다양한 커널을 이용한 이진 블록 정합 움직임 추정

*신욱진 **이혁 ***정제창

한양대학교

*wookja27@gmail.com; **yomuto@gmail.com; ***jjeong@ece.hanyang.ac.kr

1BT Motion Estimation using Adaptive Kernels

*Shin, Wook-Jin **Lee, Hyuk ***Jeong, Jechang

Hanyang University

요약

현대 사회에서 영상 콘텐츠 (contents)의 사용량이 급증함에 따라 영상압축 기술은 이동통신이나 DMB 등의 시스템에 필수적인 기술이 되었으며 이에 따라 MPEG-x와 H.26x 등 국제적인 표준들이 존재한다. 전역 탐색 알고리듬은 주어진 검색 범위 (search range) 내에서 모든 후보들의 변위의 에러 기준에 따라 최솟값을 이용해 위치를 검색하는 알고리듬이다. 그러나 전역 탐색 알고리듬은 각 화소에 대해 엄청난 양의 계산 로드를 가지며 이로 인해 심각한 문제를 발생시키는 단점이 있다. 1비트 변환 (one-bit transform) 을 이용한 고속 움직임 추정 알고리듬은 참조 프레임과 현재 프레임을 1비트, 즉 0 또는 1만 갖는 값으로 변환하는데, 이에는 exclusive-OR 연산을 통해 블록 매칭 에러 (block matching error)를 계산하는 과정과 변환하는 과정이 포함된다. 본 논문에서는 다양한 커널 (kernel)들을 이용한 1비트 변환과 움직임 추정에 대해 다루었으며, 기존에 있었던 1비트 변환에 이용된 커널과는 다른 다양한 커널을 이용한 움직임 추정 결과들을 비교해봄으로써 화질열화를 최소로 하는 커널을 찾는 것에 대해 연구했다.

1. 서론

동영상 압축이란 동영상이 가지고 있는 중복성 (redundancy)를 제거하여 정보량을 줄이는 것이다. 움직임 추정을 위해 폭넓게 사용되는 전역 탐색 알고리듬은 많은양의 계산 로드 (load)를 갖는 단점이 있다. 이의 개선을 위해 기존에 제안된 1비트 변환 알고리듬은 프레임 (frame)을 1비트 프레임으로 변환하기 위해 17×17 커널을 사용한다. [1]

$$K=\frac{1}{25}\begin{bmatrix}
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1
\end{bmatrix}$$

그림 1 1비트 변환을 위해 이용된 기존의 커널

그림 1과 같은 커널을 이용해 1비트 변환된 현재 프레임과 참조 프레임에 배타적 논리합 (Exclusive-OR)을 취하여 움직임 벡터 (motion vector)를 얻게 되는데, 본 연구에서는 더 정확한 움직임 벡터를 얻을 수 있는 커널을 찾기 위해서 다양한 형태의 커널들에 대한 실험을 하였으며, 영상들의 특성에 따라 성능이 다르게 나타나는 점에 대해 분석한다.

2. 본론

2.1 움직임 벡터를 얻기 위한 알고리듬의 원리

움직임 벡터를 얻기 위해서는 프레임을 여러 개의 매크로블록으로 나누어 현재 프레임과 참조 프레임을 비교해야 한다. 따라서 첫 번째 작업은 프레임을 일정크기 (16×16)의 매크로 블록으로 나누는 것이다. 본 논문의 실험에서는 해상도 352×288, 300개 프레임의 영상을 주로 사용하였으며, 이 경우 16×16 크기의 매크로블록들로 나누었을 경우 한 프레임 당 가로와 세로 각각 22개, 18개씩의 매크로블록들로 나누어지게 된다.

다음 단계는 현재 프레임의 매크로블록들이 참조 프레임의 어느 위치에서 이동된 것인지, 즉 움직임벡터를 찾는 단계이다. 이 때 프레임 전체의 범위 내에서 검색하면 계산 로드가 너무 크기 때문에 일정범위 내에서 검색하는 것이 효율적이며 특히 이 논문에서는 16화소의 거리를 검색 범위로 사용하였다. 다시 말해 현재 프레임의 첫 번째 매크로블록에 대한 움직임 벡터를 찾기 위해서는 그 블록이 참조 프레임의 첫 번째 매크로블록 위치로부터 상하좌우로 15화소만큼의 거리로

이동한 범위 내에서 상관 (correlation)이 가장 큰 지점을 찾아 그 거리를 움직임벡터로 판정하는 것이다.

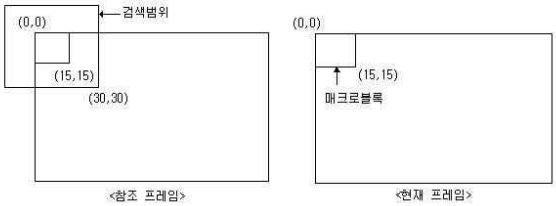

그림 2 현재 프레임과 참조 프레임의 매크로블록과 검색범위의 표현

그림 2에서와 같이 현재 프레임의 매크로블록과 가장 상관 값이 높은 매크로블록을 참조 프레임의 검색 범위 내에서 찾아 그 변위를 움직임 벡터로 판정하게 된다.

블록간의 상관 값은 전역 탐색 알고리듬의 경우 두 매크로블록 내 화소 값의 차분치에 절댓값을 취하여 블록 내 모든 화소에 대하여 더한 값 (Sum of Absolute Difference, SAD)으로 한다. 즉, 가로와 세로 길이가 b인 두 매크로블록 u, v에 대하여 두 블록의 상관 값을 다음의 수식으로 표현할 수 있다.

$$\|u,v\| = \frac{1}{b^2}\sum_{i,j}^{b}\left|u_{i,j} - v_{i,j}\right| \quad (1)$$

검색 범위 내 모든 위치에서 식 (1)을 통해 계산한 SAD 값들 중 그 값이 가장 작은 위치로부터의 변위가 움직임 벡터가 된다.

2.2 1비트 변환 을 이용한 알고리듬

1비트 변환은 다양한 값을 갖는 화소 값들을 0이나 1만의 값을 갖도록 만드는 변환이다. 서론의 그림 1과 같은 커널을 이용하여 각 화소에 대해 커널을 적용한 값이 원래의 화소 값보다 크면 1, 작으면 0으로 변환하게 된다. 그림 2는 원본 영상과 1비트 변환 후의 영상의 예를 보여준다. [2]

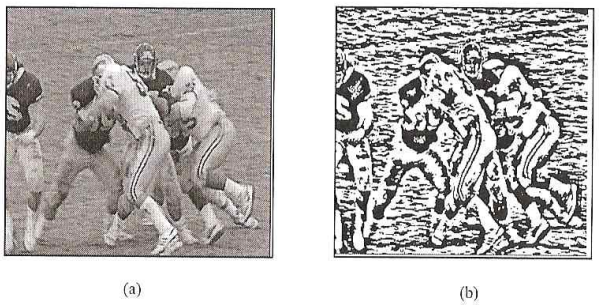

그림 2 (a) 원래의 영상 (b) 1비트 변환 후의 영상

1비트 변환에 의한 블록 정합 알고리듬의 경우, 변환을 통해 0과 1만의 값을 갖게 된 두 프레임에서의 매크로블록 간 상관 값을 계산하는 데에는 전역 탐색 알고리듬에서처럼 많은 계산 량을 요구하지 않는다. 화소 값을 비교함에 있어서 두 입력 값이 같으면 0, 다르면 1을 출력하는 exclusive-OR operation을 이용하여 화소마다의 출력을 모두 더한 값이 가장 최소인 거리를 움직임벡터로 판정한다.

블록 정합 움직임 추정에 있어서 전역 탐색 알고리듬의 경우에 덧셈과 절댓값을 취하는 계산 량이 엄청난 것에 비해, 1비트 변환된 프레임에서는 단순히 입력에 대한 결과 값을 룩업테이블 (Look-Up Table, LUT)에 미리 저장해둠으로써 계산 량과 소요 시간을 크게 단축시킬 수 있다.

2.3 다양한 커널을 이용한 움직임벡터 추정

1비트 변환을 통한 블록 정합 움직임 추정은 전역 탐색 알고리듬에 비해 속도와 계산 량에서 많은 이득이 있지만, 정확도에 있어서는 적지 않은 손실을 가지고 있다. 그렇기 때문에 기존의 커널을 이용한 1비트 변환으로 인한 화질 열화를 최소화하려는 목표에서 더 좋은 성능을 가진 커널을 찾기 위한 연구를 했다.

추정된 움직임벡터를 적용한 영상이 원래 영상과 가장 비슷하려면, 즉 화질열화를 최소화하기 위해서는 1비트 변환 후의 영상이 1비트 변환 전의 영상을 가장 잘 표현하고 있어야 한다. 그래야 블록 정합 움직임 추정 과정에서 정확한 움직임 벡터를 찾아낼 수 있기 때문이다. 1비트 변환 후의 영상이 원래의 영상의 특징을 가장 잘 표현하려면 그 영상 내에 존재하는 모서리 (edge)들을 찾아내는 것이 가장 유리하다. 모서리를 찾아내기 위해서는 고역통과필터 (High-Pass Filter, HPF)를 이용해야 하지만, 기존의 연구에 의하면 1비트 변환에 이용되는 커널은 고주파잡음(high-frequency noise)을 피하기 위해 대역통과필터 (Band-Pass Filter, BPF)를 이용하는 것으로 알려져 있다. 따라서 모서리를 잘 찾아낼 수 있는 커널을 찾기보다는 기존의 커널 내에서 1의 값을 갖는 25개의 위치들을 다른 배열을 갖도록 바꾸어 실험을 했다.

기존에 사용되는 커널은 17×17 크기로, 매크로블록의 크기 (16×16)와 거의 같다. 블록 정합 움직임 추정 과정에서 매크로블록과 같은 크기 내에서의 값들만 비교하기 때문에 추정에 영향을 주는 값들이 매크로블록 크기 내의 값들이고, 매크로블록을 포함할 수 있으면서 변환되는 자기 자신의 위치를 포함할 수 있는 최소의 홀수인 17×17 크기가 최적인 것이다.

기존의 커널은 변환되는 위치의 화소를 포함한 25개의 위치에 있는 값들의 일종의 산술평균값을 구하여 원래의 화소 값과의 비교를 통해 변환하는데, 25개의 위치는 서로 직교하게 상하, 좌우 대칭을 이루고 있다. 하지만 매크로블록 내 화소 값들의 평균을 가장 균등한 방향성을 갖는 화소 값들로 측정하려면 25개의 위치가 가운데 점을 중심으로 원과 가장 닮은 형태를 가져야 한다. 따라서 25개의 위치를 원에 가장 가깝게 바꿔보면서 다양한 영상들에 대해 나타나는 성능을 실험 하였다. 성능의 척도로는 PSNR (Peak Signal-to-Noise Ratio)을 이용하였으며, 식은 다음과 같다.

$$PSNR = 10\log[\sum_{i,j}(\frac{255}{F_{i,j}})^2] \quad dB \quad (2)$$

식 (2)에서 $F_{i,j}$는 원래의 영상과 움직임 벡터를 적용한 영상의 화소 값의 차분치를 이용해 계산하는 평균 오차 제곱 합

(Mean-Squared Error)을 의미한다.

특정 이론의 수식에 따라 커널을 설계한 것이 아니기 때문에 실험적으로 커널 내 값들의 배열과 가중치를 변화시키면서 결과 분석을 통한 반복적인 실험을 했고, 여러 종류의 영상에 대한 평균 PSNR이 높아지는 방향으로 커널을 수정하며 실험을 했다. 커널을 아주 작은 부분마다 변화시키면서 실험했기 때문에 그 중에서 결과분석에 의미 있는 특징을 나타낸 몇 가지 커널을 그림 3에 나타낸다.

$$K=\frac{1}{37}\begin{bmatrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\0&0&0&0&1&0&1&0&0&0&1&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&1&0&1&0&0&1&0&0&1&0&1&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&1&0&0&0&1&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{bmatrix}$$

(a) 커널 1

$$K=\frac{1}{25}\begin{bmatrix}1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\end{bmatrix}$$

(b) 커널 2

$$K=\frac{1}{29}\begin{bmatrix}0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\end{bmatrix}$$

(c) 커널 3

$$K=\frac{1}{26}\begin{bmatrix}1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&0&0&1&0&0&0&2&0&0&0&1&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\end{bmatrix}$$

(d) 커널 4

$$K=\frac{1}{36}\begin{bmatrix}1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\0&1&0&0&0&2&0&0&0&0&0&2&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&0&0&2&0&0&0&4&0&0&0&2&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&2&0&0&0&0&0&2&0&0&0&1&0\\0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\end{bmatrix}$$

(e) 커널 5

그림 3 1비트 변환에 쓰인 5개의 커널

그림 3에 나타난 5개의 커널과 기존 커널을 이용한 이진 블록 정합 움직임 추정, 그리고 전역 탐색 알고리즘에 의한 움직임 추정 결과를 표 1에 나타내었다. 움직임 추정은 각기 다른 9가지의 영상에 대해 이루어졌다.

CIF 영상은 352×288의 프레임 크기를 갖는 영상이고 QCIF 영상은 가로, 세로의 크기가 각각 CIF 영상의 반인 176×144의 프레임 크기를 갖는 영상이다. 프레임의 크기를 기준으로 비교해 보았을 때에는 커널별로 성능에 큰 차이를 나타내지 않았다.

1번 커널에는 커널 내 존재하는 값의 개수가 원래의 25개에서 크게 늘어난 37개가 원의 형태로 배열되어 있다. 이 경우에는 모든 종류의 영상에 대해서 성능이 떨어지는 결과가 나타났다.

4번 커널의 경우 기존 커널의 형태를 거의 유지하는 한편 4개의 위치만을 한 칸씩 중앙을 향해 이동시켜 원의 형태에 가까워지도록 바꿨고, 보다 더 모든 방향으로 대칭의 모습에 가까워지면서 성능이 향상되었다. 커널의 꼭짓점 부분에 위치하는 4개의 값만 제외하면 원의 형태와 거의 비슷하지만, 꼭짓점 부분의 값을 제외하거나 3번 커널에서처럼 오히려 꼭짓점 근처에 두 개씩의 값을 배치하였을 경우 성능이 다시 나빠지는 결과가 나왔다. 원의 형태를 유지하는 것도 중요하지만 커널에 존재하는 값의 위치가 특정 부분에 집중되지 않고 모든 부분에 최대한 균등한 거리를 두고 존재하는 것 역시 중요하다는 것을 알 수 있다.

영상 종류	알고리듬에 따른 PSNR (dB)						
	전역탐색	기존커널	실험커널1	실험커널2	실험커널3	실험커널4	실험커널5
foreman (CIF)	31.497	29.285	29.182	29.274	29.344	29.277	29.371
children (CIF)	29.116	28.245	28.072	28.228	28.217	28.231	28.208
mobile (CIF)	24.491	24.133	24.112	24.133	24.135	24.134	24.142
news (CIF)	34.975	32.060	31.526	32.151	31.931	32.165	32.320
stefan (CIF)	23.402	22.583	22.392	22.567	22.56	22.564	22.517
table (CIF)	30.442	28.898	28.521	28.870	28.847	28.874	28.762
news (QCIF)	33.475	32.531	32.314	32.655	32.622	32.669	32.671
stefan (QCIF)	24.115	23.695	23.660	23.707	23.687	23.704	23.707
table (QCIF)	29.169	27.995	27.873	28.046	27.926	28.047	27.925
averge	28.965	27.714	27.517	27.737	27.697	27.741	27.736

표 1 다양한 커널을 이용한 블록 정합 움직임 추정 성능 비교

4번과 5번 커널은 배열은 2번과 같이 유지하되 커널 중앙의 위치와 주변의 가까운 위치에 가중치를 두는 실험을 위해 사용되었다. 중앙 위치에 4배, 그 바로 주변 위치에는 2배를 하여 실험을 했을 때에는 오히려 성능이 나빠졌지만 가운데 위치에만 2배를 하여 성능을 측정 했을 경우 가중치를 두지 않은 2번 커널보다 평균적으로 높은 PSNR을 보였다. 이는 기존에 1비트 변환에 이용되었던 커널보다 약 0.027 dB 향상된 수치이다.

3. 결론

실험에 이용된 다양한 커널들 중 최종적으로 가장 나은 성능을 보인 것은 4번 커널이지만 4번 커널에서 중앙에 가중치를 두지 않은 형태인 2번 커널을 고려해보자. 이는 상하, 좌우로 일정한 거리를 두고 격자 형태로 배열된 기존의 커널과 배열상의 차이만을 가진다. 2번 커널은 다양한 영상들을 고려하여 평균적으로 볼 때에는 기존의 커널보다 나은 성능을 보였지만, 일부의 영상에서는 성능이 오히려 떨어지기도 했다.

그 이유는 움직임 추정에 이용되는 영상의 특성에 있다. 2번 커널이 기존의 커널에 비해 가장 큰 화질 향상을 보인 영상은 news (CIF)이고, 가장 큰 화질 열화를 보인 영상은 table (CIF)이다. 이 두 영상을 비교해볼 때, news (CIF) 영상은 상대적으로 원형이나 대각선 움직임보다는 좌우 혹은 상하 움직임이 많은 영상이고, table (CIF) 영상은 반대로 좌우나 상하 움직임 보다는 원형의 움직임이 많은 영상이다. 이에 비추어 볼 때 커널의 형태가 원에 가까울수록 대각선이나 원운동의 영상에 대해 좋은 성능을 나타낸다는 것을, 그리고 커널의 형태가 격자 형태에 가까울수록 좌우 혹은 상하운동이 많은 영상에 대해 좋은 성능을 나타낸다는 것을 알 수 있다. 이는 커널의 형태가 1비트 변환 후의 영상의 형태에 비슷한 영향을 준다는 것을 의미한다.

모든 영상에 대해 일반적으로 사용하려면 이번 실험에서 제안된 2번이나 4번 커널을 이용하는 것이 좋겠지만, 영상의 성격에 따라서는 기존에 사용되어 온 커널이 더 효과적일 수도 있다. 따라서 1비트 변환을 이용한 움직임 추정으로 인한 화질 열화를 최소화하기 위해서는 먼저 1비트 변환을 적용하려는 영상의 성격을 파악하여 그 특성의 영상에 가장 잘 맞는 커널을 적절히 선택하여 사용해야 할 것이다.

참고 문헌

[1] Balas Natarajan, Vasudev Bhaskaran, and Konstantinos Konstantinides, "Low-Complexity Block-Based Motion Estimation via One-Bit Transforms", IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 4, August 1997.

[2] Farhan Hussain, "비디오 압축에 있어서 빠른 움직임 추정을 위한 개선된 1비트 변환 기법", 한양대 대학원 졸업논문, 2009.