인접블록의 움직임벡터를 이용한 고속 움직임추정 방식

정회원 소 현 호*, 종신회원 김 진 상**, 정회원 조 원 경**, 김 영 수**, 서 덕 영**

Fast Motion Estimation Algorithm Using Motion Vectors of Neighboring Blocks

Hyeon-Ho So* ***Reguler Member***, Jinsang Kim** ***Lifelong Member***,
Won-Kyung Cho**, Young-Soo Kim**, Doug Young Suh** ***Reguler Members***

요 약

본 논문에서는 곱셈을 수행할 때 발생되는 스위칭 율을 줄이는 방식의 저전력 부스 곱셈기를 제안한다. radix-4 부스 알고리즘 (radix-4 Booth algorithm)은 입력에서 연속되는 3비트가 0이나 1의 같은 값을 가지게 되면, 부스 인코딩 결과로서 0을 발생시키는 특성을 가지고 있다. 따라서 곱셈기의 두 입력 중 더 작은 활성영역을 갖는 입력을 승수로 사용할 때 부분 곱셈결과가 0이 될 확률이 높다. 제안된 곱셈기는 곱셈식을 본래의 곱셈 입력 비트보다 더 작은 비트를 갖는 여러 개의 곱셈식으로 분할한 후, 각각의 곱셈들을 독립적으로 계산하여 각각의 곱셈의 결과를 더하여 최종적인 결과를 얻는다. 따라서 곱셈의 두 입력간의 교환율은 기존의 곱셈기보다 더 높아지게 된다. 이는 제안된 곱셈기의 부스 인코딩 결과가 0이 되는 확률이 기존의 곱셈기보다 더 높은 저전력 곱셈기를 구현할 수 있음을 의미한다. 제안된 곱셈기는 기존의 부스 곱셈기보다 최대 20% 정도의 소모전력이 감소됨을 확인하였다.

Key Words : Booth multiplier, dynamic range, low-power, switching activity

ABSTRACT

In this paper, we propose a low-power Booth multiplication which reduces the switching activities of partial products during multiplication process. Radix-4 Booth algorithm has a characteristic that produces the Booth encoded products with zero when input data have sequentially equal values (0 or 1). Therefore, partial products have higher chances of being zero when an input with a smaller effective dynamic range of two multiplication inputs is used as a multiplier data instead of a multiplicand. The proposed multiplier divides a multiplication expression into several multiplication expressions with smaller bits than those of an original input data, and each multiplication is computed independently for the Booth encoding. Finally, the results of each multiplication are added. This means that the proposed multiplier has a higher chance to have zero encoded products so that we can implement a low power multiplier with the smaller switching activity. Implementation results show the proposed multiplier can save maximally about 20% power dissipation than a previous Booth multiplier.

Ⅰ. 서 론

정보통신의 발달로 대용량의 동영상 정보의 전송과 저장의 요구가 많아지고 있다. 이러한 대용량 동영상의 실시간 처리를 위해 동영상 압축이 필수적이다. 대부분의 동영상 정보는 2차원 프레임상의 공

※본 연구는 한국과학재단 목적기초연구 R01-2003-000-10149-0 지원으로 수행되었습니다.
* 푸른기술 연구소 (sky4402@Puloon.co.kr), ** 경희대학교 전자공학과 ({jskim27, chowk, yskim, suh}@khu.ac.kr)
논문번호 : KICS040178-0510, 접수일자 : 2004년 5월 10일

간적 중복성과 연속된 프레임 사이의 시간적인 중복성을 가지고 있다. 동영상 압축방식에서 시간적으로 인접한 영상의 시간적 중복성을 이용하여 정보의 전송량을 줄이는 고속의 움직임추정방식에 대한 연구는 상당히 중요하다. 예를 들어 CCITT H.261에서 필요로 하는 총계산량은 1,193 MOPS (Mega Operation Per Second) 이며, 그 중에서 움직임추정 방식이 차지하는 비중은 608 MOPS로 전체의 약 51.0%이고, 움직임추정과정의 전력소모값은 전체 압축과정의 약 50%를 차지한다[1],[2].

움직임추정의 가장 일반적인 방법은 하드웨어 구현이 용이한 블록정합 방식 (BMA: block matching algorithm) 이다[3],[4][11]. 블록정합 방식에서는 전역탐색 블록정합 방식(FSBMA: full search block matching algorithm)이 주로 사용된다. FSBMA은 현재 프레임의 기준블록과 이전 프레임의 탐색영역 내의 모든 블록간의 정합오류를 계산하여 가장 오류가 적은 블록에 해당하는 움직임벡터를 찾는 방법으로 많은 계산량이 요구된다. 이러한 문제점을 해결하기 위하여 New Three Step Search (N4SS)[5], Four Step Search (4SS)[6], Diamond Search (DS)[7], Cellular Search (CS)[8-9], Cross Diamond Search (CDS)[10] 방식 등이 제안되었다.

본 논문에서 제안된 고속 움직임추정 방식은 가장 고속의 움직임추정 방식으로 평가되는 CDS 방식을 개선한 것이다. 제안된 방식은 현재 블록의 움직임벡터의 방향성을 미리 예측한 후, 예측된 방향으로 움직임추정을 함으로써 기존의 방식보다 연산 횟수를 줄인다. 실험결과 제안된 방식은 기존의 고속 움직임추정방식보다 성능의 저하가 없고 계산량이 상당히 감소됨을 확인하였다.

본 논문의 구성은 다음과 같다. II절에서는 기존의 움직임추정 방식을 간단하게 기술하고, III 절에서는 제안된 고속 움직임추정 방식을 기술한다. IV절에서는 실험결과 및 평가에 대하여 논의하며 V절에서 결론을 맺는다.

Ⅱ. 기존의 움직임추정 방식

전역탐색 블록정합 방식은 (1)식과 (2)식과 같이 현재 프레임을 N x N 블록들로 나누고, 각 블록에 대해서 이전 프레임의 정해진 탐색영역의 모든 지점에서 블록정합오류값을 계산하여 이 값이 최소가 되는 위치를 찾아 이를 움직임벡터로 결정하는 방법이다. SAD (sum of absolute difference) 는 연산이 간단하여 블록정합오류 측정값으로 많이 이용되고 있다.

$$SAD(u,v) = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} |I_t(i,j) - I_{t-1}(i+u, j+v)| \quad (1)$$

$$(u,v) = \min_{(u,v)}\{SAD(u,v)\} \quad (2)$$

전역 탐색 블록정합 방식은 블록내의 모든 픽셀에 대하여 SAD 값을 계산하기 때문에 성능이 우수하지만 계산량이 많은 단점이 있다. 이런 단점을 개선하기 위해 블록정합에 기반을 둔 N4SS, 4SS, DS, CS, CDS 등의 많은 고속 움직임추정 방식들이 제안되었다. 이 절에서는 본 연구는 CDS 방식[10]을 개선한 방법이다. CDS 방식은 그림 1 (a)의 탐색패턴을 이용하며 탐색의 예는 그림 1 (b)와 (c)와 같다. CDS 방식은 그림 1 (a), (b)와 같이 중심점을 중심으로 작은 움직임을 갖는 위치에 대해서는 DS 방식보다 좋은 성능을 보이지만, 움직임이 큰 영상에서는 성능의 저하를 보인다.

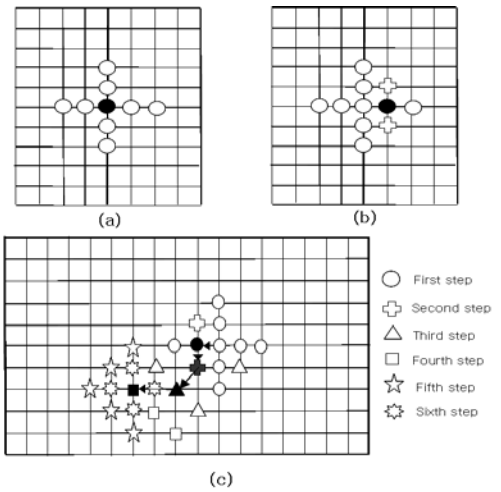


그림 1. CDS 방식의 예. (a) 움직임 좌표가 (0,0) 일 경우, 첫 번째 단계에서 멈춤. (b) 움직임 좌표가 (1,0) 일 경우, 두 번째 단계에서 멈춤. (c) 움직임 좌표가 (-4, 2) 일 경우의 탐색과정

Ⅲ. 인접블록의 움직임벡터를 이용한 고속 움직임추정 방식

제안된 움직임추정 방식은 기존의 CDS 움직임추정 방식에 기반을 둔 것으로써, CDS 움직임추정 방식의 탐색패턴보다 더 작은 탐색점을 가진 탐색패턴을 이용하고, 인접블록간의 움직임벡터의 상관성을

이용하여 움직임추정의 고속화가 가능하게 한다.

대부분의 동영상은 윤곽부분을 제외한 대부분의 영역에서의 움직임벡터 값의 차이는 크지 않다. 각 블록의 움직임벡터의 추정순서는 대부분 프레임의 위에서 아래로, 왼쪽에서 오른쪽으로 진행되므로, 추정하고자 하는 블록을 기준으로 위로 인접된 블록과 왼쪽으로 인접된 블록의 값은 미리 알 수 있다. 그러므로 현재 움직임벡터의 방향을 결정하는데 위와 같은 주변의 움직임벡터의 방향을 초기의 탐색방향으로 이용하면 탐색횟수를 줄일 수 있다.

본 연구에서 제안하는 움직임추정 방식은 상기와 같은 개념을 이용한 것이며 현재 블록을 기준으로 바로 위쪽과 왼쪽으로 인접된 두 블록의 움직임크기와 방향을 이용한다.

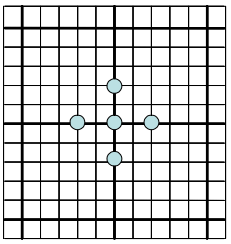


그림 2. 인접블록의 움직임벡터 크기가 3보다 작은 경우와 2단계 이상의 탐색패턴

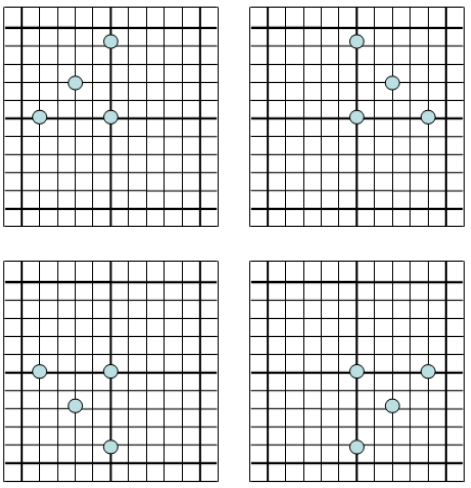


그림 3. 인접블록의 움직임벡터 크기가 3보다 크고 방향이 같을 경우의 quarter diamond 탐색패턴

그림 2는 제안된 방식에서 사용되는 기본적인 탐색패턴으로 두 벡터의 크기가 모두 3보다 작은 경우와 2단계 이상의 탐색시 사용된다. 그림 3은 인접한 두 블록 움직임의 크기가 3보다 크고 두 움직임벡터의 방향이 같은 경우의 탐색패턴이다. 인접한 두 블록의 움직임벡터의 방향이 같으므로 첫 번째 단계의 탐색은 한 방향으로 만의 탐색으로 충분하다. 이때 quarter diamond 패턴으로 탐색을 수행한다. 그림 4는 인접한 두 블록의 움직임벡터 크기가 3보다 크고 두 움직임벡터의 방향이 서로 다른 경우의 패턴이다. 이 경우 첫 번째 단계의 탐색은 두 방향을 탐색할 수 있는 half diamond 패턴으로 탐색을 수행한다. 그림 5는 기본 패턴을 이용하여 탐색할 경우 중심점에서 블록 정합 값이 최소일 때, 기본 패턴의 내부 픽셀의 탐색 위치를 결정하기 위해 주변 픽셀들의 그룹 합을 구하기 위한 그룹화 방법이다.

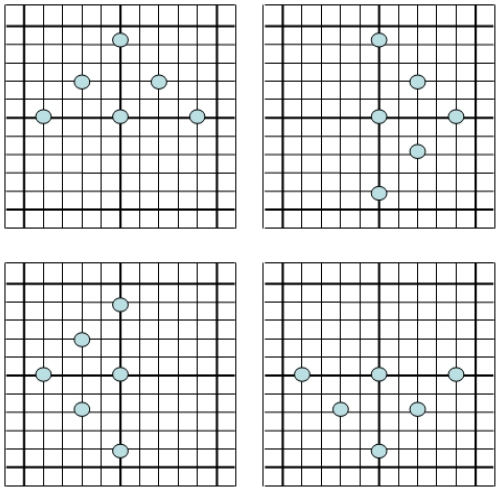
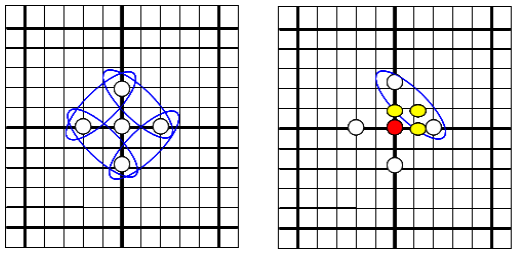


그림 4. 인접블록의 움직임벡터 크기가 3보다 크고 방향이 다른 경우의 탐색패턴



(a) 외부 픽셀의 그룹화 (b) 내부 탐색 영역 결정

그림 5. 내부 탐색영역 결정을 위한 픽셀간의 그룹

제안된 방식은 상기와 같은 탐색패턴을 이용하여 다음과 같은 단계로 움직임추정을 수행한다.

단계 1: 현재 블록의 윗 블록과 왼쪽 블록의 두

벡터의 크기가 3보다 작은 경우 단계 3으로 이동하고, 그렇지 않을 경우 단계 2로 이동한다.

단계 2: 인접한 두 블록의 벡터방향을 비교하여 같으면 그림 2와 같은 탐색패턴을 이용하고, 인접한 두 블록의 벡터 방향이 다르면 그림 4와 같은 탐색패턴을 이용하여 각 탐색 영역의 블록정합오차를 구한다. 블록정합오류가 최소가 되는 점을 탐색영역의 중심점으로 다시 잡고, 단계 3으로 이동한다.

단계 3: 그림 2와 같은 기본 탐색패턴을 이용하여 블록정합을 수행한 후 블록정합오류가 최소인 점을 찾는다. 만약 최소가 되는 점이 중심점일 경우, 단계 4로 이동한다. 그렇지 않을 경우, 최소가 되는 점을 중심점으로 잡고, 다시 이 단계를 반복한다.

단계 4: 그림 5(a)와 같이 외곽의 탐색점들을 묶어, 블록정합오차가 최소가 되는 방향을 움직임 방향으로 결정한 후, 그림 5(b)와 같이 내부의 인접한 픽셀들 중 탐색점 들의 블록정합을 수행하여 블록정합오류가 최소인 점을 찾는다. 이때 찾은 위치가 현재 블록의 움직임벡터가 된다.

그림 6은 인접한 두 블록의 움직임벡터 크기가 3보다 큰 경우, 제안된 방법을 이용하여 움직임벡터를 추정하는 과정의 예이다. 그림 6(a)에서와 같이 인접한 두 블록의 움직임벡터 크기가 3보다 크고, 현재 블록의 움직임벡터가 (0,0) 일 경우 3 단계의 탐색과정을 수행하면 (0,0)의 움직임벡터가 추정됨을 쉽게 알 수 있다. 그림 6(d)와 같이 현재 블록의 움직임벡터가 (0,-4)일 경우, 4단계의 탐색과정을 수행해야 움직임벡터가 추정됨을 알 수 있다.

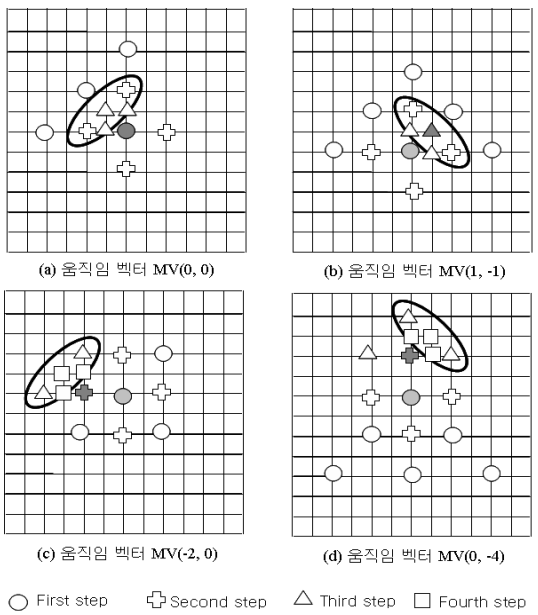


그림 6. 인접블록의 움직임벡터 크기가 3보다 클 경우의 제안된 방식의 움직임 추정과정

Ⅳ. 실험 및 평가

기존의 고속 움직임추정 방식 중 평균적으로 가장 적은 연산을 수행하는 CDS 방식은 9~24의 NSP(number of search points)가 필요하다. 제안된 방식은 8~13의 NSP을 이용하여 고속으로 움직임추정이 가능함을 알 수 있다.

제안된 방식의 타당성을 검증하기 위하여 QCIF (176 x 144) 포맷의 동영상을 이용하여 기존의 고속 움직임추정 방식과 제안된 움직임추정 방식을 비교하였다. 블록의 크기는 16 x 16 이고 탐색범위는 -8에서 +8이다.

실험결과는 표 1과 2와 같으며, 표 1과 2는 다양한 동영상과 다양한 방식을 이용하여 움직임벡터를 추정하는데 필요한 프레임당 평균탐색점수 (ANSP: average NSP)와 평균절대차 (MAD: mean of absolute difference)를 비교한 것이다. hall object, mother & daughter 같이 움직임정도가 1 픽셀 정도로 매우 작은 경우, 제안된 방식은 ANSP가 8이며, 중간정도의 움직임이 있는 miss america, car phone, table tennis와 foreman의 경우, 제안된 방식은 8.23~9.27 의 ANSP가 필요함을 확인할 수 있다. flower garden과 akiyo with crowd와 같은 움직임정도가 큰 영상은 12.29~13.78의 ANSP가 필요함을 알 수 있다.

표 1과 같이 제안된 방식은 모든 동영상에 대하여 기존의 고속 움직임 추정방식보다 ANSP가 작으므로 SAD 연산시 필요한 절대차 연산의 횟수가 작다. 제안된 방식은 전역탐색 (FS) 경우보다 30배 정도의 ANSP의 감소가 있으며, 기존의 고속 움직임추정 방식 중 가장 빠르다고 할 수 있는 CDS의 경우와 비교하여도 1.13(9.04/8)~1.76(21.63/12.29) 배의 연산횟수의 감소가 있었다.

표 2와 같이 제안된 방식은 움직임이 작은 경우 FS 방식과 비교해서 약 1%내의 오차를 가졌으며, 움직임이 많은 경우 기존의 고속 움직임추정 방식들과 비교하여 정합오류값의 큰 저하없이 블록정합 움직임추정시 필요한 탐색점수가 감소됨을 확인하였다. 특히, 움직임이 많은 flower garden이나 akio

표 1. 움직임추정시 필요한 ANSP 값 비교

	FS	TSS	DS	CDS	제안된 방식
hall object	289	25	13.03	9.04	8
mother daughter	289	25	13.03	9.14	8
miss america	289	25	13.33	9.48	8.23
carphone	289	24.95	13.84	9.59	8.43
table tennis	289	24.96	13.71	10.11	8.45
foreman	289	25	14.85	11.93	9.27
flower garden	289	24.41	20.96	21.63	12.29
akiyo_crowd	289	14.37	20.89	19.95	13.78

표 2. 움직임벡터의 MAD 값 비교

	FS	TSS	DS	CDS	제안된 방식
hall object	1.222	1.222	1.222	1.222	1.222
mother daughter	0.800	0.800	0.801	0.801	0.801
miss america	1.041	1.043	1.042	1.042	1.043
carphone	1.778	1.787	1.784	1.793	1.791
table tennis	1.629	1.800	1.630	1.629	1.673
foreman	2.283	2.307	2.283	2.350	2.337
flower garden	10.317	11.476	11.157	11.026	10.765
akiyo_crowd	7.750	8.488	8.432	8.660	8.516

표 3. ANSP x MAD 값 비교

	FS	TSS	DS	CDS	제안된 방식
hall object	353.158	30.550	15.923	11.047	9.776
mother daughter	231.200	20.000	10.437	7.321	6.408
miss america	300.849	26.075	13.890	9.878	8.584
carphone	513.842	44.586	24.691	17.195	15.098
table tennis	470.781	44.928	22.347	16.469	14.137
foreman	659.787	57.675	33.903	28.036	21.664
flower garden	2981.613	280.129	233.851	238.492	132.302
akiyo_crowd	2239.750	121.973	176.144	172.767	117.350

with crowd 동영상의 경우에는 제안된 방식이 CDS 방식보다 오류가 적음을 알 수 있다. ANSP가 감소하여도 MAD 값이 커지면 정합오류값을 표현하고 인코딩하기 위한 알고리즘 상의 부가적인 부하가 존재한다.

따라서, 본 연구에서는 ANSP와 MAD 값을 동시에 고려하는 평가치 (ANSP x MAD)를 기본으로 움직임추정 방식의 최종성능을 비교하였다. ANSP x MAD 값이 작을수록 우수한 움직임추정 방식이라고 판단할 수 있으며, 표 3과 같이 제안된 움직임추정 방식은 기존의 모든 움직임추정 방식보다 ANSP x MAD 값이 작음을 알 수 있다.

V. 결 론

제안된 움직임추정 방식은 CDS 움직임추정 방식을 기반을 둔 것으로써, 인접한 블록의 움직임벡터의 크기와 움직임벡터의 방향을 이용하여 탐색영역의 탐색횟수를 줄임으로써 움직임추정의 고속화를 가능하게 한다. 제안된 방식은 전역탐색 (FS) 경우보다 30배 정도의 평균탐색점수의 감소가 있으며, 기존의 고속 움직임추정 방식 중 가장 빠르다고 할 수 있는 CDS의 경우와 비교하여도 1.13(9.04/8)~1.76(21.63/12.29) 배의 평균탐색점수의 감소가 있었다. 또한 움직임이 많은 flower garden이나 akio with crowd 동영상의 경우에는 제안된 방식이 CDS 방식보다 움직임추정 오류가 적음을 확인하였다. 평균탐색점수와 평균절대값차를 동시에 고려하는 평가치 (ANSP x MAD)를 기본으로 움직임추정 방식의 최종성능을 비교한 결과, 제안된 움직임추정 방식은 기존의 모든 움직임추정 방식보다 ANSP x MAD 값이 항상 작음을 알 수 있다. 이와 같이 제안된 움직임추정 방식은 기존의 고속 움직임추정 방식보다 움직임추정 오류의 큰 증가없이 평균탐색점수가 감소됨으로 움직임추정의 고속화가 가능하여, 고속의 움직임추정이 필요한 동영상 응용분야에 유용하게 적용될 수 있다.

참 고 문 헌

[1] CCITT. Recommendation H.261. Dec. 1990. "Line transmission on non-telephone signals. Video codec for audiovisual services at p x 64 kbits/s".

[2] K. Guttag et al. "A single-Chip Multi-processor For Multimedia: The MVP". IEEE Computer Graphics and Applications, pp 53-64, Nov, 1992.

[3] S. Mietens et al, "Computational-complexity scalable motion estimation for mobile MPEG encoding," IEEE Trans. on Consumer Electronics, vol. 50, no. 1, pp. 281-291, Feb. 2004.

[4] 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. 1981.

[5] R. Li. B. Zeng, and M.L Lion, "a new three step search algorithm for block motion

estimation," IEEE Trans. Circuits and Systems for Video Technology, vol. 4, pp. 438-442, Aug. 1994.

[6] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. on Circuits and Systems for Video Technology, vol. 6, pp. 313-317. June 1996.

[7] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation." IEEE Trans. on Image Processing., vol. 9, no. 2, pp. 287-290, Feb. 2000.

[8] Jeanson Hung; Wen-Sheng Su; Jun-Hua Wang, "Cellular search algorithm for motion estimation", Second International Workshop on Digital and Computational Video, pp. 173-179, Feb. 2001.

[9] Jeanson Hung; Wen-Sheng Su; Jun-Hua Wang, "A novel cellular search algorithm for block matching motion estimation", International Conference on Information Technology: Coding and Computing, Proceedings, pp. 629-633, Apr. 2001.

[10] Chun-Ho Cheung; Lai-Man Po "A novel cross-diamond search algorithm for fast block motion estimation" IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 12, pp. 1168-1177, Dec. 2002.

[11] 강현수, 박성모, "고속 움직임 추정을 위한 움직임 추정 생략 기법," 한국통신학회 논문지, 28권 7호, 페이지 726-732, 2003년 7월.

소 현 호 (Hyeon-Ho So) 정회원
2004년 2월 경희대학교 대학원 전자공학과 석사
2004년 3월~현재 푸른기술 연구소
<관심분야> 비디오코딩, SoC, 디지털시스템설계

김 진 상 (Jinsang Kim) 종신회원
2000년 12월 미국 콜로라도 주립대 전자공학박사
1990년 2월~2001년 8월 KT 연구소
2001년~현재 경희대학교 전자정보학부 조교수
<관심분야> 영상처리와 이동통신용 SoC 설계

조 원 경 (Won-Kyung Cho) 정회원
1986년 8월 한양대학교 전자공학과 공학박사
1980년~현재 경희대학교 전자정보학부 정교수
<관심분야> 컴퓨터시스템 구조, VLSI 설계

김 영 수 (Young-Soo Kim) 정회원
1988년 미국 애리조나 주립대 공학박사
1992년~현재 경희대학교 전자정보학부 정교수
<관심분야> 이동통신, SDR

서 덕 영 (Doug Young Suh) 정회원
1990년 미국 조지아 공대 공학박사
1992년~현재 경희대학교 전자정보학부 정교수
<관심분야> 비디오압축, 멀티미디어 통신