論 文

시각시스템의 Edge 검출용 고속 마스크 Operator

正會員 崔 太 永*

Fast Mask Operators for the edge Detection in Vision System

Tae Young CHOI*, *Regular Member*

要 約 행렬분해에 의하여 edge 검출용 mask operator의 고속계산법을 제시하였다. 다방향성 mask operator의 각방향에 대한 계산결과는 mask 원소와 화상간의 곱한 것을 합산한 것이다. 이것은 각 mask의 원소를 행으로하는 행렬과 mask에 포함되는 화소를 원소로하는 벡터와의 곱과 마찬가지여서 1차원 신호변환으로도 취급할 수 있다. 본 논문에서는 Sobel 및 Prewitt operator의 경우에 대하여 변환행렬을 구한 다음 분해하여 계산량을 산출하여 직접 계산할 때의 방법과 비교하였다. 이 결과 여분의 기억점을 사용치 않고서도 이 방법에 의한 계산량은 Sobel 및 Prweitt operator의 경우에 각각 직접 계산량의 42.86%, 50%밖에 되지 않았다. 그리고 100×100 화소의 화상인 경우에 301개의 여분의 기억점을 사용할때에 Sobel opreator는 직접 계산량의 35.93%로 줄어 들었다.

ABSTRACT A newmethod of fast mask operators for edge detection is proposed, which is based on the matrix factorization. The output of each component in the multi-directional mask operator is obtained adding every image pixels in the mask area weighting by corresponding mask element. Therefore, it is same as the result of matrix-vector multiplication like one dimensional transform, i, e., transform of an image vector surrounded by mask with a transform matrix consisted of all the elements of each mask row by row. In this paper, for the Sobel and Prewitt operators, we find the transform matrices, add up the number of operations factoring these matrices and compare the performances of the proposed method and the standard method. As a result, the number of operations with the proposed method, for Sobel and Prewitt operators, without any extra storage element, are reduced by 42.85% and 50% of the standard operations, respectively and in case of an image having 100×100 pixels, the proposed Sobel operator with 301 extra storage locations can be computed by 35.93% of the standard method.

1. 서 론

Edge 검출은 비젼시스템에서 물체 인식의 가장 중요한 기본 과정에 속한다. Edge는 화면 내의 물체의 형태 또는 크기를 잘 나타낼뿐 아니라 edge 검출과정자체가 데이타 압축작용도 한다.

여러가지 edge 검출방법(Laplacian, mask ,operator, edge fitting 등[1]~[5])이 잘 알려져 있으나 edge 검출시간을 고려한 연구는 근자에 mask operator의 하나인 Sobel operator에 관한것이 있지만

* 亞洲大學校 工科大學 電子科
Dept. of Elec. Eng., Ajou Univ., Suwon, 170 Korea.
論文番號: 86－29 (接受 : 1986. 8. 7)

[6]~[8] 각종 edge 검출법의 고속 계산법의 연구는 필요하다.

본 논문에서는 mask operator에 의한 edge 검출 시간을 줄이기 위한 하나의 새로운 방법으로 mask operator의 연산과정을 행렬화한 다음 이를 부분행렬의 곱으로 나타냄으로서 전체적인 계산량을 줄이고자한다. 그 예로서 4 방향성 Sobel 및 Prewitt operator의 경우에 적용하였다.

본 논문의 구성은 Ⅱ장에서 mask operator의 행렬화 과정을 다루고, Ⅲ장에서는 Sobel과 Prewitt operator의 행렬 및 부분행렬을 구하여 계산량을 구하고, Ⅳ장에서는 Sobel과 Prewitt operator의 상호관계를 구하였고 마지막 결론순으로 되어있다.

Ⅱ. Mask operator의 행렬화

기존의 edge 검출용 mask operator의 mask의 크기는 대개 3 × 3 이고 mask의 종류는 edge의 방향성을 고려하여 대개 4 또는 8 이다. 여기서 일반성을 부여하기 위하여, operator의 mask 의 크기를 S, mask의 종류를 K개, mask에 일치되는 화소를 $p(l)$ ($l=1,2,3,\cdots, S$), 화소 $p(l)$ 에

$$g(j)=\sum_{l=1}^{S} wj(l)p(l),\quad j=1,2,\cdots,\ K \tag{1}$$

$$G=WP \tag{2}$$

여기서, $G=[g(1)\ g(2)\cdots g(j)\cdots g(K)]^t$

$$W=\begin{bmatrix} w1(1) & w1(2)\cdots & w1(l)\cdots & w1(S) \\ w2(1) & w2(2)\cdots & w2(l)\cdots & w2(S) \\ \cdot & \cdot & j & \cdot \\ wj(1) & wj(2)\cdots & wj(l)\cdots & wj(S) \\ K\cdot & K\cdot & & \cdot \\ wK(1) & wK(2)\cdots & wK(l)\cdots & wK(S) \end{bmatrix}$$

$P=[p(1)\ p(2)\cdots p(l)\cdots p(S)]^t$

이다.

해당하는 j 번째 mask의 원소를 $wj(l)(j=1,2,\cdots, K)$라고 한다면(그림 1 : S = 9 (3 × 3인 mask의 예)), 이 edge 검출 operator의 K개의 mask에 의한 계산결과 g(j)는 식(1) 또는 식(2)처럼 행렬식으로 표시할 수 있다.

wj(1)	wj(2)	wj(3)
wj(8)	wj(9)	wj(4)
wj(7)	wj(6)	wj(5)

p(1)	p(2)	p(3)
p(8)	p(9)	p(4)
p(7)	p(6)	p(5)

$j=1,2,\cdots, K$

$S=9$

그림 1 마스크와 화소의 원소배열
Arrangement of the elements of mask and pixels.

만약 식(2)의 W의 원소 $wj(l)$이 모두 1 또는 −1 이라면 계산량(합 또는 차)은 K (S−1) 이다. 그런데 식 (2)의 W를 식(3)처럼 부분행렬의 곱으로 나타낼 수 있다면 전체적인 계산량은 부분행렬 W_q의 원소에 따라 감소할 수도 있다[9].

$$W=W_1\cdot W_2\cdots W_q \tag{3}$$

예를 들어 4 점 Hadamard변환행렬을 부분행렬의 곱으로 나타내면 식 (4)의 결과를 얻을 수 있는데, 4 × 1 의 신호벡터를 식 (4)의 좌측행렬(원래의 Hadamard 행렬)로 변환할 때의 계산량은 12 회이나 식 (4)의 우측 부분행렬의 곱으로 변환할 때의 계산량은 8 회로 4 회가 감소됨을 알 수 있다.

$$\underset{W}{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}} = \underset{W_1}{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}} \underset{W_2}{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}} \tag{4}$$

이러한 부분행렬화 과정은 원행렬(W)의 서로 다른 행(row) 간의 공통원소를 찾아 W_q의 행으

로 놓은 다음에 W_q의 행간의 선형합으로 W의 행을 나타낼 수 있도록 W(q−1)을 정한다. 다음 W(q−1) 행렬이 서로 다른 행간에 공통원소가 있다면, W(q−1) 행렬을 새로운 원행렬로 보고 위의 과정을 반복하여 부분행렬의 곱으로 고친다.

(예)

식(4)의 W의 1,3행의 1,2열의 원소가 공통이다. 따라서 (1100)을 W_2의 하나의 행으로 놓는다. 또한 W의 1,3행의 3,4열의 원소가 공통이므로 마찬가지로 (0011)을 W_2의 또 다른 행으로 놓는다. 이와같이 W의 2,4행의 경우에도 공통원소를 찾아 W_2의 행으로 놓으면 식(4)의 W_2을 얻는다.

다음, W_1의 행렬원소 산출법을 알아보자.

W의 3행은 (1 1−1−1)은 W_2의 1행 (1100)과 2행(0011)의 차이므로 W_1의 3행은(1-1 00)이다. 위와 마찬가지로 나머지 W_1의 행을 쉽게 구할 수 있다. 그런데 W_1 행렬은 이제는 더 서로 다른 행 간에 공통원소가 없으므로 부분행렬의 곱으로 나타낼 수가 없다.

Ⅲ. Sobel 및 Prewitt operator

Sobel 및 Prewitt operator의 마스크에서 변환행렬과 부분행렬을 구하는 방법을 고찰한다.

Ⅲ. 1 Mask의 구조

대칭 operator로서는 Sobel과 Prewitt operator를 들 수 있는데[1]-[5], 원래는 8방향성이나 대칭성 때문에 4개의 마스크만 취급하면 된다(45도 간격으로 4개만 취한다). 이들 4개의 mask는 그림 2와 같고 mask의 원소 wj(l)은 식(5)로 표시할 수 있다(p(l)의 순서는 그림 1과 같고, p(9)는 사용치 않으므로 화소는 p(1)에서 p(8)까지만 있음).

$$wj(l)=[C(l+j)]\,R(l),\quad j=1,2,3,4,\quad l=1,2,\cdots,8 \tag{5}$$

여기서 C(l)은 유한 sequence ($0<l<7$) 로 {C(0), C(1), C(2), C(3), ⋯, C(7)} = {−1, 0, 1, a, 1, 0, −1, −a}의 값을 갖고 [C(l+j)]는 circular shift[10]를 뜻하며 R(l)은 l =1, 2, ⋯, 8에서 1의 값을 갖고 나머지는 0인 rectangular sequence, 즉 rectangular window를 뜻한다.

K = 1

−1	−a	−1
0		0
1	a	1

K = 2

−a	−1	0
−1		1
0	1	a

K = 3

−1	0	1
−a		a
−1	0	1

K = 4

0	1	a
−1		1
−a	−1	0

그림 2 4방향성 Sobel(a=2)및 Prewitt(a=1) operator의 마스크
4 directional masks of Sobel(a=2) and Prewitt (a=1) operators.

Ⅲ. 2 부분행렬에 의한 계산량 산출

Sobel과 Prewitt의 행렬을 구분하기 위하여 각각의 행렬을 W_S, W_P로 첨자 s와 p를 사용한다. 식(2), (5) 및 그림 2에서 W_S 및 W_P의 행렬을 구하면 각각 식(6), (7)과 같다.

$$W_S=\begin{pmatrix} -1 & -2 & -1 & 0 & 1 & 2 & 1 & 0 \\ -2 & -1 & 0 & 1 & 2 & 1 & 0 & -1 \\ -1 & 0 & 1 & 2 & 1 & 0 & -1 & -2 \\ 0 & 1 & 2 & 1 & 0 & -1 & -2 & -1 \end{pmatrix} \tag{6}$$

$$W_P=\begin{pmatrix} -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ -1 & -1 & 0 & 1 & 1 & 1 & 0 & -1 \\ -1 & 0 & 1 & 1 & 1 & 0 & -1 & -1 \\ 0 & 1 & 1 & 1 & 0 & -1 & 1 & -1 \end{pmatrix} \tag{7}$$

식(6), (7)의 각 행의 공통원소들을 찾아 부분행렬로 고쳐 보기로 한다.

(1) W_s의 경우

식 (6)을 부분행렬의 곱으로 고치면 식(8)의 결과를 얻을 수 있다.

$$W_s = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & -1 \\ 1 & 0 & -1 & -2 \\ 0 & -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & \overset{W3}{\downarrow}\,0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & -1 & 0 \end{bmatrix}}_{W1} \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix}}_{W2} \cdot W3$$

$$= W1 \,.\, W2 \,.\, W3 \qquad (8)$$

식(8)에서 W1, W2, W3는 모두 각 행의 두 원소가 1 또는 −1이고 나머지 원소는 0이므로 식(2), (3)에서 W3. P계산시 계산량은 4이다 (W3. P는 수직 vector).

W3. P계산후에 W2. W3. P를 계산할 때에 다시 4번의 계산량이 W1. W2. W3. P계산시에 또다시 4회가 필요하므로 한 화소에 대한 4방향성 Sobel operator의 전체적인 계산량은 12번이다(표1의 1번 참조).

여기서 이 방법은 따로 기억자리를 둘 필요가 없으면서도 계산량은 참고문헌(8)의 결과보다 적다(참고문헌(8)에서 (N+9)개의 기억자리를 사용하여 한 화소 계산에 약 13번 계산량이 필요함).

위 식 (8)에 의한 W_s. P 계산법의 흐름도는 그림 3과 같다.

한편 식 (8)에서 W2. W3를 계산하면 식(9)로 된다.

$$W2 \,.\, W3 = \begin{bmatrix} -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \end{bmatrix} \qquad (9)$$

W2. W3의 각 행 순서대로 다시 mask화하면 그림 4와 같이 되는데, 각 mask는 수평, 수직으로 인접한 두 원소들로 되어 있어 계산과정에서 수직, 수평 인접화소를 합한 것을 기억시켜 재사용한다면 계산량은 다음의 기술과 같이 더욱 줄일 수 있다.

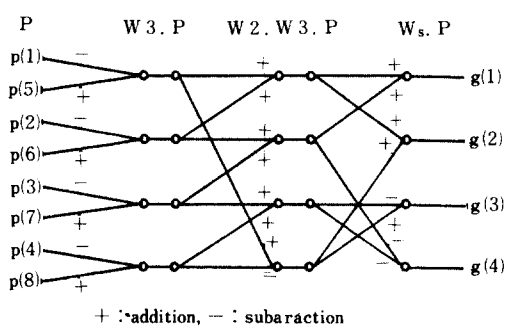


그림 3 식 (8)의 부분행렬에 의한 W_s. P 계산법의 흐름도
Flow graph of W_s. P using matrix factorization method by eq. (8).

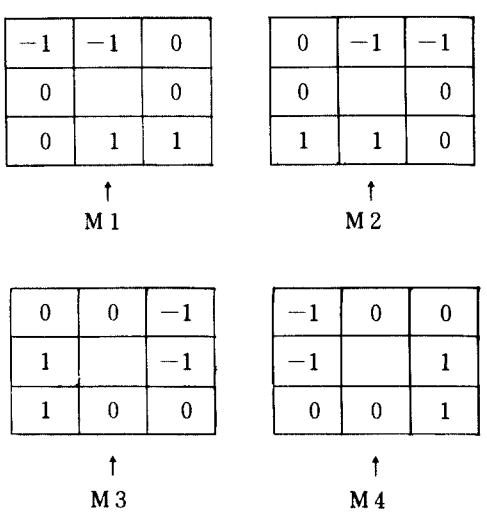


그림 4 W2. W3에 의한 마스크
Representation of two matrices multiplication results (W2. W3) as masks.

그러면 마스크를 수평, 수직 이동하여 계산할 때의 두 과정으로 나누어 기억점의 수 및 계산량을 산출하여 보자.

1) 기억점수 산출

가) 수평이동

그림 4 에서

A) M 1의 우하의 두 수평인접원소에 의한 합산결과는 우측 인접화소를 계산할 때에 M 2 의 좌하 수평합과 동일하므로 기억시켜 사용한다(기억점 1개).

B) M 2의 우상의 수평합은 우측 인접화소 계산시 M 1의 좌상의 수평합으로 사용할 수 있어 기억시켜 사용한다(기억점 1개).

C) M 3의 우상의 수직합은 우측 2점 떨어진 화소 계산시 M 4 의 좌상의 수직합으로 재사용되므로 기억시킨다(기억점 2 개).

D) M 4 의 우하 수직합은 우측 2 점 떨어진 화소 계산시 M 3 의 좌하의 수직합으로 재사용되므로 기억시킨다(기억점 2개).

이상에서 M 1의 우하의 수평합과 M 2의 우상의 수평합은 우측 인접화소 계산시 바로 재사용되므로 기억자리는 2개가 필요하고 M 3의 우상의 수직합과 M 4의 우하 수직합은 우측으로 2 점 떨어진 화소에 재사용되므로 기억자리는 4개이다.

나) 수직이동

수평이동의 계산과 동일한 방법으로 그림 4 에서 기억점을 산출한다.

A) M 1의 우하 수평합은 수직으로 2 점 아래 화소 계산시 M 2의 우상의 수평합으로 M 2 의 좌하수평합은 수직으로 2 점아래 화소 계산시 M 1의 좌상의 수평합으로 재사용되므로 기억점은 2 (N－1) 개이다(N은 한줄 화상의 화소수).

B) M 3의 좌하 수직합은 수직으로 인접한 화소 계산시에 M 4 의 좌상의 수직합으로, M 4 의 우하 수직합은 수직으로 인접한 화소 계산시에 M 3의 우상의 수직합으로 재사용되므로 기억점은 N개이다.

다) 수평, 수직이동의 종합계산

위 가), 나)에서, 가)－A의 과정은 나)－A와 중복되므로 가)－A의 기억점은 따로 기억점이 필요없다. 가)－D의 과정은 나)－B와 중복되므로 기억점이 필요없다. 가)－B 및 가)－C는 나)와 무관하므로, 기억점이 3개 필요하다. 따라서 전체적인 기억점수는 (3N＋1) 개이다. (2N－2＋N＋3).

2) 계산량

우선 화상의 화소수를 M×N이라하자(M줄, N점/줄). 위 1)의 과정에서 수직, 수평인접화소의 합을 계산한 뒤에 기억시켜서 사용하므로, 결과적으로 그림 4 의 각 마스크의 계산은 수직, 수평인접 화소의 합을 한번씩만 계산하면 된다. 그러면 이들의 계산량을 구해보면, 우선 수직인접화소의 합계산에는 (M－1)×N번의 계산량이 필요하다. 왜냐하면 하나의 수직 인접화소 합계산에 1번의 계산량이 필요하고 전체 화상에 대한 수직인접 화소의 합계산이 (M－1)×N 번있기 때문이다. 마찬가지로 수평인접 화소 합계산에도 M×(N－1) 번의 계산량이 필요하다.

그리고 식(9)의 W 2. W 3. P 계산에는 4(M－2)(N－2) 번의 계산량이 필요하다. 왜냐하면 그림 4 에서 알 수 있듯이 식(9)의 계산은 mask M 1, M 2, M 3, M 4에 대한 계산과 같으므로인접화소의 합 간의 합산이므로 각 마스크에 1회씩 계산량이 필요하기 때문이다.

따라서 W 2. W 3. P 계산에는 (3N＋1) 개의 여분의 기억점을 사용할 때에 필요한 계산량은 4(M－2)(N－2)＋M(N－1)＋N(M－1)이다. 그리고 식 (8)에서, W 2. W 3. P 계산후에 W 1. W 2. W 3. P의 계산에는 4(M－2)(N－2)의 계산량이 추가로 필요하다.

결과적으로 4 방향성 Sobel operator 계산에는 (3N＋1)개의 기억점을 사용할 때에 필요한 전체 계산량은 8(M－2)(N－2)＋N(M－1)＋M(N－1)이 되어 한 화소를 계산할 때에 약10번의 계산량이 필요함을 알 수 있다(표 1의 2번).

M＝N＝100인 화상인 경우 Sobel Operator인

경우 표 1의 1번 및 2번 방법은 직접적인 계산 방법의 계산량의 각각 42.86%, 35.93%밖에 되지 않는다.

표 1 (M×N) 화상에 대한 계산량
Number of operations for M×N size image.

Operator	번호	기억점수	계 산 량	직접적인 계산량
Sobel	1	0	12(M−2)(N−2)	28(M−2)(N−2)
	2	3N−1	8(M−2)(N−2)+N(M−1)+M(N−1)	
Prewitt	3	0	10(M−2)(N−2)	20(M−2)(N−2)

(2) W_P의 경우

식 (7)을 부분행렬의 곱으로 고치면 식 (10)의 결과를 얻는다.

$$W_P = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & -1 \\ 0 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (W3)$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot W3$$

(W1) (W2)

$$= W1 \cdot W2 \cdot W3 \qquad (10)$$

식 (10), (2)에서 계산량은, W3 . P 계산에 4번, W2 . W3 . P 계산에 추가로 4번의 계산량이 필요하여 W_P. P 계산에 필요한 계산량은 10번이다.

따라서 전체 화상에 필요한 계산량은 10(M−2)(N−2)이고 기억점은 필요없다(표 1의3번). 직접적인 계산법에 비하여 50%로 계산량은준다.

Ⅳ. Sobel 및 Prewitt Operator의 관계

그림 2에서 Sobel 및 Prewitt의 mask를 K의 값에 따라 각각 M_SK, M_PK, 그리고 식 (1)의 $g(j)$를 각각 $g_S(j)$, $g_P(j)$라 하면 식(11), (12)의 관계를 얻을 수 있다.

$$\begin{aligned} M_S1 &= M_P2 - M_P4 \\ M_S3 &= M_P2 + M_P4 \\ M_S2 &= M_P3 + M_P1 \\ M_S4 &= M_P3 - M_P1 \end{aligned} \qquad (11)$$

$$\begin{pmatrix} g_S(3) \\ g_S(1) \end{pmatrix} = H \begin{pmatrix} g_P(2) \\ g_P(4) \end{pmatrix}, \quad \begin{pmatrix} g_P(2) \\ g_P(4) \end{pmatrix} = \frac{H}{2} \begin{pmatrix} g_S(3) \\ g_S(1) \end{pmatrix}$$

$$\begin{pmatrix} g_S(2) \\ g_S(4) \end{pmatrix} = H \begin{pmatrix} g_P(3) \\ g_P(1) \end{pmatrix}, \quad \begin{pmatrix} g_P(3) \\ g_P(1) \end{pmatrix}^{(3)}_{(1)} = \frac{H}{2} \begin{pmatrix} g_S(2) \\ g_S(4) \end{pmatrix} \qquad (12)$$

여기서, $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ 이다.

윗 식에서 Sobel 및 Prewitt operator는 상호 선형변환관계가 있음을 알 수 있다. 이들 2가지 operator를 동시에 계산할 필요성이 있을 경우에는 먼저 Prewitt operator를 계산하여 선형변환하여 Sobel operator를 계산하는 것이 그 반대의 경우보다 계산량면에서 효과적임을 알 수 있다.

Ⅴ. 결 론

Edge 검출방법의 하나인 mask operator의 연산법의 고속화의 일환으로 mask operator의 연산과정을 행렬수식으로 나타내고 이 행렬을 부분행렬의 곱으로 나타냄으로서 전체적인 계산량을 감소할 수 있음을 4방향성 Sobel 및 Prewitt operator에 적용한 결과로 알 수 있었다.

100×100 화상에 이 방법을 적용한 결과, Sobel operator의 경우 여분의 기억점을 301개를 사용

할 때에 전체계산량은 직접계산량의 36%밖에 되질 않았다. 또한 여분의 기억점을 하나도 사용치 않고서도 **Sobel** 및 **Prewitt operator**의 계산량은 화상의 크기에 관계없이 각각을 직접 계산할때의 계산량의 43%, 50%밖에 되지 않았다.

본 논문에서 제시한 **mask operator**의 연산의 행렬식 표현은 본문에서 다룬 4방향성보다 더 많은 다방향성(8방 또는 12방) **mask operator**에 더욱 효과적으로 적용할 수 있으리라 본다.

참 고 문 헌

(1) T. Paylidis, Structural pattern recognition, Springer-Verlag Berlin Heidelberg New York, 1977.

(2) W.K. Pratt, Digital image processing, John Wiley & Sons, Inc.,

(3) A. Rosenfeld and A.C. Kak, Digital Picture Processing, 2nd ed., D. Reidel Publishing, 1982

(4) J. Kittler et al., Pattern recognition theory and applications, D. Reidel Publishing Company, 1982

(5) M.D. Levine, Vision in man and machine, McGraw-Hill, Inc., 1985

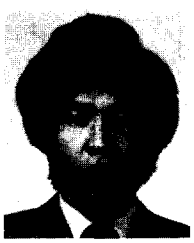
(6) C.C. Lee, "Elimination of Redundant operations for a Fast Sobel Operator," IEEE Trans., Sys., Man, Sybern., vol. SMC-13. No. 3, pp. 242-245, Mar./Apr. 1983

(7) P.D. Piction, "Comment on Elimination of Redundant Operations far a Fast Sobel Operator," IEEE Trans., Sys., Man, Sybern., vol. SMC-14, No. 3, pp. 560-561, May/June, 1984

(8) 최우영, 박래홍, "Sobel Operator의 고속 처리 방법," 대한 전자공학회 추계종합학술대회 논문집, vol. 8, No. 2, pp. 239-241, 1985년 11월

(9) W.M. Gentleman. "Matrix Multiplication and Fast Fourier Transforms," B.S.T.J., pp. 1099-1103, July/Aug., 1968

(10) A.V. Oppenheim and R.W. Schafar, Digital Signal Processing, Prentics-Hall, Inc., 1975



崔 太 永(Tae Young CHOI) 正會員

1950年10月24日生

1970年2月～1974年2月：서울大學校工科大學電子工學科卒業(工學士)

1976年9月～1978年8月：서울大學校大學院電子工學科(工學碩士)

1979年10月～1982年12月：프랑스 Aix-Marseill 3 大學(工學博士)

1978年9月～現在：亞州大學校工科大學電子工學科助敎授