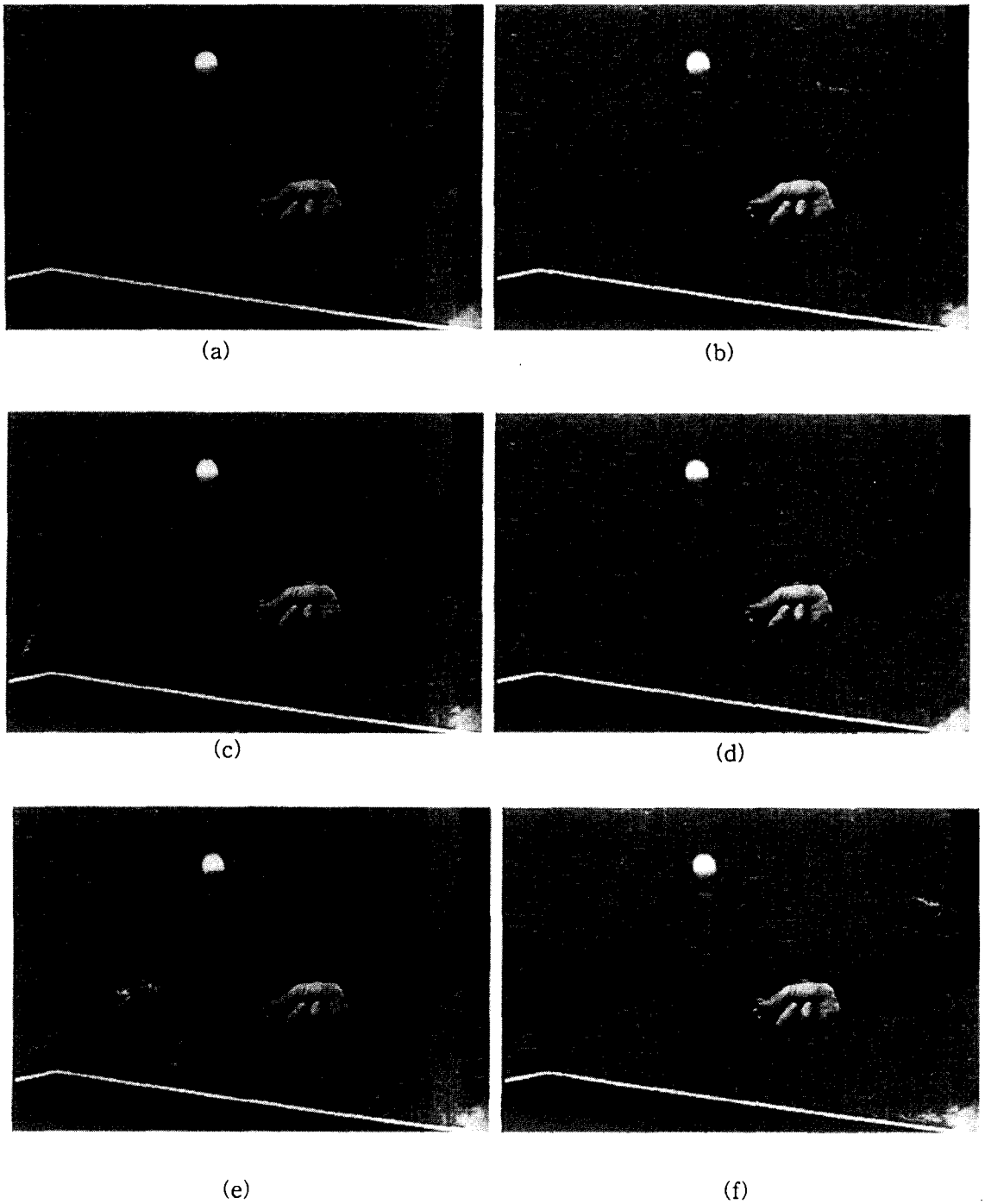



그림 10. Table tennis 영상의 2번째 프레임 (a) 원 영상 (b) MSE (c) MAE (d) 1-비트화소 분류 방법 (e) 2-비트 화소 분류 방법 (f) 3-비트 화소 분류 방법

Fig. 10. 2nd frame of the Table tennis sequence (a) original sequence (b) MSE (c) MAE (d) 1-bit pixel classification method (e) 2-bit pixel classification method (f) 3-bit pixel classification method

Noise Ratio)을 이용한 객관적 화질 평가와 눈으로 직접 화면을 관찰하는 주관적 화질 평가를 통해 하였다. 실험결과를 보면 2-비트 화소 분류 방법은 평균 절대 오차(MAE)를 이용한 전역 탐색 방법 보다 대체로 비슷하거나 평균적으로 높은 성능을 보였으며, 3-비트 화소 분류 방법은 이 보다 더 높은 화질 향상을 가져왔다. 그림 9에서 Football 영상에 대해 실험한 여러 알고리즘들의 PSNR변화에 대해 볼 수 있다. 이에 비해 제안하는 방법 3, 즉 2차 비용함수를 이용한 화소 분류방법은 평균 절대 오차를 이용한 방법보다는 월등한 화질을 보였으며, 거의 평균 제곱 오차(MSE)를 이용한 전역 탐색 방법에 근접하는 성능을 보였다. 또 제안한 방법과 여러 고속 알고리즘 중 3단계 탐색 방법(three-step search)[3]과 화소 부표본화 방법(pixel subsampling)[2]을 결합한 모의 실험을 행하였다. 이 결과에서는 움직임이 적은 Table tennis 영상이 다른 두 영상에 비해 좀 더 나은 성능을 보임을 알 수 있다. 복원된 영상을 통한 주관적 화질 평가도 Football영상과 Flower garden영상, 그리고 Table tennis영상에 대해 비교해 보았다. 그중에서 그림 10에 보인 Table tennis영상에 대한 여러 알고리즘들의 복원된 영상들에서 보듯이 탁구라켓의 옆면이나 왼손 주위등을 보면 제안된 화소 분류를 이용한 탐색 방법들이 MAE를 이용한 방법보다 좀 더 자연스러운 화면을 보이고 있음을 알 수 있다.

## VI. 결 론

본 논문에서는 움직임 추정 부호화 방법 중 블록 정합 알고리즘에 있어 블록 정합 에러를 구해 움직임 벡터를 구하는 새로운 블록 정합 기준을 제안하였다. 기존의 1-비트 화소 분류 방법에서 변형된 새로운 비용 함수를 이용한 2-비트, 3-비트 화소 분류 방법과, 화소 분류시 사용되는 임계치를 결정하는 방법을 제안하였다. 또한 평균 제곱 오차에 근사화하는 모델링(modeling)인 2차 비용함수를 이용한 화소 분류 방법도 제안하였다. 이 새로운 블록 정합 기준은 기존의 블록 정합 기준인 평균 제곱 오차나 평균 절대 오차에 비해 각 화소들이 움직임 벡터를 결정하는데 중요한 역할을 함으로써 좋은 결과를 가져오게 하며, 간단한 하드웨어 구조와 계산량의 감소로 실시간 시스템의 구현을 용이하게 해주는 장점이 있다. 또한 기존의 다른 고속 움직임 추정 알고리즘과의 결합에서도 우수한 성능을 보여 여러 가지 다른 응용 분야에서도 널리 쓰일 수 있을 것이다. 또한 화소 분류에서 쓰이는 임계치는 장면 전환 검출에도 그대로 쓰일 수 있는 장점이 있다. 향후 과제로는 2차 비용함수를 이용한 화소 분류 방법에서 좀 더 적절한 임계치를 결정하는 방법을 찾고, 탐색 블록의 크기와 탐색 영역의 크기를 변화시켜 실험하는 것과, 제안하는 방법의 부화소 단위 움직임 추정 기법에의 응용 등이 있겠다.

## 참 고 문 헌


[1] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, no. 12, pp. 1799-1808, Dec. 1981.

[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *in Proc. Nat. Telecommun. Conf.*, no. 29-Dec. 3, 1981, pp. G5.3.1-5.3.5.

[3] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Transactions on Communications*, vol. COM-33, no. 8, pp. 888-896, Aug. 1985.

[4] Q. Wang and R. J. Clake., "Motion compensated sequence coding using image pyramids," *Electronics Letters*, 26(9):575-6, Apr 1990.

[5] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimation of Block Motion Vectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148-157, Apr. 1993.

[6] H. Gharavi and M. Mills, "Blockmatching Motion Algorithms - New Results," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 5, pp. 649-651, May 1990.

[7] Y. Q. Shi and X. Xia, "A Thresholding Multi-resolution Block Matching Algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 437-440, Apr. 1997.

[8] S. Lee and S. I. Chae, "Motion Estimation Algorithm using Low Resolution Quantization," *Electronics Letters*, vol. 32 no. 7, pp. 647-648, 28th Mar. 1996.

[9] J. Lu and M. L. Liou, "A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation," *IEEE Transactions on Circuits and Systems for video technology*, vol. 7, no. 2, pp. 429-433, Apr. 1997.

[10] F. Dufaux and F. Moscheni, "Motion Estimation Techniques for Digital TV : A Review and a New Contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858-876, Jun. 1995.

[11] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," *ELSEVIER Signal Processing*, vol. 11, no. 4, pp. 387-404, Dec. 1986.

[12] H. G. Musmann, P. Pirsch and H. J. Gralleer, "Advances in picture coding,: *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523-548, Apr. 1985.

## 저 자 소 개


### 조 한 욱

1996. 2. 한양대학교 공과대학 전자통신공학과 졸업 (공학사)
1998. 2. 한양대학교 대학원 전자통신공학과 졸업 (공학석사)
1998. 3. ~ 현재 삼성전자 근무
주관심분야:영상 통신 및 압축, 영상처리, 디지털 신호처리 등

### 서 정 욱

1996. 2. 한양대학교 공과대학 전자통신공학과 졸업 (공학사)
1998. 2. 한양대학교 대학원 전자통신공학과 졸업 (공학석사)
1998. 3. ~ 현재 삼성전자 근무
주관심분야:영상 통신 및 압축, 영상처리, 디지털 신호처리 등

### 정 제 창

1980. 2. 서울대학교 공과대학 전자공학과 졸업 (공학사)
1982. 2. 한국 과학기술원 전기전자공학과 졸업 (공학석사)
1990. 8. 미시간대학교(앤아버) 전기공학과 졸업 (공학박사)
1982. 2. ~ 1986년 7월 한국방송공사 기술연구소 연구원 (뉴미디어 연구개발)
1990. 9. ~ 1991년 1월 미시간대학교(앤아버) Post-dotorial Research Fellow
1991. 2. ~ 1995년 2월 삼성전자 멀티미디어 연구센터 신호처리연구소 수석연구원
(HDTV 및 멀티미디어 연구개발)
1995. 3. ~ 현재 한양대학교 전자통신공학과 교수
주관심분야:영상 및 음성 압축, 영상처리, 디지털 신호처리, 웨이블릿 변환,
MPEG-4/JPEG2000 등 국제표준