論文99-36S-11-13

탐색 영역 부표본화 및 이웃 화소간의 차를 이용한 고속 전역 탐색 블록 정합 알고리듬

(Fast Full Search Block Matching Algorithm Using The Search Region Subsampling and The Difference of Adjacent Pixels)

鄭 元 植 *, 李 法 基 *, 李 京 桓 *, 崔 正 鉉 *, 金 慶 圭 *,
金 德 奎 *, 李 健 一 *

(Won-Sik Cheong, Bub-Ki Lee, Kyeong-Hwan Lee, Jung-Hyun Choi, Kyeong-Kyu Kim, Duk-Gyoo Kim, and Kuhn-Il Lee)

요 약

본 논문에서는 탐색 영역에서의 탐색점 부표본화와 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용한 고속 전역 탐색 블록 정합 알고리듬을 제안하였다. 제안한 방법에서는 각 탐색점에서의 평균 절대치 오차 (mean absolute difference; MAD) 값의 최소 범위를 이웃 탐색점에서의 MAD와 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용하여 구한 뒤, 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 이때, 현재 탐색점에서의 MAD의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAD를 사용한다. 그러므로 제안한 방법에서는 먼저, 탐색 영역에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 부표본화 된 탐색점에 대하여 블록 정합을 행하여 MAD를 구한다. 그리고, 나머지 탐색점에 대하여서는 각 탐색점의 MAD 값의 최소 범위를 부표본화 된 탐색점에서의 MAD와 현재 블록 내의 화소들의 이웃 화소간의 화소 값의 차를 이용하여 구한 뒤, 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행하였다. 즉, 제안한 방법에서는 각 탐색점에서 MAD의 최소 범위를 이용하여 블록 정합이 필요한 탐색점 수를 줄임으로써 전역 탐색 블록 정합 알고리듬 (full search block matching algorithm; FSBMA)과 동일한 성능을 유지하면서도 고속으로 움직임을 추정할 수 있었다. 모의 실험을 통하여 제안한 방법이 FSBMA와 동일한 성능을 유지하면서도 많은 계산량의 감소를 얻을 수 있음을 확인하였다.

Abstract

In this paper, we propose a fast full search block matching algorithm using the search region subsampling and the difference of adjacent pixels in current block. In the proposed algorithm, we calculate the lower bound of mean absolute difference (MAD) at each search point using the MAD value of neighbor search point and the difference of adjacent pixels in current block. After that, we perform block matching process only at the search points that need block matching process using the lower bound of MAD at each search point. To calculate the lower bound of MAD at each search point, we need the MAD value of neighbor search point. Therefore, the search points are subsampled at the factor of 4 and the MAD value at the subsampled search points are calculated by the block matching process. And then, the lower bound of MAD at the rest search points are calculated using the MAD value of the neighbor subsampled search point and the difference of adjacent pixels in current block. Finally, we discard the search points that have the lower bound of MAD value exceed the reference MAD which is the minimum MAD value of the MAD values at the subsampled search points and we perform the block matching process only at the search points that need block matching process. By doing so, we can reduce the computation complexity drastically while the motion compensated error performance is kept the same as that of full search block matching algorithm (FSBMA). The experimental results show that the proposed method has a much lower computational complexity than that of FSBMA while the motion compensated error performance of the proposed method is kept same as that of FSBMA.

* 正會員, 慶北大學校 電子電氣工學部
(School of Electronic and Electrical Engineering, Kyungpook National University)

接受日字:1999年5月19日, 수정완료일:1999年10月21日

Ⅰ. 서 론

동영상 압축 (video compression)은 중요한 디지털 기술의 하나로서, 고선명 TV, 주문형 비디오, 영상 회의 등의 다양한 응용 분야에 사용된다. 특히, 최근에는 초 저속 (low bit rate) 동영상 전송을 위한 움직임 보상 동영상 압축 (motion compensated video compression)에 대하여 많은 연구가 진행되고 있다. 여기서 움직임 추정 및 보상은 연속된 프레임간의 시간적인 중복성을 제거함으로써 초 저속 전송을 위한 높은 압축율을 얻는데 핵심적인 역할을 담당하고 있으며, 동영상 부호화를 위한 표준인 H.261,[1] H.263,[2] MPEG[3],[4] 등에 사용되고 있다.

움직임 추정 및 보상 기법으로는 수행시간이 비교적 적게 소요되고, 하드웨어 구현이 용이한 블록 정합 알고리듬 (block matching algorithm; BMA)[6]이 많이 사용되고 있다.[5] 이 방법에서는 입력 영상을 임의의 작은 블록으로 나눈 뒤, 블록내의 모든 화소들은 같은 방향으로 평행 이동한다는 것을 가정하여 이전 프레임의 탐색영역에서 정합 척도가 최적인 블록을 찾는다. 이때, 정합 척도로는 평균 자승 오차와 비슷한 성능을 가지면서도 계산량이 적은 MAD가 많이 사용된다.

BMA를 이용하여 움직임을 추정하는 경우에 가장 좋은 성능을 얻을 수 있는 방법은 이전 프레임의 탐색영역의 모든 탐색점에 대하여 탐색을 행하는 전역 탐색 블록 정합 알고리듬이다. 이 방법에서는 모든 탐색점에 대하여 탐색을 행하여 정합 척도가 최적인 블록을 찾기 때문에 움직임 추정 오차 측면에서 최적인 움직임 벡터를 얻을 수 있지만 계산량이 많은 단점이 있다.

이와 같은 단점을 줄이기 위하여 움직임 벡터를 고속으로 추정할 수 있는 여러 가지 고속 움직임 추정 기법들이 제안되었다. 대표적인 고속 BMA 방법으로는 2차원 로그 탐색 (two dimensional logarithm search; 2-D LOG), 3단계 탐색 (three step search; TSS), 교차 탐색 (cross search) 알고리듬 등이 있다.[6]-[10],[13] 이 방법들 중 간단하면서도 성능이 좋은 TSS 알고리듬이 가장 많이 사용되고 있으며, 이를 보완한 움직임 추정 기법들이 연구되었다.[10],[13] 그러나, 이 방법들은 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색영역의 일부분에 대하여서만 탐색을 행하므로, 계산량은 줄일 수 있었지만 국부 최소 (local minimum)에 빠질 수 있는 단점을 가진다. 즉, 이 방법들은 움직임 추정의 정확성을 어느 정도 희생함으로써 계산량의 감소를 얻는다.

또한, Liu 등[11]은 전역 탐색을 수행하면서 움직임 추정을 고속으로 행하기 위한 방법으로 블록의 화소들을 수직, 수평방향으로 부표본화 (subsampling)한 뒤 이를 이용하여 블록 정합을 행하는 방법을 제안하였다. 그러나 이 방법에서도 블록의 화소의 일부분만이 블록 정합의 계산에 사용되므로 전역 탐색 블록 정합 알고리즘에 비하여 성능이 떨어지는 단점을 가진다. 그러므로 움직임 추정 오차 측면에서 최적인 FSBMA와 동일한 성능을 유지하면서도 계산량을 줄일 수 있는 방법이 필요하다.

본 논문에서는 탐색 영역에서의 탐색점 부표본화와 현재 블록 내의 화소들의 이웃 화소간의 차를 이용한 고속 전역 탐색 블록 정합 알고리듬을 제안하였다. 제안한 방법에서는 각 탐색점에서의 MAD 값의 최소 범위를 이웃 탐색점에서의 MAD와 현재 블록 내의 화소들의 이웃 화소간의 화소 값의 차를 이용하여 구한 뒤, 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행하였다. 이때 현재 탐색점에서의 MAD의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAD를 사용하여야한다. 그러므로 제안한 방법에서는 먼저, 이전 프레임에 설정된 탐색 영역의 탐색점들에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 부표본화 된 탐색점에 대하여 블록 정합을 행하여 MAD들을 구하고, 이때 얻어지는 MAD들 중 최소 MAD를 기준 MAD (reference MAD)로 정한다. 그리고, 탐색을 행하지 않은 탐색점에 대하여서는 각 탐색점에서 가질 수 있는 MAD의 최소 범위를 구한 뒤, 이를 이용하여 탐색점에서의 MAD가 기준 MAD보다 작을 가능성이 있는 탐색점, 즉 MAD의 최소 범위가 기준 MAD보다 작은 탐색점에 대하여서만 블록 정합을 행하였다. 이때, MAD의 최소 범위는 부표본화된 탐색점에서 블록 정합을 통하여 얻은 MAD들과 현재 블록 내의 화소들의 이웃 화소간의 화소 값의 차를 이용하여 구하였다. 그러므로, 제안한 방법에서는 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행함으로써 FSBMA와 동일한 움직임 추정 성능을 유지하면서도 움직임 벡터의 추정을 위한 계산량을 줄일 수 있었다

제안한 방법의 성능을 평가하기 위한 여러 가지 동영상에 대한 컴퓨터 모의 실험을 통하여, 제안한 방법

이 FSBMA와 동일한 성능을 유지하면서 많은 계산량의 감소를 얻을 수 있음을 확인하였다.

Ⅱ. 블록 정합 알고리듬을 이용한 움직임 추정

BMA는 현재 입력 프레임을 일정한 크기의 블록으로 나눈 후, 이전 프레임에서 설정된 탐색영역에서 정합 척도가 최적인 위치를 찾는 것이다. 이때 사용되는 정합 척도로는 MSD와 비슷한 성능을 유지하면서도 계산량이 적고 하드웨어 구현이 용이한 MAD가 널리 이용되고 있다. 이때, MAD는

$$MAD_{(k,l)}(x,y) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | I_t(k+i, l+j) - I_{t-1}(k+x+i, l+y+j) | \quad (1)$$

와 같다. 여기서, N은 블록의 수직 및 수평 방향의 크기, $I_t(i,j)$는 현재 프레임의 (i,j) 좌표에서 화소의 휘도 값, $I_{t-1}(i,j)$는 이전 프레임의 (i,j) 좌표에서 화소의 휘도 값, (k,l)은 블록의 위치 좌표, 그리고 (x,y)는 탐색영역에서의 탐색점의 위치를 나타낸다. 이를 이용하여 (k,l)번째 블록의 움직임 벡터는

$$v(k,l) = \arg \min_{(x,y)} MAD_{(k,l)}(x,y) \quad (2)$$

와 같이 구한다.

BMA를 이용하여 움직임 추정을 행하는 경우에 가장 좋은 성능을 얻을 수 있는 방법은 전역 탐색 블록 정합 알고리듬으로서, 이 방법은 정합 될 수 있는 모든 화소에 대해 MAD를 구하고 그 중 가장 작은 MAD 값을 갖는 탐색점 (x,y)의 값을 움직임 벡터로 결정하는 것이다. 이때 $N\times N$ 화소 크기의 블록의 움직임 벡터의 수직 및 수평 방향의 최대 변위가 w 인 경우에 탐색영역에서 정합을 행하여야 할 탐색점수는 $(2w+1)^2$다. 따라서 전역 탐색 블록 정합 알고리즘은 탐색영역 내에서 움직임 추정 오차 측면에서 최적인 움직임 벡터를 추정할 수 있지만, 탐색점 수가 너무 많으므로 계산량이 많은 단점이 있다.

이와 같은 단점을 줄이기 위하여 탐색점을 줄이는 방법과 블록의 화소들을 부표본화한 뒤 블록 정합을 행하는 방법 등이 연구되었다. 그러나 탐색영역에서 탐색점을 줄이는 방법의 경우에는 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 전역 탐색을 행하지 않고 탐색영역의 일부에 대해서만 탐색을 행하므로, 계산량은 크게 줄일 수 있었지만 국부 최소에 빠질 수 있는 단점을 가진다. 즉, 움직임 추정의 정확성을 어느 정도 희생함으로써 계산량의 감소를 얻기 때문에 움직임 추정 오차가 커지며 특히, 이 방법에서 사용된 가정이 잘 맞지 않는 경우에는 움직임 추정 오차가 매우 커지는 단점이 있다. 움직임 추정 오차의 큰 증가는 복원영상에서 블록화 현상 등의 현저한 화질 열화를 일으킬 수 있으며, 움직임 보상 오차의 전송에 필요한 비트율을 크게 증가 시킬수 있다. 그러므로 고품질 (high-quality) 동영상 부호화기에서는 FSBMA가 많이 사용되고 있으며,[15],[16] FSBMA와 동일한 성능을 유지하면서도 움직임 추정에 필요한 계산량을 줄이기 위한 방법이 연구되고 있다.[14] 또한, 블록 내의 화소들 중 부표본화 된 화소들만을 정합에 이용하는 방법의 경우에는 화소의 일부분만이 블록 정합의 계산에 사용되므로 전역 탐색 블록 정합 알고리즘에 비해 정확한 움직임 벡터를 추정할 수 없는 단점이 있다. 그러므로 추정 오차 측면에서 최적인 전역 탐색 블록 정합 알고리듬의 성능을 유지하면서 계산량을 줄이는 방법이 필요하다.

Ⅲ. 제안한 고속 전역 탐색 블록 정합 알고리듬

본 논문에서는 FSBMA와 동일한 움직임 추정 성능을 유지하면서도 움직임 벡터의 추정을 위한 계산량을 줄일 수 있는 고속 전역 탐색 블록 정합 알고리듬을 제안한다. 제안한 방법에서는 각 탐색점에서의 MAD 값의 최소 범위를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행하였다. 이때 각 탐색점에서의 MAD의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAD를 사용하여야한다. 그러므로 제안한 방법에서는 먼저, 이전 프레임에 설정된 탐색 영역의 탐색점들에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 부표본화 된 탐색점에 대하여 블록 정합을 행하여 MAD들을 구하여, 이들을 이용하여 탐색을 행하지 않은 각 탐색점에서의 MAD의 최소 범위를 구하였다. 즉, 제안한 방법에서는 각 탐색점의 MAD값의 최소 범위

를 이용하여 블록 정합을 행하여야 할 탐색점 수를 줄임으로써 FSBMA와 동일한 성능을 얻으면서도 고속으로 움직임을 추정하였다.

본 장에서는 먼저, 이웃 탐색점에서의 MAD와 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용하여 각 탐색점에서의 MAD의 최소 범위를 구하는 방법에 대하여 알아본 뒤, MAD의 최소 범위를 이용하여 FSBMA와 동일한 성능을 얻으면서도 고속으로 움직임을 추정할 수 있는 고속 전역 탐색 블록 정합 알고리듬에 대하여 알아본다.

1. MAD의 최소 및 최대 범위

본 논문에서는 현재 움직임 추정을 행하고자 하는 블록의 블록 내의 화소들의 이웃 화소간의 화소 값의 차와 이웃 탐색점에서의 MAD를 이용하여 현재 탐색점의 MAD의 최소 범위를 구한 뒤, 이를 이용하여 블록정합이 필요한 탐색점에 대하여서만 블록 정합을 행한다. 이때 각 탐색점의 MAD의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAD가 필요하다. 즉, 탐색 영역의 (x,y) 위치에서의 MAD를 구하였다면, 이를 이용하여 $(x+1,y)$, $(x,y+1)$ 및 $(x+1,y+1)$ 위치에서의 MAD의 최소 범위를 구할 수 있다. 그러므로 제안한 방법에서는 먼저, 이전 프레임에 설정된 탐색 영역에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 이들 탐색점에 대하여 블록 정합을 행하여 이들 탐색점에 대하여 MAD를 구한 뒤, 이들을 이용하여 나머지 탐색점에서의 MAD의 최소 범위를 구한다. 제안한 방법을 자세히 살펴보기 위하여 수평 및 수직 방향의 크기가 N인 현재 블록 C와 탐색 영역의 (x,y)위치에서의 후보 정합 블록 (candidate matching block) $\mathbf{M}(x,y)$의 화소들의 휘도 값을 $N\times N$ 행렬로 나타내면 각각

$$\mathbf{C}=\begin{bmatrix} c_{0,0} & \cdots & c_{0,N-1} \\ \vdots & & \vdots \\ c_{N-1,0} & \cdots & c_{N-1,N-1} \end{bmatrix} \tag{3}$$

$$\mathbf{M}(x,y)=\begin{bmatrix} m_{y,x} & \cdots & m_{y,x+N-1} \\ \vdots & & \vdots \\ m_{y+N-1,x} & \cdots & m_{y+N-1,x+N-1} \end{bmatrix} \tag{4}$$

와 같고, 수평 방향의 이웃 탐색점에서의 후보 정합 블록 $\mathbf{M}(x+1,y)$의 화소들의 휘도 값을 $N\times N$ 행렬로 나타내면

$$\mathbf{M}(x+1,y)=\begin{bmatrix} m_{y,x+1} & \cdots & m_{y,x+N} \\ \vdots & & \vdots \\ m_{y+N-1,x+1} & \cdots & m_{y+N-1,x+N} \end{bmatrix} \tag{5}$$

와 같다. 그리고, 탐색점 (x,y)에서의 오차 블록 $\mathbf{D}(x,y)=\mathbf{M}(x,y)-\mathbf{C}$ 는

$$\mathbf{D}(x,y)=\begin{bmatrix} m_{y,x}-c_{0,0} & \cdots & m_{y,x+N-1}-c_{0,N-1} \\ \vdots & & \vdots \\ m_{y+N-1,x}-c_{N-1,0} & \cdots & m_{y+N-1,x+N-1}-c_{N-1,N-1} \end{bmatrix} \tag{6}$$

와 같고, 이로부터 탐색점 (x,y)에서의 MAD는

$$\begin{aligned} MAD(x,y) &= \frac{1}{N^2}\|\mathbf{M}(x,y)-\mathbf{C}\| \\ &= \frac{1}{N^2}\|\mathbf{D}(x,y)\| \end{aligned} \tag{7}$$

와 같이 구할 수 있다. 여기서 $\|\cdot\|$는 $N\times N$ 행렬의 합의 놈 (sum norm)으로서, 임의의 $N\times N$ 행렬 $\mathbf{A}$의 합의 놈은

$$\|\mathbf{A}\|=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|a_{ij}| \tag{8}$$

와 같다.[14] 여기서, a_{ij}는 행렬 $\mathbf{A}$의 (i,j)번째 원소를 나타낸다.

또한, $(x+1,y)$, $(x,y+1)$ 및 $(x+1,y+1)$ 위치의 탐색점에서의 MAD는 이웃 탐색점 (x,y)에서의 오차 블록 $\mathbf{D}(x,y)$ 및 블록 내의 화소들의 이웃 화소간의 화소 값의 차를 이용하여 구할 수 있다. 이를 탐색점 (x,y)의 수평 방향의 이웃 탐색점인 $(x+1,y)$ 위치의 탐색점을 예를 들어 살펴보면, 먼저 탐색점 $(x+1,y)$에서의 오차 블록 $\mathbf{D}(x+1,y)$는 식 (5)에서 정의된 $(x+1,y)$ 위치에서의 후보 정합 블록 $\mathbf{M}(x+1,y)$와 현재 블록 $\mathbf{C}$의 차로서

$$\mathbf{D}(x+1,y)=\mathbf{M}(x+1,y)-\mathbf{C}$$

$$=\begin{bmatrix} m_{y,x+1}-c_{0,0} & \cdots & m_{y,x+N-1}-c_{0,N-2} & m_{y,x+N}-c_{0,N-1} \\ \vdots & & \vdots & \vdots \\ m_{y+N-1,x+1}-c_{N-1,0} & \cdots & m_{y+N-1,x+N-1}-c_{N-1,N-2} & m_{y+N-1,x+N}-c_{N-1,N-1} \end{bmatrix} \tag{9}$$

와 같다. 이를 이웃 탐색점 (x, y)에서의 오차 블록 $\mathbf{D}(x, y)$를 이용하여 표현하기 위하여 현재 블록 내의 화소들의 이웃 화소간의 수평 방향으로의 화소값의 차를

$$\mathbf{C}_{DH} = \begin{bmatrix} c_{0,0} - c_{0,N-1} & c_{0,1} - c_{0,0} & \cdots & c_{0,N-1} - c_{0,N-2} \\ \vdots & \vdots & & \vdots \\ c_{N-1,0} - c_{N-1,N-1} & c_{N-1,1} - c_{N-1,0} & \cdots & c_{N-1,N-1} - c_{N-1,N-2} \end{bmatrix} \tag{10}$$

와 같이 정의하면, $\mathbf{D}(x, y)$와 $\mathbf{C}_{DH}$의 합은

$$\mathbf{D}(x, y) + \mathbf{C}_{DH} = \begin{bmatrix} m_{y,x} - c_{0,N-1} & m_{y,x+1} - c_{0,0} & \cdots & m_{y,x+N-1} - c_{0,N-2} \\ \vdots & \vdots & & \vdots \\ m_{y+N-1,x} - c_{N-1,N-1} & m_{y+N-1,x+1} - c_{N-1,0} & \cdots & m_{y+N-1,x+N-1} - c_{N-1,N-2} \end{bmatrix} \tag{11}$$

와 같이 나타낼 수 있다. 이때 이 식을 식 (9)과 비교해 보면, 식 (9)의 마지막 열을 제외한 나머지 부분과 식 (9)의 첫 번째 열을 제외한 나머지 부분이 동일한 원소로 구성되어 있음을 알 수 있다. 그러므로 $\mathbf{D}(x, y)$의 첫 번째 열 $[\, m_{y,x} - c_{0,0} \;\; \cdots \;\; m_{y+N-1,x} - c_{N-1,0} \,]^T$를 $[\, m_{y,x+N} - c_{0,0} \;\; \cdots \;\; m_{y+N-1,x+N} - c_{N-1,0} \,]^T$ 로 대치시킨 새로운 오차 블록 $\hat{\mathbf{D}}_H(x, y)$를

$$\hat{\mathbf{D}}_H(x, y) = \begin{bmatrix} m_{y,x+N} - c_{0,0} & m_{y,x+1} - c_{0,1} & \cdots & m_{y,x+N-1} - c_{0,N-1} \\ \vdots & \vdots & & \vdots \\ m_{y+N-1,x+N} - c_{N-1,0} & m_{y+N-1,x+1} - c_{N-1,1} & \cdots & m_{y+N-1,x+N-1} - c_{N-1,N-1} \end{bmatrix} \tag{12}$$

와 같이 구한 뒤, 이를 $\mathbf{C}_{DH}$와 더하면

$$\hat{\mathbf{D}}_H(x, y) + \mathbf{C}_{DH} = \begin{bmatrix} m_{y,x+N} - c_{0,N-1} & m_{y,x+1} - c_{0,0} & \cdots & m_{y,x+N-1} - c_{0,N-2} \\ \vdots & \vdots & & \vdots \\ m_{y+N-1,x+N} - c_{N-1,N-1} & m_{y+N-1,x+1} - c_{N-1,0} & \cdots & m_{y+N-1,x+N-1} - c_{N-1,N-2} \end{bmatrix} \tag{13}$$

와 같다. 이 식을 식 (9)와 비교해 보면 각 행이 오른쪽으로 한 열씩 이동한 형태로서 두 수식은 동일한 원소들로 구성되어있음을 알 수 있다. 그러므로 탐색점 $(x+1, y)$에서의 MAD는 식 (9) 대신 식 (13)을 이용하여 구할 수 있으며, 이를 수식으로 나타내면

$$MAD(x+1, y) = \frac{1}{N^2} \| \mathbf{M}(x+1, y) - \mathbf{C} \| = \frac{1}{N^2} \| \hat{\mathbf{D}}_H(x, y) + \mathbf{C}_{DH} \| \tag{14}$$

와 같다. 이 식으로부터 $MAD(x+1, y)$는 미리 구해진 이웃 탐색점 (x, y)에서의 오차 블록으로부터 구할 수 있음을 알 수 있다. 그러나 이 식을 이용하여 $MAD(x+1, y)$를 구한다면 $\mathbf{C}_{DH}$를 구하는 과정과 $\hat{\mathbf{D}}_H(x, y)$의 첫 번째 열을 대치시키는 과정으로 인하여 계산량의 이득을 얻을 수 없다. 그러므로 본 논문에서는 삼각 부등식 (triangular inequality)을 이용하여 $MAD(x+1, y)$의 최소 범위를 구한 뒤 이를 이용하여 움직임 추정을 위한 계산량을 줄였다.

임의의 $N \times N$ 행렬 $\mathbf{A}_1$ 및 $\mathbf{A}_2$에 삼각 부등식을 적용하면

$$| \| \mathbf{A}_1 \| - \| \mathbf{A}_2 \| | \le \| \mathbf{A}_1 + \mathbf{A}_2 \| \le \| \mathbf{A}_1 \| + \| \mathbf{A}_2 \| \tag{15}$$

와 같고, 이를 식 (14)에 적용하면

$$| \| \hat{\mathbf{D}}_H(x, y) \| - \| \mathbf{C}_{DH} \| | \le \| \hat{\mathbf{D}}_H(x, y) + \mathbf{C}_{DH} \| \le \| \hat{\mathbf{D}}_H(x, y) \| + \| \mathbf{C}_{DH} \| \tag{16}$$

와 같다. 또한 식 (14)로부터 $\| \hat{\mathbf{D}}_H(x, y) + \mathbf{C}_{DH} \| = \| \mathbf{M}(x+1, y) - \mathbf{C} \|$이므로 이를 식 (16)에 대입한 뒤, 식의 양변을 N^2으로 나누면 식 (16)는

$$\left| \frac{1}{N^2} \| \hat{\mathbf{D}}_H(x, y) \| - \frac{1}{N^2} \| \mathbf{C}_{DH} \| \right| \le MAD(x+1, y) \le \frac{1}{N^2} \| \hat{\mathbf{D}}_H(x, y) \| + \frac{1}{N^2} \| \mathbf{C}_{DH} \| \tag{17}$$

와 같다. 이 식의 좌변은 탐색점 $(x+1, y)$에서의 MAD가 가질 수 있는 최소 범위를 나타내고, 우변은 탐색점 $(x+1, y)$에서의 MAD가 가질 수 있는 최대 범위를 나타낸다. 즉, 이 식으로부터 탐색점 $(x+1, y)$에서의 MAD의 최소 및 최대 범위을 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차 $\mathbf{C}_{DH}$와 탐색점 (x, y)에서의 오차 블록 $\mathbf{D}(x, y)$의 첫 번째 열을 대치한 오차 블록 $\hat{\mathbf{D}}_H(x, y)$ 로부터 구할 수 있음을 알 수 있다. 이때 식 (17)에서 $\| \mathbf{C}_{DH} \|$는 전체 탐색 영역에 대하여 한번만 계산하면 되고, $\| \hat{\mathbf{D}}_H(x, y) \|$는 N개의 열 중에서 첫 번째 열을 대치시키는 과정이 필요하므로, 블록 정합을 행하는 경우에 비하여 매우 적은 계산량으로 MAD의 최소 범위를 구할 수 있다. 그러므로 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 정합을 행한다면 계산량을 크게 줄일 수 있다. 그리

고 이것은 수직 및 대각선 방향의 탐색점에 대하여서도 동일하게 적용할 수 있다.

2. 탐색점 부표본화 및 MAD의 최소 범위를 이용한 탐색점 줄임

각 탐색점에서의 MAD의 최소 범위는 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차와 이웃 탐색점에서의 MAD로부터 구할 수 있다. 그러나, 현재 탐색점에서의 MAD의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAD를 이용하여야한다. 그러므로 제안한 방법에서는 먼저, 탐색 영역의 탐색점을 4:1로 부표본화하여 이에 대한 MAD를 블록 정합을 통하여 구한다. 또한 나머지 탐색점에 대하여서는 앞에서 구한 MAD들과 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용하여 현재 탐색점에서의 MAD의 최소 범위를 구한 뒤, 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행한다.

제안 방법을 자세히 살펴보면 먼저, 이전 프레임에 설정된 탐색 영역에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 이들 탐색점에 대하여 블록 정합을 행하여 MAD를 구하고, 기준 평균 절대치 오차 $MAD_{Ref.}$를

$$MAD_{Ref.} = \min MAD(x,y) \tag{18}$$

where, (x,y): The subsampled search points.

와 같이 부표본화 된 탐색점의 MAD들 중 최소값을 가지는 MAD로 구한다. 그리고, 부표본화 된 탐색점 (x,y)의 수평, 수직, 및 대각 방향의 이웃 탐색점인 $(x+1,y)$, $(x,y+1)$ 및 $(x+1,y+1)$ 위치의 탐색점에 대하여서는 탐색점 (x,y)의 MAD인 $MAD(x,y)$를 이용하여 각각에 대응되는 첫 번째 열 및 첫 번째 행을 대치한 오차 블록의 놈인 $\| \hat{\mathbf{D}}_k(x,y) \|$를 구한 뒤,

$$IF\left(MAD_{Ref.} < \left| \frac{1}{N^2} \| \hat{\mathbf{D}}_k(x,y) \| - \frac{1}{N^2} \| \mathbf{C}_{Dk} \| \right| \right)$$

No block matching (19)

ELSE {

Block matching

$IF(MAD(i,j) < MAD_{Ref.})$

$MAD_{Ref.} = MAD(i,j)$

}

where, (i,j): The search points excluded by subsampling.

$k = H, V, D$

와 같이 MAD의 최소 범위가 기준 MAD인 $MAD_{Ref.}$보다 작은 경우에 대하여서만 블록 정합을 행한다. 즉, 탐색점에서의 MAD의 최소 범위가 $MAD_{Ref.}$보다 크다면 블록 정합을 행하여 얻어지는 실제 MAD는 MAD의 최소 범위보다 작아질 수 없으므로 이 탐색점에서는 블록 정합을 수행할 필요가 없고, MAD의 최소 범위가 $MAD_{Ref.}$보다 작다면 블록 정합을 통하여 얻어지는 실제 MAD가 $MAD_{Ref.}$보다 작을 가능성이 있으므로 블록 정합을 행하여 이 탐색점에서의 실제 MAD인 $MAD(i,j)$를 구한 뒤 $MAD_{Ref.}$와 비교한다. 이때 $MAD(i,j)$가 $MAD_{Ref.}$보다 작으면 $MAD_{Ref.}$는 $MAD(x,y)$로 갱신되고, 이것의 최종값을 지닌 탐색점이 최종 움직임 벡터로 결정된다. 이와 같은 방법으로 제안한 방법에서는 현재 블록의 움직임 탐색 영역의 모든 탐색점에 대하여 탐색을 행하지 않고도, FSBMA와 마찬가지로 움직임 추정 오차 측면에서 최적인 움직임 벡터를 찾을 수 있다.

3. 탐색점 줄임에 의한 계산량 감소

FSBMA는 이전 프레임에 설정된 모든 탐색점에 대하여 MAD를 계산하며, 식 (1)로부터 MAD 계산에는 $2N^2$번의 덧셈과 뺄셈의 계산이 필요함을 알 수 있다. 그러므로 현재 프레임의 블록의 전체 개수가 T이고, 움직임 탐색 영역이 상하, 좌우의 최대 변위가 w이면 한 프레임에 대한 전역 탐색 블록 정합 알고리즘의 총 계산량은

$$C_{FSBMA} = 2T(2w+1)^2N^2 \tag{20}$$

와 같다.

반면, 제안한 방법의 계산량은 4:1로 탐색점 부표본화된 탐색점에 대하여 블록 정합을 행하여 MAD를 구하고, 이들로부터 기준 MAD를 구하는 과정인 식 (18)의 계산량과 각 탐색점의 MAD의 최소 범위를 구한 뒤, 블록 정합이 필요한 탐색점에 대하여 블록 정합을 행하는 식 (19)의 계산량의 합으로 볼 수 있다. 먼저 식 (18)의 계산량을 살펴보면, 전체 탐색점의 1/4에 해당하는 탐색점에 대하여 블록 정합을 행하여야하므로 FSBMA의 계산량의 1/4에 해당하는 $T(2w+1)^2N^2/2$의 계산량이 필요하다. 또한, 식 (19)의 계산에서는 전체 탐색 영역의 3/4에 해당하는 탐색점에 대하여 MAD의 최소 범위를 구하기 위한 계산량과 MAD의 최소 범위

가 $MAD_{Ref.}$보다 작은 탐색점에서의 블록 정합을 위한 계산량이 필요하다. 여기서, MAD의 최소 범위를 구하기 위해서는 수평, 수직 및 대각선 방향에 대응되는 $\| \hat{\mathbf{D}}_k \|$와 $\| \mathbf{C}_{Dk} \|$를 구하여야 한다. 이때, $\| \hat{\mathbf{D}}_k \|$를 구하기 위한 계산량을 살펴보면, 수평 방향의 이웃 탐색점의 MAD를 이용하여 구하는 경우, 즉 탐색점 (x, y)에서의 MAD를 이용하여 $(x+1, y)$에서의 MAD를 구하는 경우에는 식 (12)에서와 같이 첫 번째 행을 대치시키는 과정이 필요하며, 이를 위해서는 $2(2N-1)$번의 덧셈과 뺄셈의 계산이 필요하다. 또한, 수직 방향의 경우에 대하여서도 같은 방법으로 계산하면 $2(2N-1)$번의 계산이 필요하며, 대각 방향의 경우에는 수평 및 수직 방향에 대하여 계산된 결과를 이용하면 4번의 계산만이 필요하다. 그리고, $\| \mathbf{C}_{Dk} \|$의 계산을 위해서는 한 블록 전체에 대하여 수평, 수직, 및 대각 방향으로 각각 $2N^2$번의 덧셈과 뺄셈의 계산이 필요하다. 그러므로 제안한 방법의 계산량은

$$C_{proposed} = TN^2(2w+1)^2/2 + 2PN^2 + 2TN(2w+1)^2 + 6TN^2 \quad (21)$$

와 같다. 여기서 P는 식 (18)에서 MAD의 최소 범위가 $MAD_{Ref.}$보다 작은 탐색 점의 총 개수이다. 이 식으로부터 제안한 방법의 계산량은 MAD의 최소 범위가 기준 평균 절대 오차 $MAD_{Ref.}$보다 큰 탐색점 수가 많을수록 전역 탐색 블록 정합 알고리듬의 계산량에 비하여 많이 감소됨을 알 수 있다.

Ⅳ. 실험 결과 및 고찰

본 논문에서 제안한 방법의 성능을 평가하기 위하여 컴퓨터 모의실험을 행하였다. 본 실험에서는 720×480의 공간해상도를 가지는 FOOTBALL, FLOWER GARDEN 및 MOBILE 영상 각 40프레임을 사용하였으며, 블록 정합에 사용된 블록의 크기는 16×16, 탐색 영역의 크기는 수직과 수평방향으로 -16~15로 MPEG의 권고안과 동일하게 하였다. 본 실험에서 사용한 FOOTBALL 영상은 운동장을 배경으로 가지는 전체적으로 복잡하고, 물체 및 카메라의 움직임이 매우 빠른 영상이며, FLOWER GARDEN 영상은 꽃밭과 같은 복잡한 부분이 많이 존재하고, 정지된 물체에 카메라가 빠르게 움직이는 영상이다. 그리고 MOBILE 영상은 고정된 배경에 기차가 움직이는 영상으로서 움직임이 거의 없는 배경과 물체의 움직임이 빠른 영역이 혼재 하는 영상이다. 그리고 정합 척도로는 계산량이 적은 MAD를 사용하였다. 또한 본 실험에서는 전역 탐색 알고리즘과 제안한 방법에 대한 결과를

$$PSNR = 10\log\frac{255^2}{\sigma_e^2}[dB] \quad (22)$$

를 이용하여 평가하였다. 여기서, σ_e^2 은 원 영상과 움직임 보상된 영상의 평균 자승 오차이다.

표 1. PSNR [dB] 비교
Table 1. The comparison of PSNR [dB].

Sequences	MOBILE	FLOWER GARDEN	FOOT BALL
FSBMA	32.66	26.67	25.98
Proposed method	32.66	26.67	25.98

MOBILE, FLOWER GARDEN 및 FOOTBALL 영상에 대한 40 프레임 평균 PSNR을 표 1에 나타내었다. 이 표로부터 제안한 방법과 FSBMA의 평균 PSNR이 동일함을 알 수 있다. 이는 제안한 방법이 식 (20)에서와 같이 블록 정합이 필요한 부분에 대하여서만 블록 정합을 행하지만, 블록 정합의 수행 여부를 각 탐색점에서의 MAD의 최소값을 이용하여 결정함으로써, 기준 MAD보다 작은 MAD를 가질 수 없는 탐색점에 대하여서만 블록 정합을 행하지 않으므로 FSBMA와 동일한 성능을 얻을 수 있기 때문이다. 또한, 표 1에서 움직임과 영상의 복잡한 부분이 FLOWER GARDEN 및 FOOTBALL 영상보다 상대적으로 적은 MOBILE 영상이 PSNR이 가장 큼을 알 수 있다. 또한 영상의 복잡한 부분은 비슷하지만 FLOWER GARDEN 영상보다 빠른 움직임을 가지는 FOOTBALL영상의 PSNR이 가장 작음을 볼 수 있다.

FSBMA와 제안한 방법의 블록 정합을 행한 탐색점 수의 40 프레임 평균치를 표 2에 나타내었다. 이 표에 나타난 수치는 FSBMA의 탐색점 수를 100%로 하여 제안한 방법의 탐색점 수를 FSBMA의 탐색점 수에 대하여 상대적으로 나타내었다. 이 표로부터 제안한 방법

의 경우 블록 정합이 필요한 탐색점 수는 FSBMA의 약 38%~43%로 FSBMA에 비하여 훨씬 적음을 알 수 있다. 이를 자세히 살펴보기 위하여 각 실험 영상 40 프레임에 대한 제안한 방법의 FSBMA에 대한 탐색점 감소율을 그림 1에 나타내었다. 이 그림을 살펴보면 FOOTBALL 영상의 경우에는 탐색점 감소율이 프레임에 따라 20%의 변동폭을 가짐을 알 수 있다. FOOTBALL 영상의 앞부분에서는 풋볼 선수들의 빠른 움직임으로 인하여, 이전 프레임에는 없던 선수들이 나타나기 때문에 이전 프레임에는 없는 새로운 영역이 많이 발생하게된다. 그러므로 이 부분에 대하여서는 탐색점의 감소가 다소 적음을 알 수 있고, 프레임이 진행되면서 많은 탐색점의 감소를 얻으면서 탐색점 감소율이 안정화됨을 이 그림으로부터 확인할 수 있다. 또한, FLOWER GARDEN 영상의 경우에는 프레임에 따라 약간의 탐색점 감소율의 변화가 있음을 알 수 있고, MOBILE 영상의 경우에는 프레임에 따른 탐색점 감소율의 변화가 거의 없음을 확인할 수 있다. 즉, 식 (18)을 살펴보면 제안한 방법의 탐색점 수는 MAD의 최소 범위가 기준 MAD보다 큰 탐색점의 수에 따라 달라질 수 있음을 알 수 있지만, 그림 1로부터 탐색점 수의 변화량이 평균 값을 중심으로 큰 변화가 없음을 확인할 수 있다. 이상의 결과로부터 제안한 방법의 경우에는 움직임 추정 오차 측면에서 최적인 FSBMA와 동일한 움직임 추정 성능을 유지하면서도 평균 50% 이상의 탐색점 감소를 얻을 수 있음을 확인할 수 있었다.

표 2. FSBMA에 대한 제안한 방법의 평균 탐색점 수 비교 [%]

Table 2. The comparison of average search point of propsed method with FSBMA [%].

Sequences	MOBILE	FLOWER GARDEN	FOOT BALL
FSBMA	100.0	100.0	100.0
Proposed method	42.1	38.2	43.0
Reduction ratio	57.9	61.8	57.0

또한, 제안한 방법의 계산량은 식 (21)에서 볼 수 있는바와 같이 블록 정합을 위한 계산량과 블록 정합 여부를 결정하기 위한 부가적인 계산량으로 이루어진다. 이와 같은 부가적인 계산량까지 포함한 제안한 방법의 FSBMA에 대한 계산량의 감소율을 그림 2에 나타내었다. 이 그림에서 나타난 제안한 방법의 FSBMA에 대한 계산량의 감소 정도 R은

$$R = (1 - \frac{C_{proposed}}{C_{FSBMA}}) \times 100 \quad \% \qquad (22)$$

와 같이 표현 될 수 있다. 이 그림을 살펴보면 제안한 방법의 부가적인 계산량은 모든 프레임에 대하여 동일한 양이므로, 그림 1에 나타낸 제안한 방법의 탐색점 감소율과 동일한 모양을 가짐을 알 수 있다. 또한 이 그림으로부터 제안한 방법이 영상에 따라 다소의 차이는 있지만 FSBMA에 비하여 약 51%에서 56% 정도의 계산량 감소를 얻을 수 있음을 알 수 있다.

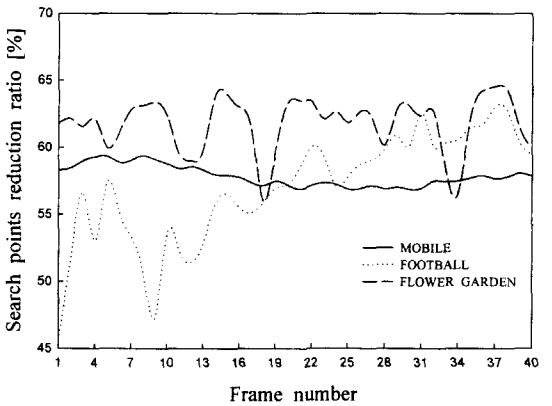

그림 1. 제안한 방법의 탐색점 감소율

Fig. 1. The search point reduction ratio of proposed method.

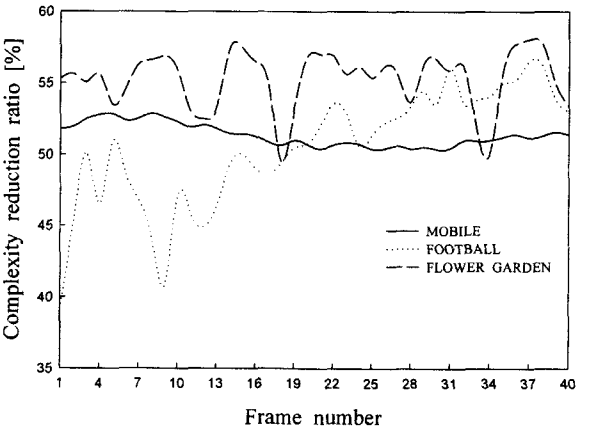

그림 2. 제안한 방법의 계산량 감소율

Fig. 2. The computational complexity reduction ratio of proposed method.

이상의 모의 실험 결과로부터 제안한 방법은 FSBMA와 동일한 움직임 추정 오차를 유지하면서도 블록 정합이 필요한 탐색점 수는 57%~62%, 전체적인 계산량은 51%~56% 정도 줄일 수 있음을 확인할 수 있었다.

V. 결 론

본 논문에서는 탐색 영역에서의 탐색점 부표본화와 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용한 고속 전역 탐색 블록 정합 알고리듬을 제안하였다. 제안한 방법에서는 각 탐색점에서의 MAD 값의 최소 범위를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행하였다. 제안한 방법에서는 먼저, 이전 프레임에 설정된 탐색 영역의 탐색점들에 대하여 4:1로 탐색점 부표본화를 행한 뒤, 부표본화된 탐색점에 대하여 블록 정합을 행하여 MAD들을 구하여, 이들을 이용하여 탐색을 행하지 않은 각 탐색점에서의 MAD의 최소 범위를 구하였다. 이렇게 구한 각 탐색점의 MAD값의 최소 범위를 이용하여 블록 정합을 행하여야 할 탐색점 수를 줄임으로써 FSBMA와 동일한 성능을 얻으면서도 고속으로 움직임을 추정하였다.

제안한 방법의 성능을 평가하기 위하여 여러 가지의 동영상에 대하여 컴퓨터 모의 실험을 수행하였다. 모의 실험 결과로부터 제안한 방법은 FSBMA와 동일한 움직임 추정 오차를 유지하면서도 블록 정합이 필요한 탐색점 수는 57%~62%, 전체적인 계산량은 51%~56% 정도 줄일 수 있음을 확인할 수 있었다.

참 고 문 헌

[1] *ITU-T Recommendation H.261*, "Video codec for audiovisual services at p×64 kbits/s."

[2] *ITU-T Recommendation H.263*, "Video coding for low bit rate communication."

[3] *ISO/IEC 11172-2*, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s: Video."

[4] *ISO/IEC 13818-2*, "Information technology - Generic Coding of Moving Pictures and Associated Audio Information: Video."

[5] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI design for the motion compensation block-matching algorithm," *IEEE Trans. Circuit and Systems*, vol. 36, no. 10, pp. 1309-1316, 1989.

[6] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799-1801, Dec., 1981.

[7] T. Koga, K. Iinuma, A Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," Proc. *Nat. Telecommun. Conf.*, pp. G5.3.1-G5.3.5, Nov./Dec. 1981.

[8] R. Strinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COM-33, pp. 1011-1014, Sept. 1985.

[9] M. Ganbari, "The cross search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. COM-38, no. 7, pp. 950-953, July 1990.

[10] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 4, no. 4, pp. 438-442, Aug. 1994.

[11] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 3, no. 2, pp. 148-157, Apr. 1993.

[12] Y. L. Chan and W. C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 6, no. 1, pp. 113-118, Feb. 1996.

[13] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 7, no. 2, pp. 429-433, Apr. 1997.

[14] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, no. 1. pp. 105-107, Jan. 1995.

[15] H. Fujiwara, M. L. Liou, M.-T. Sun, K.-M. Yang, M. Maruyama, K. Shomura, and K. Ohyama, "An all ASIC implementation of a low bit-rate video codec," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 2, no. 2, pp. 123-134, June 1992.

[16] P. A. Ruetz, P. Tong, D. A. Luthi, and P. H. Ang, "A high-performance full-motion video compression chip set," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 2, no. 2, pp. 111-122, June, 1992.

저 자 소 개

鄭 元 植(正會員) 第 34卷 S編 第 9號 參照

李 法 基(正會員) 第 35卷 S編 第 5號 參照

李 京 桓(正會員) 第 35卷 S編 第 5號 參照

崔 正 鉉(正會員) 第 35卷 S編 第 5號 參照

金 慶 圭(正會員) 第 34卷 S編 第 9號 參照

金 德 奎(正會員) 第 35卷 S編 第 5號 參照

李 健 一(正會員) 第 34卷 S編 第 9號 參照