네 방향 마스크를 가진 Sobel 연산자의 고속처리 방법

(Fast Methods for a Sobel Operator with Four Directional Masks)

崔 祐 榮*, 朴 來 弘**

(Woo Young Choi and Rae Hong Park)

要 約

네 방향 마스크를 가진 Sobel 연산자의 두가지 고속처리 방법을 제시하였다. 이 방법들은 적은 부가의 저장 장소를 사용함으로써 연산 시간을 50% 이상 줄일 수 있었다. 또한 기존의 연산방법과 제안된 고속처리 방법의 비교결과를 제시하였다.

Abstract

Two fast methods for a Sobel operator with four directional masks are proposed. They reduce the computation time more than 50% at the expense of a small extra storage locations. Comparisons of the original method and the proposed fast methods are included.

Ⅰ. 序 論

화상분할(image segmention)을 위해 사용되는 경계추출(edge detection)방법에는 여러가지 방법들이 있다.[1] 그 중에는 gradient 형태인 Sobel 연산자가 있는데 수행면에서 좋은 결과를 얻을 수 있어서 많이 사용되고 있다. 그러나 이 Sobel 연산자는 반복되는 계산과정을 포함하기 때문에 계산 시간면에서 많은 낭비가 있다.

본 고에서는 Lee가 제안한 수평, 수직방향 성분을 가진 Sobel 연산자의 고속처리 방법[2]에 기초를 둔 네 방향의 경계추출을 위한 두가지 고속처리 방법에 대해서 기술하고 있다. 즉, Lee의 방법을 확장시켜 Sobel 연산자에서 수평, 수직, 대각선, 반대각선 방향의 경계를 추출할 때 낭비되는 계산량을 크게 줄일 수가 있었다.

*準會員, **正會員, 西江大學校 電子工學科

(Dept. of Elec. Eng., Sogang Univ.)

接受日字 : 1986年 2月 27日

Ⅱ. 용어 정의

Sobel 연산자 S(m, n)은 그림 1과 같은 네 방향의 경계를 추출하는 3×3 마스크를 화소(m, n)을 중심으로 한 화상에 적용하여 그중 가장 큰 값을 갖는 방향의 성분을 경계값(edge value)으로 정한다. 이를 식으로 나타내면 아래와 같다.[3]

$$S(m,n)=MAX\{|dx(m,n)|,\ |dy(m,n)|,\ |d45(m,n)|,\ |d135(m,n)|\}, \quad (1)$$

여기서

$$dx(m,n)=\{f(m-1,n+1)+2f(m,n+1)+f(m+1,n+1)\}-\{f(m-1,n-1)+2f(m,n-1)+f(m+1,n-1)\}, \quad (2)$$

$$dy(m,n)=\{f(m+1,n-1)+2f(m+1,n)+f(m+1,n+1)\}-\{f(m-1,n-1)+2f(m-1,n)+f(m-1,n+1)\}, \quad (3)$$

$$d45(m,n)=\{f(m-1,n)+2f(m-1,n+1)+f(m,n+1)\}-\{f(m,n-1)+2f(m+1,n-1)+f(m+1,n)\}, \quad (4)$$

$$d135(m,n)=\{f(m+1,n)+2f(m+1,n+1)+f(m,n+1)\}-\{f(m,n-1)+2f(m-1,n-1)+f(m-1,n)\} \quad (5)$$

으로 주어지며 f(m,n)은 (m,n) 점에서의 빛의 밝기(gray level)를 나타낸다.

f(m−1, n−1)	f(m−1, n)	f(m−1, n+1)
f(m, n−1)	f(m, n)	f(m, n+1)
f(m+1, n−1)	f(m+1, n)	f(m+1, n+1)

−1	0	1	−1	−2	−1	0	1	2	−2	−1	0
−2	0	2	0	0	0	−1	0	1	−1	0	1
−1	0	1	1	2	1	−2	−1	0	0	1	2

수평방향 수직방향 45° 방향 135° 방향

그림 1. 일반적인 3×3마스크 형태와 각 방향의 마스크

Fig. 1. General 3×3 mask form and four directional masks.

본 고에서 제안한 고속 알고리즘 유도과정의 이해를 돕기 위해서 3가지 용어를 정의하면 다음과 같다.[2] 첫째로 중간합(intermediate sum)은 수평 또는 수직방향으로 연속된 두 화소의 빛의 밝기의 합으로 정의한다. 즉,

$Ir(m,n)=f(m,n)+f(m,n+1)$,

단 $1\le m\le M,\ 1\le n\le N-1$,

$Ic(m,n)=f(m,n)+f(m+1,n)$,

단 $1\le m\le M-1,\ 1\le n\le N$.

둘째로 부분합(partial sum)은 3×3 마스크에서 그 중심인 점(m,n)을 제외한 둘레의 8개 화소중 어느 한 화소의 성분을 공통으로 포함하고 있는 두 중간합의 합으로 정의한다. 이때 주의할 점은 Lee[2]는 부분합을 수평방향의 두 중간합의 합 또는 수직방향의 두 중간합의 합으로 정의를 내렸다. 그러나 본 고에서는 4방향의 경계성분을 고려하기 위하여 그 의미를 확대하여 수평방향의 중간합과 수직방향의 중간합 즉, 방향이 다른 두 중간합의 합도 부분합으로 정의하였다.

마지막으로 부분차(partial difference)는 다음과 같이 정의된다.[4]

$Pdx(m,n)=Ic(m,n+1)-Ic(m,n-1)$,

단 $1\le m\le M-1,\ 2\le n\le N-1$,

$Pdy(m,n)=Ir(m+1,n)-Ir(m-1,n)$,

단 $2\le m\le M-1,\ 1\le n\le N-1$.

본 고에서 마스크는 좌측에서 우측으로 이동시키면서 적용시키고 맨 우측열에 도달하면 한 행 아래로 이동하여 최하단 맨 우측열에 도달할 때까지 반복 수행한다고 가정한다.

Ⅲ. 반복 계산과정의 제거

식 (2)-(5)를 앞 절에서 정의한 중간합의 형태로 나타내면 다음과 같다.

$$dx(m,n)=Ic(m-1,n+1)+Ic(m,n+1)-Ic(m-1,n-1)-Ic(m,n-1), \quad (2a)$$

$$dy(m,n)=Ir(m+1,n-1)+Ir(m+1,n)-Ir(m-1,n-1)-Ir(m-1,n), \quad (3a)$$

$$d45(m,n)=Ir(m-1,n)+Ic(m-1,n+1)-Ic(m,n-1)-Ir(m+1,n-1), \quad (4a)$$

$$d135(m,n)=Ir(m+1,n)+Ic(m,n+1)-Ic(m-1,n-1)-Ir(m-1,n-1). \quad (5a)$$

윗 식중 앞의 두식 (2a)와 (3a)를 뒤의 두식 (4a)와 (5a)와 비교해 보면, 앞의 두 식에서 사용된 중간합들이 뒤의 두 식에서 중복되어 사용되었음을 알 수 있다. 그러므로 dx(m,n)과 dy(m,n)을 계산할 때 사용한 중간합을 저장하여 d45(m,n)과 d135(m,n)을 계산시 다시 사용함으로써 계산과정을 줄일 수 있다.

다음은 dx(m,n)과 dy(m,n)을 계산하는데 있어서 중복되는 계산과정을 줄일 수 있는 방법에 대해서 설명하고자 한다.

첫째로 dy(m,n)을 계산하는 경우에, dy(m,n+1)은 마스크를 우측으로 한 열 이동시킴으로써 계산할 수 있는데 이것을 중간합의 형태로 나타내면 다음과 같다.

$$dy(m,n+1)=Ir(m+1,n)+Ir(m+1,n+1)-Ir(m-1,n)-Ir(m-1,n+1). \quad (6)$$

식 (3a)와 식 (6)을 비교해 보면 중간합 Ir(m+1, n)과 Ir(m−1, n) 즉, 부분차 Pdy(m,n)(=Ir(m+1, n)−Ir(m−1, n))이 중복되어 사용되고 있음을 알 수 있다. 따라서 dy(m,n)을 계산할 때 얻은 부분차 Pdy(m,n)을 저장하여 dy(m,n+1)을 계산할 때 다시 사용할 수 있다. 또한 dy(m,n)에서 얻은 두 개의 중간합 Ir(m+1, n)과 Ir(m−1, n)도 저장하여 d45(m,n+1)과 d135(m,n+1)을 계산할 때 각각 다시 사용할 수 있다. 그러므로 이들 부분차와 중간합들은 단 한번씩만 계산하면 되므로 한 행에 대해서 다음 S(m,n+1)을 계산할 때는 반복 계산하지 않아도 되게 된다.

둘째로 dx(m,n)를 계산하는 경우에, 마스크를 우측으로 두 행 이동시켰을 때 얻는 dx(m,n+2)를 중간합 형태로 나타내면 다음 식과 같다.

$$dx(m,n+2)=Ic(m-1,n+3)+Ic(m,n+3)-Ic(m-1,n+1)-Ic(m,n+1). \quad (7)$$

식 (2a)와 식 (7)을 비교해 보면 두 개의 중간합 Ic (m−1, n+1)과 Ic(m, n+1)이 서로 중복되어 사용되었음을 알 수가 있다. 즉, 부분합(Ic(m−1, n+1)+Ic(m, n+1))이 반복 사용되고 있음을 의미하므로 이 부분합을 저장하여 dx(m, n+2)를 계산시 다시 사용할 수 있다. 또한 두 중간합 Ic(m, n+1)과 Ic(m−1, n+1)도 d45(m, n+2)와 d135(m, n+2)를 계산시 다시 사용할 수 있다. 그러므로 이들 부분합과 중간합들은 한 행에 대해서 두 열 이동한 S(m, n+2)을 계산할 때 반복 계산하지 않아도 되게 된다.

Ⅳ. 제안된 두가지 고속처리 방법

≫고속처리 방법 1≪

이 방법에서는 9개의 저장장소(storage)를 필요로 한다. 즉, Ⅲ절에서 설명한 바와 같이 dy(m, n) 경우 두 개의 중간합과 한개의 부분차를 저장하기 위한 세 개의 저장장소와 dx(m, n)경우 네개의 중간합과 두개의 부분합을 저장하기 위한 여섯개의 저장장소가 필요하게 된다.

a) dy(m, n)−S(m, n)에서 dy(m, n)은 두 개의 인접된 부분차를 더해줌으로써 얻을 수 있다. 부분차를 구하기 위해 두 개의 중간합의 차를 구해야 하므로 이때 한 개의 뺄셈이 필요하고 이 두 중간합을 구하는데 각각 한 개의 덧셈이 필요하다.

Ⅲ절에서 설명한 바와 같이 dy(m, n)에서 계산된 부분차는 한 개의 저장장소에 저장하였다가 dy(m, n+1)을 계산시 사용하면 되므로 화상의 한 행에 대하여 필요한 계산수는 중간합을 계산하기 위한 2(N−1) 개의 덧셈과 부분차를 위한 (N−1)개의 뺄셈과 두 부분차를 합하기 위한 (N−2)개의 덧셈이 필요하게 된다. 또한 dy(m, n)에서 계산된 두 중간합을 두 개의 저장장소에 저장하여 d45(m, n)과 d135(m, n)을 구하는데 사용할 수 있다.

b) dx(m, n)−dx(m, n)과 dx(m, n+1)의 우측열의 부분합은 각각 dx(m, n+2)와 dx(m, n+3)의 좌측열의 부분합과 같다. dx(m, n)과 dx(m, n+1)에서 구한 이 두 부분합을 두 개의 저장장소에 저장하였다가 dx(m, n+2)와 dx(m, n+3)을 계산할 때 다시 사용할 수 있으므로 화상의 한 행에 대해서 부분합을 한번씩만 계산하면 된다. 한 개의 부분합을 계산하기 위해서는 두 개의 중간합의 합을 구해야 하므로 한 개의 덧셈이 필요하고 두 중간합의 계산을 위해서 각각 한 개의 덧셈이 필요하다. 따라서 화상의 한 행에 대해서 부분합의 계산을 위해 3N개의 덧셈과 부분합의 차를 계산하기 위해 (N−2)개의 뺄셈이 필요하다. 또한 dx(m, n)에서 계산된 두 중간합을 두 개의 저장장소에 저장하여 d45(m, n)과 d135(m, n)을 구하는데 사용할 수가 있다.

c) d45(m, n)과 d135(m, n)−Ⅲ절에서 살펴본 바와 같이 dx(m, n)과 dy(m, n)에서 구한 중간합을 이용하여 d45(m, n)과 d135(m, n)의 부분합을 계산하는데 각각 두 개의 덧셈이 필요하고, 두 부분합의 차를 계산함으로써 d45(m, n)과 d135(m, n)을 구하게 되므로 각각 한 개의 뺄셈이 필요하다. 그러므로 한 행에 대해서 부분합 계산을 위해 2(N−2)개의 덧셈과 두 부분합의 차를 위해 (N−2)개의 뺄셈이 필요하다.

위의 모든 계산 과정들은 화상의 모든 행에 대해서 반복적으로 계산되어야 하므로 M×N 화상 경우 모든 dy(m, n)을 구하기 위해 (M−2)(2(N−1)+(N−1)+(N−2))개의 연산이 필요하고, 모든 dx(m, n)에 대해서는 (M−2)(3N+(N−2))개의 연산이 필요하다. 또한 모든 d45(m, n)과 d135(m, n)에 대해서 각각 (M−2)(2(N−2)+(N−2))개의 연산이 필요하다.

≫고속처리 방법 2≪

방법 1에 비해 N개의 저장장소를 더 사용함으로써 인접한 두 행에서 dx(m, n)계산시 필요한 중간합의 반복적 계산을 제거한다.

마스크가 한 행의 맨 우측에 도달하면 다음 행으로 이동하여 계산과정을 반복하게 된다. 이때 dx(m, n)과 dx(m+1, n)을 비교해 보면 dx(m, n)에서 구한 두 중간합 Ic(m, n+1)과 Ic(m, n−1)이 dx(m+1, n)에서 중복되어 사용되었다. 즉, 한 행에서 구한 중간합 N개를 N개의 저장장소에 저장하였다가 바로 다음 행에서 다시 사용할 수 있다. 따라서 전체화상에 대해서 dx(m, n)들을 계산시 필요한 중간합들은 단 한번씩만 계산하면 되므로 방법 1에서 중간합 계산시 필요한 2N(M−2)개의 덧셈을 N(M−2)개의 덧셈으로 줄일 수 있다.

다음 표 1은 기존의 방법과 제안된 두가지 방법을 비교한 결과이다.

표 1. 전체 연산수의 비교
Table 1. Comparisons of total number of operations.

	총 연 산 수	추가 저장장소 개수	CPU TIME
기존방법	28(M−2)(N−2)	0	1.633sec
방 법 1	(M−2)(14N−19)	9	0.740sec
방법 2	(M−2)(12N−19)+(M−1)N	N+9	0.696sec

Ⅴ. 결 론

Lee가 제안한 부분합의 의미를 확장하여 네 방향의

마스크를 가진 Sobel 연산자의 고속 처리방법을 제안하였다. 식 (2)-(5)와 같이 반복되는 계산과정을 가진 기존의 Sobel 연산자는 각 방향성분에 대해서 각각 7(M-2)(N-2)개의 연산이 필요하게 된다.[2]

근사적으로 계산하기 위해 M과 N이 매우 큰 수이고 서로 같다고 가정하면, 기존의 방법과 방법 1의 연산수 비는 약 1대 2가 되고, 방법 2와는 대략 13대28이 된다. 따라서 제안된 방법은 적은 수의 추가 저장장소를 사용함으로써 50% 이상의 연산시간을 줄일 수가 있다.

100×100 크기의 화상에 대해 CDC Cyber-170 computer system과 FORTRAN V를 사용하여 시뮬레이션한 결과 표 1과 같은 연산시간이 걸렸으며 실제적으로 방법 1과 방법 2가 기존의 방법에 비해 각각 54.7%와 57.4%의 연산시간을 줄일 수 있음을 입증하였다.

參 考 文 獻

[1] A. Rosenfeld and A.C. Kak, *Digital Picture Processing. 2nd ed.*, vol. 2. Academic Press, New York, 1982.

[2] C.C. Lee, "Elimination of redundant operations for a fast sobel operator," *IEEE Trans. Sys., Man, Cybern.*, vol. SMC-13, pp. 242-245, Mar./Apr., 1983.

[3] M.D. Levine, *Vision in Man and Machine*. McGraw-Hill, New York, pp. 181-188, 1985.

[4] P.D. Picton, "Comment on elimination of redundant operations for a fast Sobel operator," *IEEE Trans. Sys., Man, Cybern.*, vol. SMC-14, pp. 560-561, May/June, 1984.