



07 장 영역 처리를 이용한 에지 검출

- 에지 검출의 개요
- 에지 검출기
- 1차 미분을 이용한 에지 검출
- 2차 미분을 이용한 에지 검출

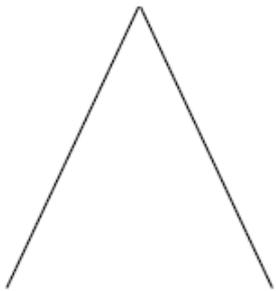
학습목표

- ✓ 에지의 개념을 학습한다.
- ✓ 이동과 차분을 기본으로 한 에지 검출기를 학습한다.
- ✓ 미분과 에지의 관련성을 소개한다.
- ✓ 1차 미분 회선 마스크를 이용한 에지 검출기를 학습한다.
- ✓ 2차 미분 회선 마스크를 이용한 에지 검출기를 학습한다.

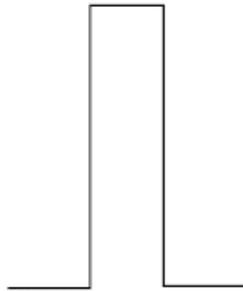
Section 01 에지 검출의 개요

👤 에지(edge)

- 디지털 영상의 밝기가 낮은 값에서 높은 값으로 또는 높은 값에서 낮은 값으로 변하는 지점
- 디지털 영상을 구성하는 객체 간의 경계(= 경계선)
- 디지털 영상의 에지: 물체 식별, 위치/모양/크기 등을 인지하고 방향성을 탐지할 수 있는 정보 제공



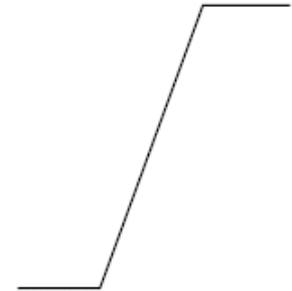
(a) 지붕형



(b) 선형



(c) 계단형



(d) 경사형

[그림 7-1] 다양한 에지 패턴

에지 검출의 개요(계속)

👤 영역처리

- 에지를 검출하기 위한 방법으로 사용됨

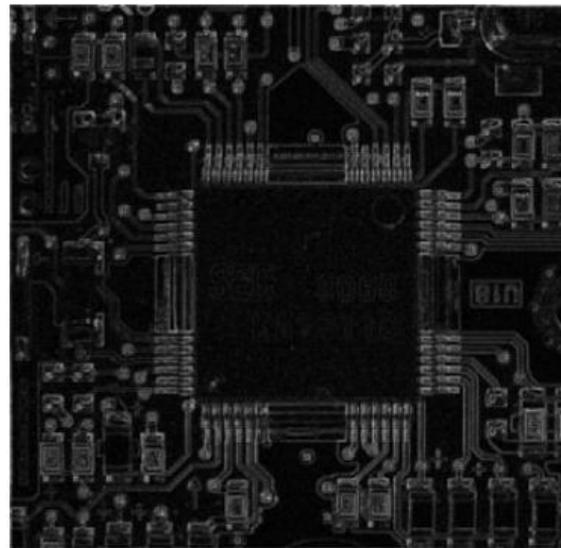
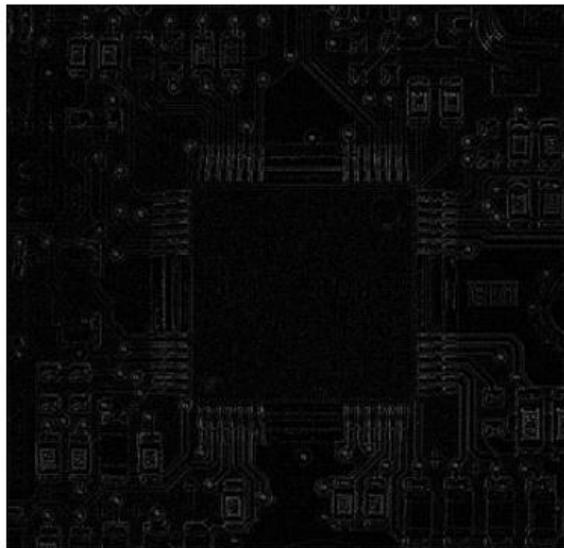
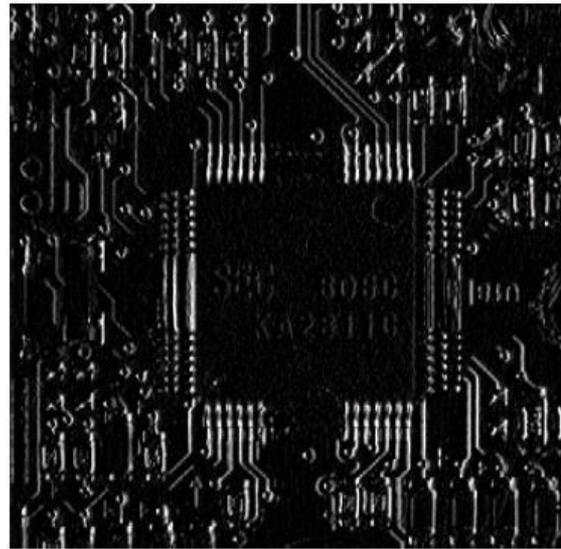
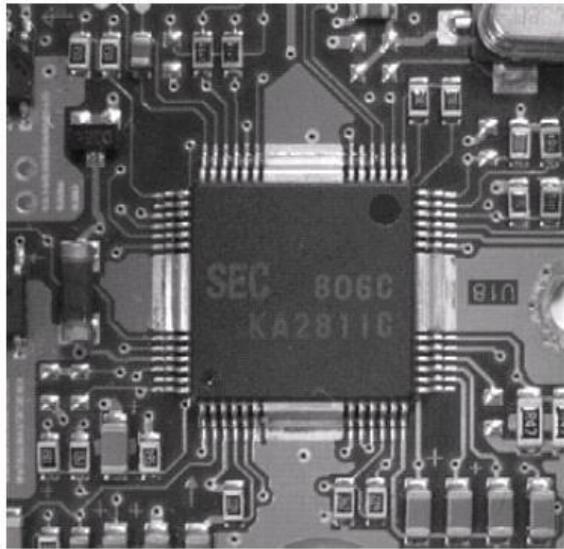
👤 간단한 에지 추출 기법

- 연산 자체가 간단하고 빠름.
- 유사 연산자(Homogeneity Operator)와 차 연산자(Difference Operator)가 있음.
- 이 두 방법으로 얻은 에지를 강화하거나 약화시키는 추가적인 임계 값(Threshold)을 처리하는 방법이 있음.

👤 미분을 이용한 에지 검출 방법

- 에지가 화소의 밝기 변화율에 관여한다는 것
- 1차 미분을 이용한 검출 방법과 2차 미분을 이용한 검출 방법 있음
- 2차 미분을 이용한 검출 방법: 1차 미분으로 얻은 결과에 미분을 한 번 더 추가하여 에지 검출의 성능을 향상시킨 것

에지 검출의 개요(계속)



[그림 7-2] 다양한 1차 미분법으로 검출된 에지 영상

Section 02 에지 검출기

👤 디지털 영상의 에지를 검출하는 가장 쉬운 방법은 화소 간의 차이를 이용하는 것

👤 이동과 차분(Shift And Difference)

- 화소의 위치를 위쪽, 아래쪽, 왼쪽, 오른쪽으로 하나씩 이동시킨 뒤 원래 화소에서 이동 위치의 화소를 빼서 에지를 구함.
- 에지 검출기는 대부분 이 방법을 응용함.

0	0	0
-1	1	0
0	0	0

(a) 수직 에지 검출 마스크

0	-1	0
0	1	0
0	0	0

(b) 수평 에지 검출 마스크

[그림 7-3] 이동과 차분 방법에서의 에지 검출 마스크

[실습하기 7-1] 이동과 차분 처리 프로그램

- ① **ResourceView** 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가

ID	ID_DIFF_OPERATOR_HOR
Caption	이동과 차분(수평처리)

- ② [MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 이동과 차분 처리를 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnDiffOperatorHor
Doc Class	void	OnDiffOperatorHor

- ③ **Doc** 클래스에 다음 프로그램 추가

[실습하기 7-1] 이동과 차분 처리 프로그램

```
void CImageProcessingDoc::OnDiffOperatorHor()
{
    int i, j;
    double DiffHorMask[3][3]
        = {{0., -1., 0.}, {0., 1., 0.}, {0., 0., 0.}};
    // 수평 필터 선택

    m_Re_height = m_height;
    m_Re_width = m_width;
    m_Re_size = m_Re_height * m_Re_width;
    m_OutputImage = new unsigned char [m_Re_size];

    m_tempImage = OnMaskProcess(m_InputImage, DiffHorMask);
    // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);
}
```

[실습하기 7-1] 이동과 차분 처리 프로그램

```
for(i=0 ; i< m_Re_height ; i++){
    for(j=0 ; j< m_Re_width ; j++){
        if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
        if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
    }
}

for(i=0 ; i< m_Re_height ; i++){
    for(j=0 ; j< m_Re_width ; j++){
        m_OutputImage[i* m_Re_width + j]
            = (unsigned char)m_tempImage[i][j];
    }
}
}
```

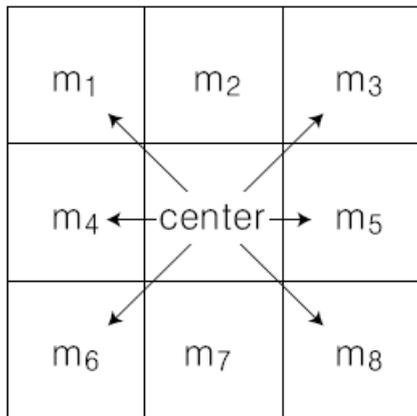
[실습하기 7-1] 이동과 차분 처리 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnDiffOperatorHor()  
{  
    CImageProcessingDoc* pDoc = GetDocument();  
    ASSERT_VALID(pDoc);  
  
    pDoc->OnDiffOperatorHor();  
  
    Invalidate(TRUE);  
}
```

유사 연산자 기법

- 가장 단순한 에지 검출 방법으로 화소를 감산한 값에서 최대값을 결정하여 에지를 검출
- 뿔셈연산이 여러 번 수행되므로 계산 시간이 많이 소요됨



$$\text{New Pixel} = \max(|\text{center} - m_1| \cdots |\text{center} - m_8|)$$

총 8번 계산

예

5	2	1
6	9	3
7	2	6

$$\begin{aligned} \text{New Pixel} &= \max(|9-5|, |9-2|, |9-1|, |9-6|, |9-3|, |9-7|, |9-2|, |9-6|) \\ &= 8 \end{aligned}$$

[그림 7-4] 유사 연산자의 동작 원리

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

- ① ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가

ID	ID_HOMOGEN_OPERATOR
Caption	유사 연산자 기법

- ② [MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 유사 연산자 에지 검출 처리를 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnHomogenOperator
Doc Class	void	OnHomogenOperator
Doc Class	double	DoubleABS(double X)

- ③ Doc 클래스에 다음 프로그램 추가

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

❶ OnHomogenOperator 함수 추가하기

```
void CImageProcessingDoc::OnHomogenOperator ()
{
    int i, j, n, m;
    double max, **tempOutputImage;

    m_Re_height = m_height;
    m_Re_width = m_width;
    m_Re_size = m_Re_height * m_Re_width;
    m_OutputImage = new unsigned char [m_Re_size];

    m_tempImage = Image2DMem(m_height + 2, m_width + 2);
    tempOutputImage = Image2DMem(m_Re_height, m_Re_width);

    for(i=0 ; i<m_height ; i++){
        for(j=0 ; j<m_width ; j++){
            m_tempImage[i+1][j+1] = (double)m_InputImage[i * m_width + j];
        }
    }
}
```

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

❶ OnHomogenOperator 함수 추가하기(계속)

```
for(i=0 ; i<m_height ; i++){
    for(j=0 ; j<m_width ; j++){
        max = 0.0; // 블록이 이동할 때마다 최대값 초기화
        for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
                if(DoubleABS(m_tempImage[i+1][j+1] -
                    m_tempImage[i+n][j+m]) >= max)
                    // 블록의 가운데 값 - 블록의 주변 픽셀 값의 절대 값
                    // 중에서 최대값을 찾는다.

                    max = DoubleABS(m_tempImage[i+1]
                        [j+1] - m_tempImage[i+n][j+m]);
            }
        }
        tempOutputImage[i][j] = max; // 찾은 최대값을 출력 값으로 지정
    }
}
```

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

❶ OnHomogenOperator 함수 추가하기(계속)

```
for(i=0 ; i< m_Re_height ; i++){
    for(j=0 ; j< m_Re_width ; j++){
        if(tempOutputImage[i][j] > 255.)
            tempOutputImage[i][j] = 255.;
        if(tempOutputImage[i][j] < 0.)
            tempOutputImage[i][j] = 0.;
    }
}

for(i=0 ; i< m_Re_height ; i++){
    for(j=0 ; j< m_Re_width ; j++){
        m_OutputImage[i* m_Re_width + j]
            = (unsigned char)tempOutputImage[i][j];
    }
}
}
```

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

② DoubleABS 함수 추가하기

```
double CImageProcessingDoc::DoubleABS(double X)
{ // 실수의 절대 값 연산 함수
    if(X >= 0)
        return X;
    else
        return -X;
}
```

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

④ View 클래스에 다음 프로그램 추가

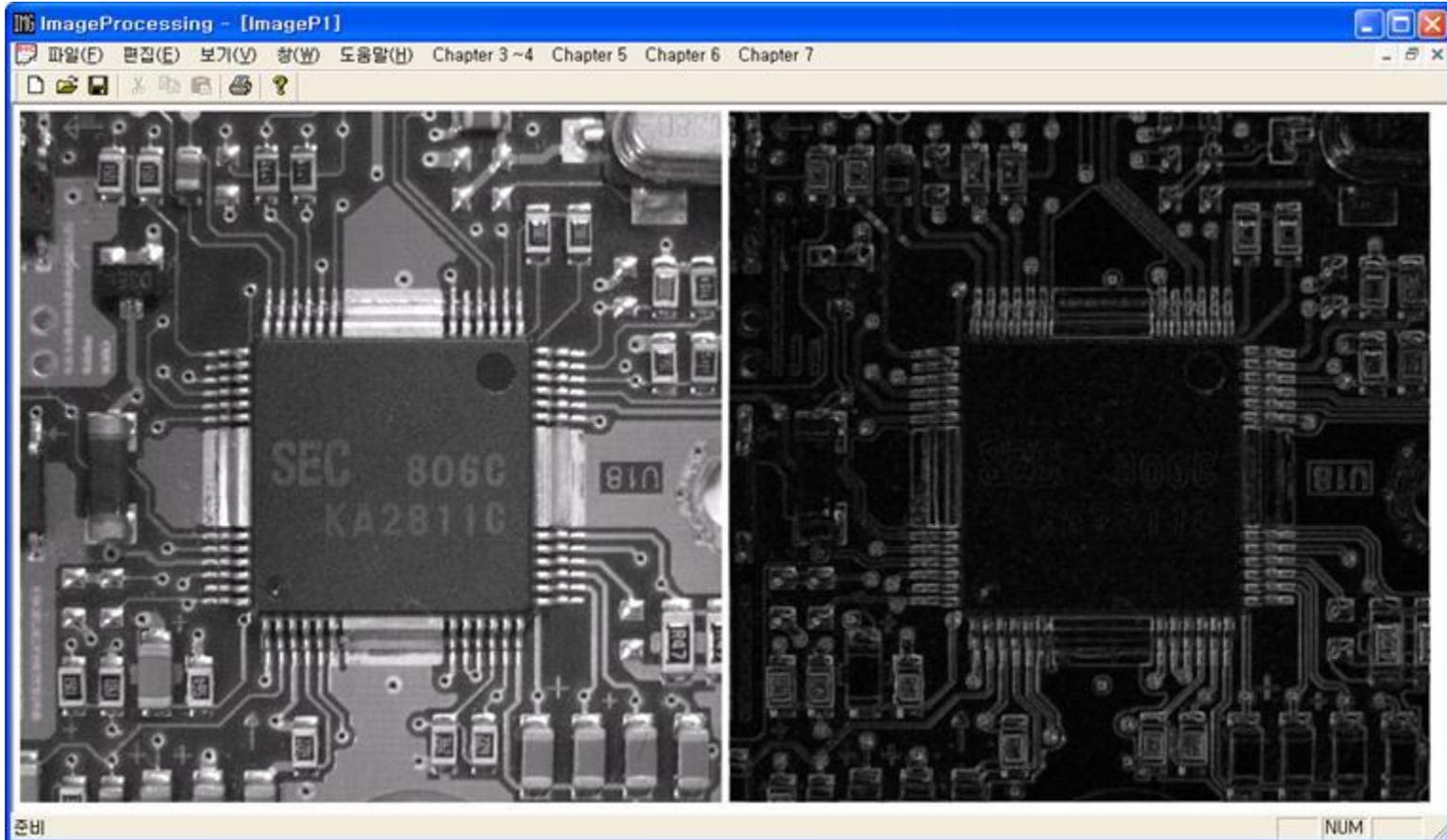
```
void CImageProcessingView::OnHomogenOperator ()
{
    CImageProcessingDoc* pDoc = GetDocument ();
    ASSERT_VALID (pDoc) ;

    pDoc->OnHomogenOperator () ;

    Invalidate (TRUE) ;
}
```

[실습하기 7-2] 유사 연산자 에지 검출 처리 프로그램

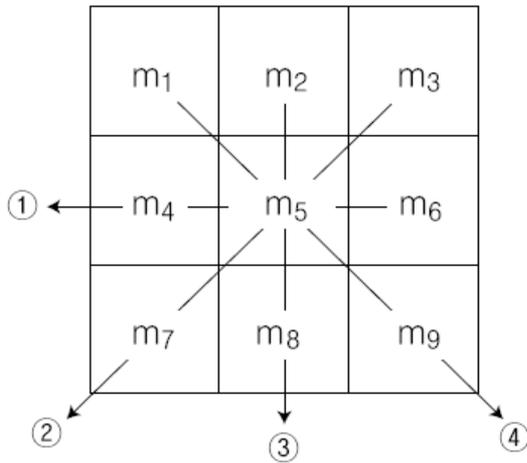
⑤ 프로그램 실행 결과 영상



유사 연산자 기법으로 검출된 에지

차 연산자 기법

- 유사 연산자의 계산 시간이 오래 걸리는 단점을 보완해 주는 방법
- 빨셈 연산이 여덟 번 필요한 유사 연산자와는 달리 화소당 네 번만 사용되어 연산 시간이 빠름.



$$\text{New Pixel} = \max(|m_1 - m_9|, |m_3 - m_7|, |m_2 - m_8|, |m_6 - m_4|)$$

총 4번 계산

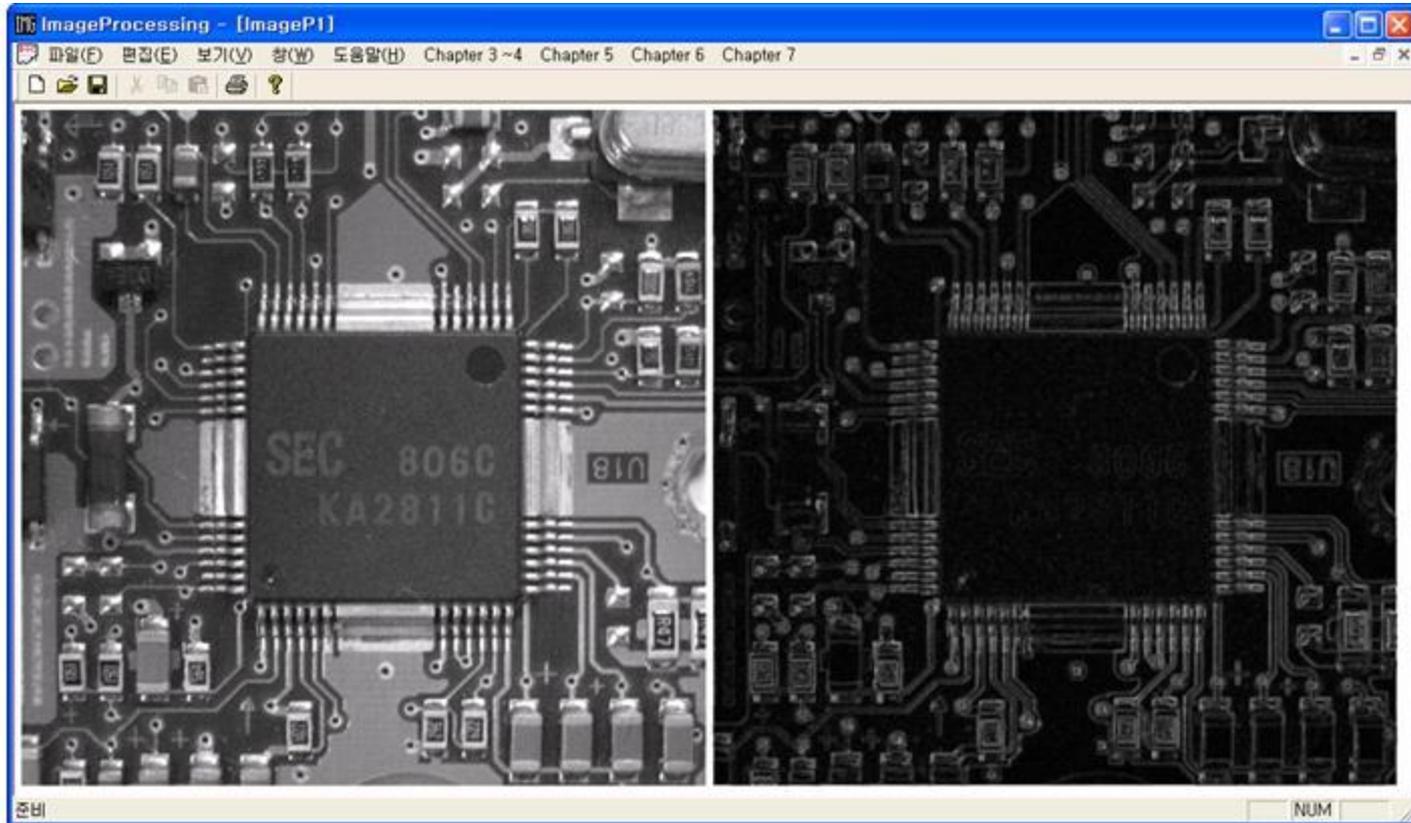
예)

5	2	1
6	9	3
7	2	6

$$\begin{aligned} \text{New Pixel} &= \max(|1 - 9|, |2 - 8|, |3 - 7|, |6 - 4|) \\ &= 8 \end{aligned}$$

[그림 7-5] 차 연산자의 동작 원리

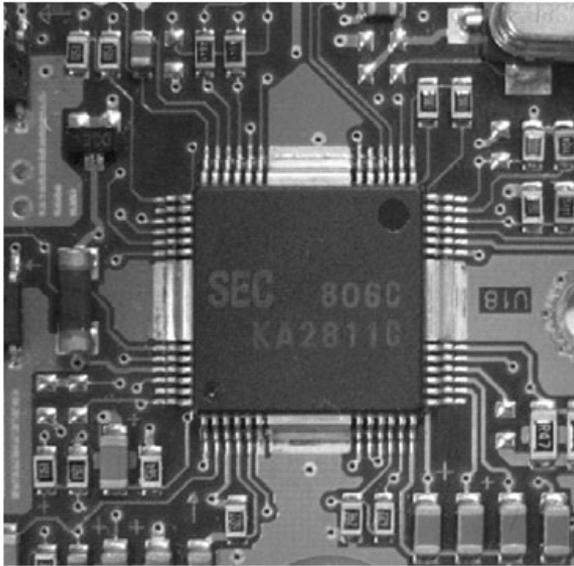
차 연산자 기법(계속)



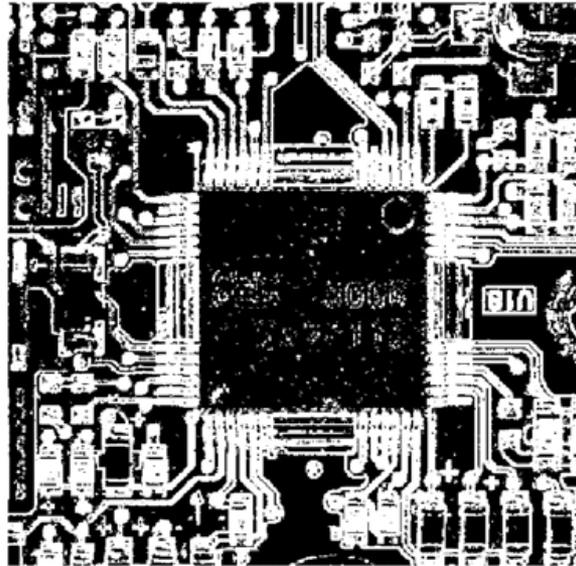
[그림 7-6] 차 연산자 기법으로 검출된 에지

임계 값을 이용한 에지 처리

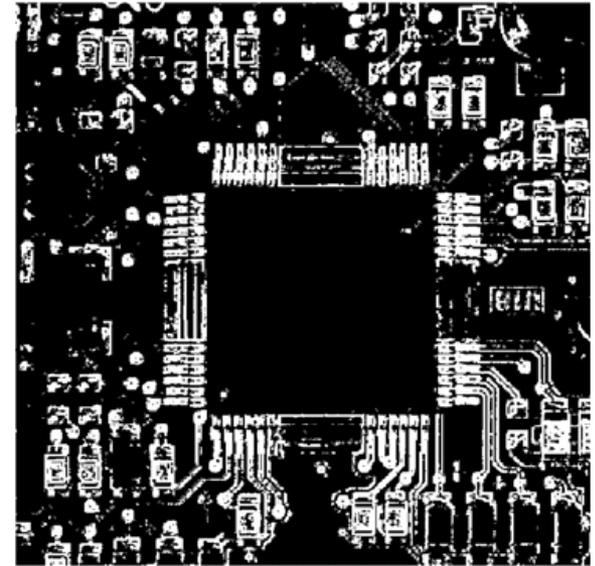
- ▶ 보통 에지 추출기와 함께 사용되어 강한 에지는 강하게, 약한 에지를 약하게 함.
 - 효율적으로 영상처리를 하기 위해서는 검출된 에지를 강조하거나 약화시킬 필요가 있음.



(a) 원본 영상



(b) 단일 임계 값 처리

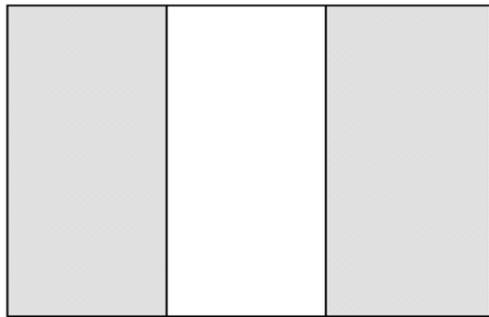


(c) 다중 임계 값 처리

[그림 7-7] 임계 값을 이용해 에지 처리한 영상

Section 03 1차 미분을 이용한 에지 검출

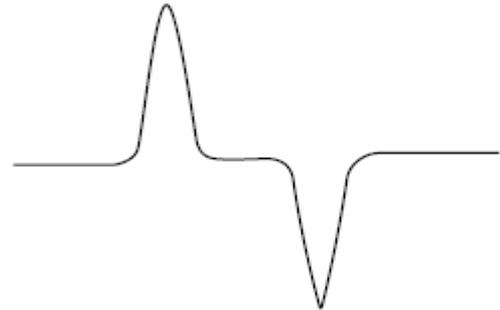
- 디지털 영상의 에지는 화소의 밝기 값이 급격히 변하는 부분이므로, 이 변화 부분을 탐지하는 연산을 이용해 에지 검출



(a) 영상



(b) 명암도 변화



(c) 미분 값 변화

[그림 7-8] 영상의 밝기 변화와 미분 값 변화

- 에지 추출에는 함수의 변화분을 찾는 미분 연산이 이용됨.

1차 미분을 이용한 에지 검출(계속)

좌표 (x, y) 에서 각 방향으로의 편미분

$$\nabla H(x, y) = \begin{bmatrix} H_r(x, y) \\ H_c(x, y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1, y) - f(x, y) \\ f(x, y+1) - f(x, y) \end{bmatrix}$$

- 이웃 데이터와의 차이 값으로 표현되므로, 디지털 영상의 미분은 각 방향의 변화율을 나타냄.
- H_r 는 행 검출기, H_c 는 열 검출기

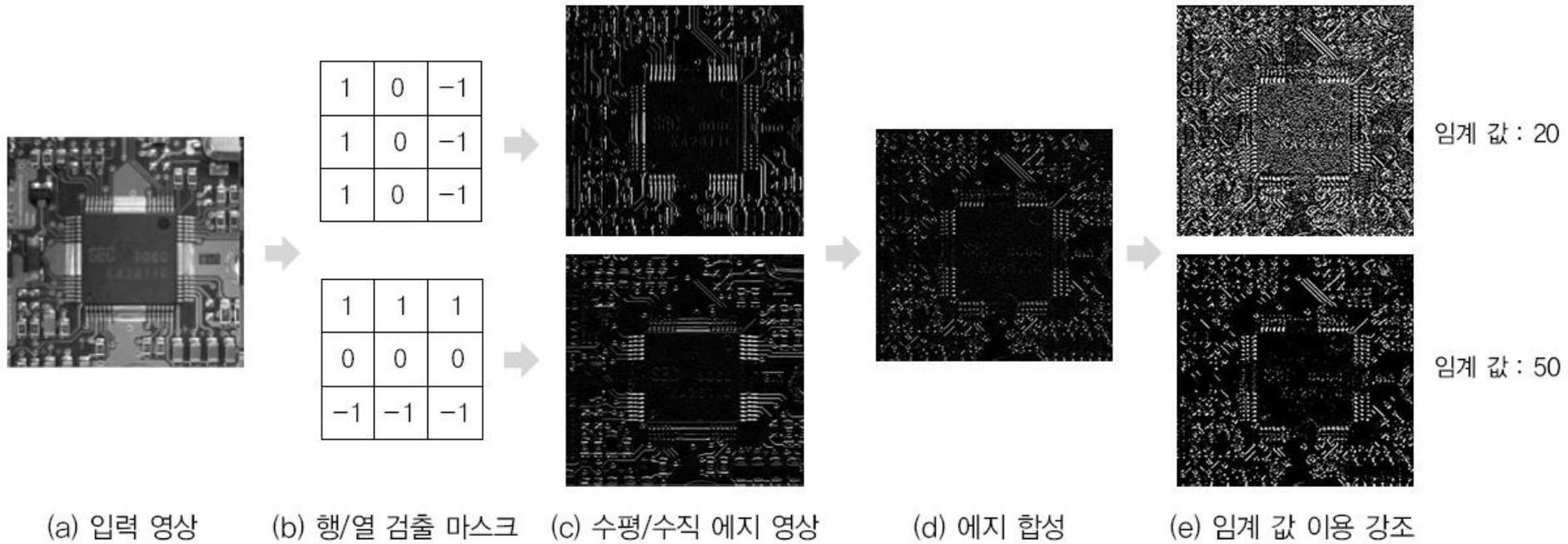
1차 미분을 이용한 에지 검출(계속)

영상의 전체 변화분의 크기 계산

$$H(x, y) \approx |H_r(x, y)| + |H_c(x, y)| = \sqrt{H_r^2(x, y) + H_c^2(x, y)}$$

- 에지 검출을 위한 1차 미분 연산을 영역처리 기법의 회선 처리로 수행하려면 행 검출기 H_r 과 열 검출기 H_c 를 회선 마스크로 생성
- 편미분 식에서 제시된 이웃 데이터와의 차이를 표현하는 회선 마스크를 얻을 수 있음.
- 얻은 1차 미분 회선 마스크는 이동과 차분의 회선 마스크와 형태가 비슷함.
- 마스크의 크기가 클수록 상세한 에지를 검출하기 어렵고, 작으면 잡음에 민감하며, 회선 마스크의 합은 0이 됨

1차 미분을 이용한 에지 검출(계속)



[그림 7-9] 1차 미분 회선 마스크를 이용한 에지 검출 영상

1차 미분 회선 마스크의 종류

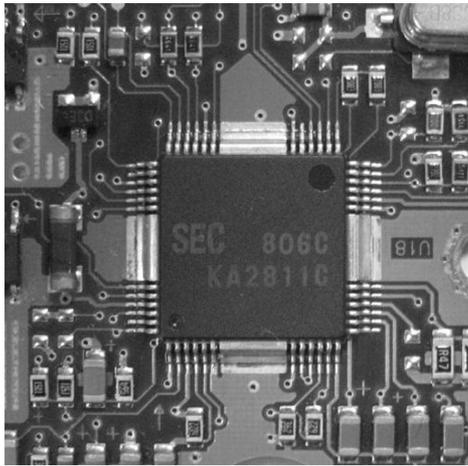
- 종류가 다양함.
- 로버츠(Roberts), 소벨(Sobel), 프리윗(Prewitt) 마스크가 대표적
- 행 검출 마스크와 열 검출 마스크가 있으며, 각 회선 마스크는 고유한 특징이 있음.

	행 검출 마스크	열 검출 마스크
로버츠	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
프리윗	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$
소벨	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

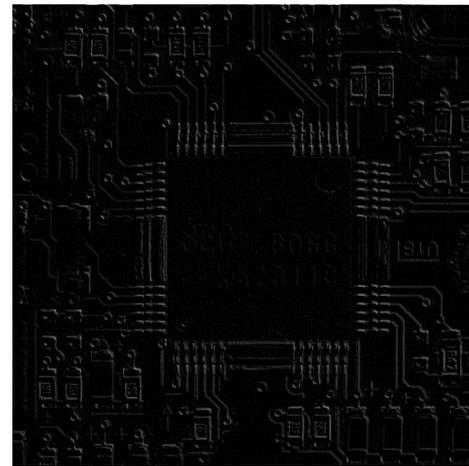
[그림 7-10] 다양한 1차 미분 회선 마스크

로버츠 마스크

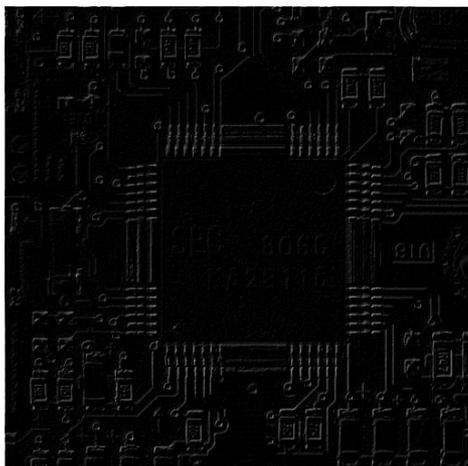
- 장점 : 크기가 작아 매우 빠른 속도로 동작하여 효과적으로 사용 가능.
- 단점 : 돌출된 값을 잘 평균할 수 없으며, 잡음에 민감함.



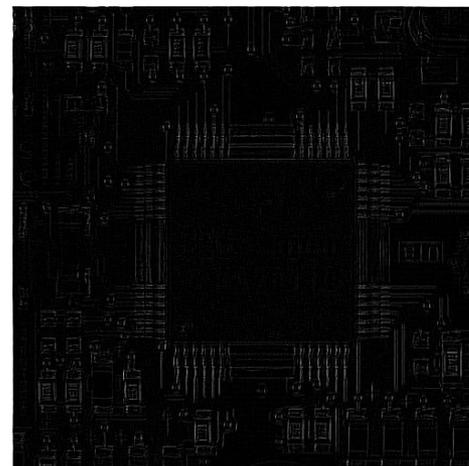
(a) 원본 영상



(b) 수평 방향 에지



(c) 수직 방향 에지

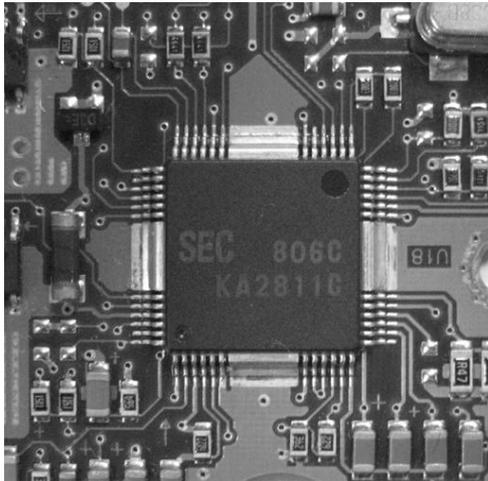


(d) 수평/수직 방향 에지

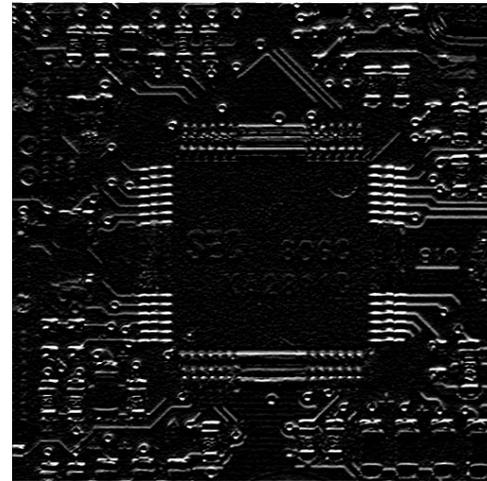
[그림 7-11] 로버츠 마스크를 이용한 에지 검출 영상

프리릿 마스크

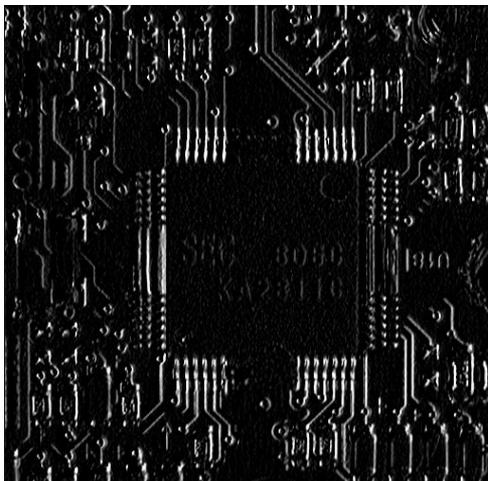
- 장점 : 돌출된 값을 비교적 잘 평균화함.
- 단점 : 대각선보다 수평과 수직에 놓인 에지에 더 민감하게 반응함.



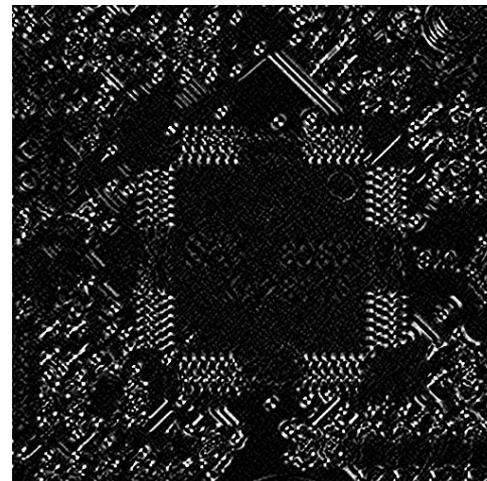
(a) 원본 영상



(b) 수평 방향 에지



(c) 수직 방향 에지

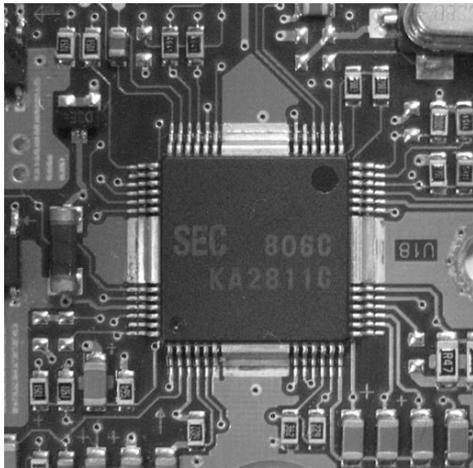


(d) 수평/수직 방향 에지

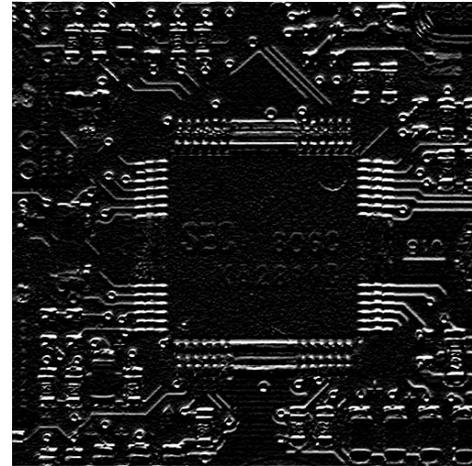
[그림 7-12] 프리릿 마스크를 이용한 에지 검출 영상

소벨 마스크

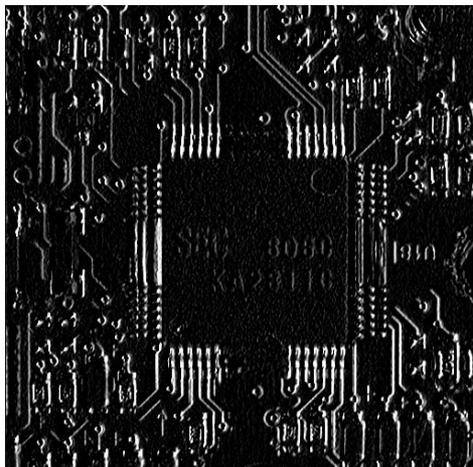
- 장점 : 돌출된 값을 비교적 잘 평균화함.
- 단점 : 대각선 방향에 놓인 에지에 더 민감하게 반응함.



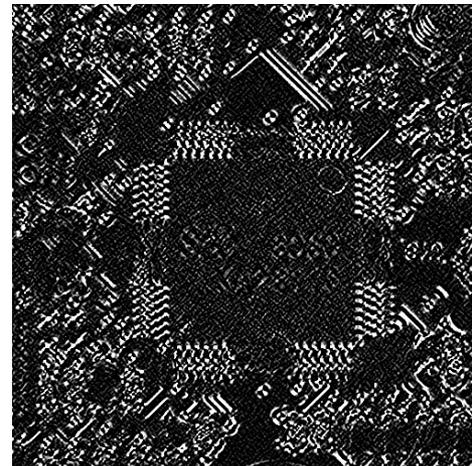
(a) 원본 영상



(b) 수평 방향 에지



(c) 수직 방향 에지



(d) 수평/수직 방향 에지

[그림 7-13] 소벨 마스크를 이용한 에지 검출 영상

Compass Gradient Operator

- 👤 에지를 좀더 정확하게 검출하려고 각기 다른 방향의 마스크 여덟 개를 이용하여 에지를 검출하는 방법
- 👤 여덟 방향으로 수행한 연산의 결과 중 최대값을 최종 출력으로 결정함.
- 👤 단점
 - 마스크의 크기가 작을수록 잡음에 민감하고,
 - 클수록 상세한 에지를 검출할 수 없음.

Section 04 2차 미분을 이용한 에지 검출

- 👤 임의의 이산 함수 $f(x)$ 에서의 1차 미분과 2차 미분 결과

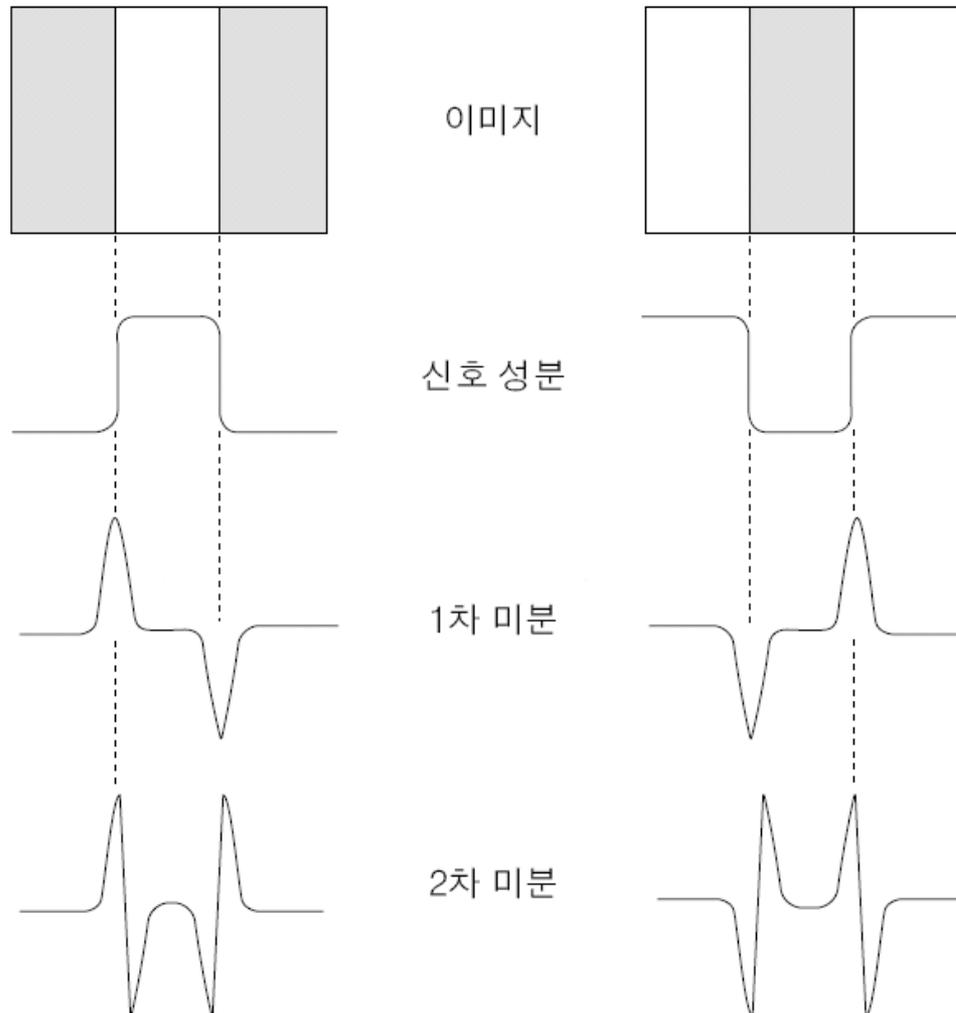
$$f'(x) = \frac{df}{dx} = f(x+1) - f(x)$$

$$\begin{aligned} f''(x) &= \frac{d^2f}{dx^2} = \frac{df(x+1)}{dx} - \frac{df(x)}{dx} = (f(x+1) - f(x)) - (f(x) - f(x-1)) \\ &= f(x+1) - 2f(x) + f(x-1) \end{aligned}$$

- 👤 2차 미분을 이용한 에지 검출기는 미분을 한 번 더 수행하므로, 1차 미분의 단점을 완화시켜 둔감하게 반응하도록 만듦.
 - 1차 미분을 이용한 에지 검출기의 단점: 에지가 있는 영역을 지날 때 민감하게 반응
- 👤 2차 미분을 이용한 에지 검출기의 장점과 단점
 - 장점: 검출된 에지를 끊거나 하지 않고 연결된 폐곡선을 형성함.
 - 단점: 고립된 잡음에 민감하고, 윤곽의 강도만 검출하지 방향은 구하지 못함.

2차 미분을 이용한 에지 검출(계속)

- 2차 미분 연산은 에지 부분에서 부호가 바뀌는 영교차(Zero Crossing)의 특성이 있음.



[그림 7-15] 1차 미분과 2차 미분 검출기의 비교

라플라시안(Laplacian) 연산자

• 대표적인 2차 미분 연산자로, 모든 방향의 에지를 강조함.

• 라플라시안 연산자 공식

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

$$\frac{\partial^2 f(x, y)}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

$$\frac{\partial^2 f(x, y)}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

$$\begin{aligned}\nabla^2 f(x, y) &= f(x+1, y) - 2f(x, y) + f(x-1, y) + f(x, y+1) - 2f(x, y) + f(x, y-1) \\ &= f(x, y+1) + f(x-1, y) + f(x+1, y) + f(x, y-1) - 4f(x, y)\end{aligned}$$

라플라시안(Laplacian) 연산자(계속)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(b)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(c)

[그림 7-16] 라플라시안 회선 마스크

- 1차 미분의 회선 마스크에는 행과 열 방향의 회선 마스크가 있으나,
- 2차 미분의 라플라시안 회선 마스크에는 행과 열 방향이 합쳐져 한 개만 있음.
 - 회선 마스크의 합은 0

[실습하기 7-3] 라플라시안 처리 프로그램

- ① **ResourceView** 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가

ID	ID_LAPLACIAN
Caption	라플라시안

- ② [MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 라플라시안 처리를 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnLaplacian
Doc Class	void	OnLaplacian

- ③ **Doc** 클래스에 다음 프로그램 추가

[실습하기 7-3] 라플라시안 처리 프로그램

```
void CImageProcessingDoc::OnLaplacian()
{
    int i, j;
    double LaplacianMask[3][3] = {{0., 1., 0.}, {1., -4., 1.}, {0., 1., 0.}};

    m_Re_height = m_height;
    m_Re_width = m_width;
    m_Re_size = m_Re_height * m_Re_width;
    m_OutputImage = new unsigned char [m_Re_size];

    m_tempImage = OnMaskProcess(m_InputImage, LaplacianMask);

    // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

    for(i=0 ; i< m_Re_height ; i++){
        for(j=0 ; j< m_Re_width ; j++){
            if(m_tempImage[i][j] > 255.)
                m_tempImage[i][j] = 255.;
            if(m_tempImage[i][j] < 0.)
                m_tempImage[i][j] = 0.;
        }
    }

    for(i=0 ; i< m_Re_height ; i++){
        for(j=0 ; j< m_Re_width ; j++){
            m_OutputImage[i* m_Re_width + j]
                = (unsigned char)m_tempImage[i][j];
        }
    }
}
```

[실습하기 7-3] 라플라시안 처리 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnLaplacian()  
{  
    CImageProcessingDoc* pDoc = GetDocument();  
    ASSERT_VALID(pDoc);  
  
    pDoc->OnLaplacian();  
  
    Invalidate(TRUE);  
}
```

[실습하기 7-3] 라플라시안 처리 프로그램

⑤ 프로그램 실행 결과 영상

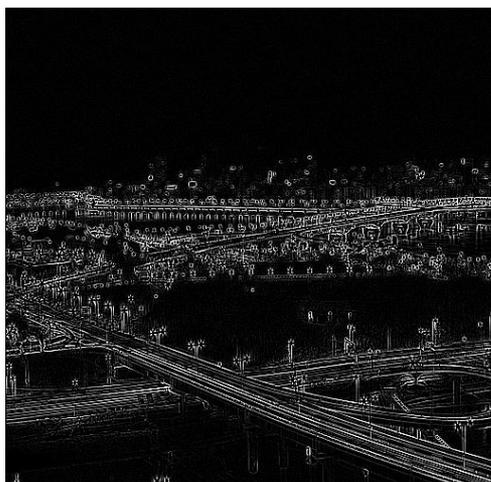
- 잡음 성분에 매우 민감하여 실제보다 더 많은 에지를 검출하므로 에지의 크기를 서로 비교하여 임계 값 이상일 때만 에지로 정의



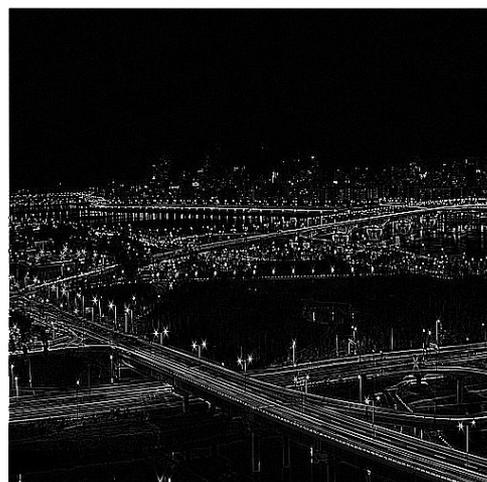
(a) 원본 영상



(b) 마스크 1 적용



(c) 마스크 2 적용



(d) 마스크 3 적용

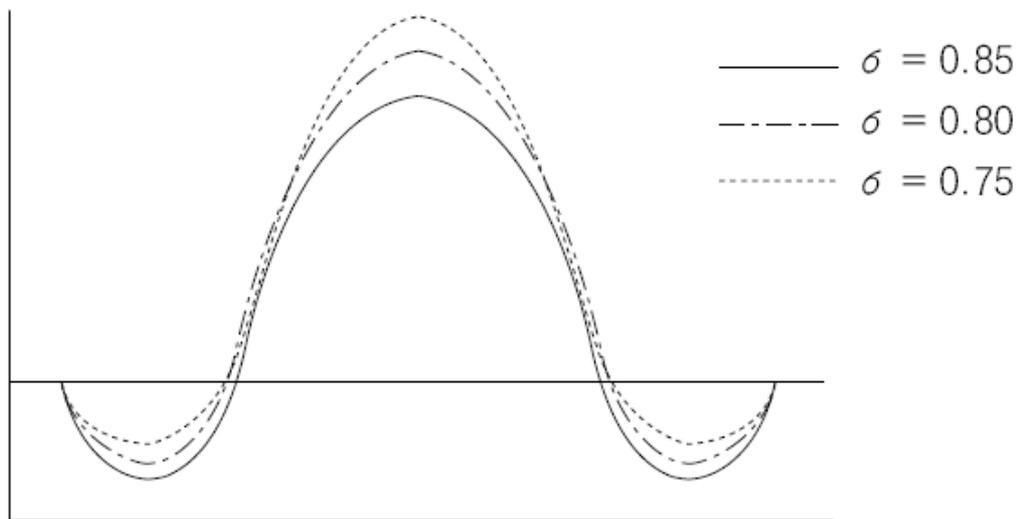
라플라시안 회선 마스크를 사용하여 검출한 에지 영상

LoG(Laplacian of Gaussian) 연산자

- 잡음에 매우 민감한 라플라시안 마스크를 이용한 에지 검출기의 문제점을 해결하기 위해 만듦.
- 가우시안 스무딩을 수행하여 잡음 제거 과정을 거친 뒤 에지를 강조하려고 라플라시안을 이용함.
- LoG 연산자 공식

$$\text{LoG}(x, y) = \frac{1}{\pi\sigma^4} \left[1 - \frac{(x^2 + y^2)}{2\sigma^2} \right] - e^{-\frac{(x^2 + y^2)}{2\sigma^2}}$$

LoG(Laplacian of Gaussian) 연산자(계속)



(a)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

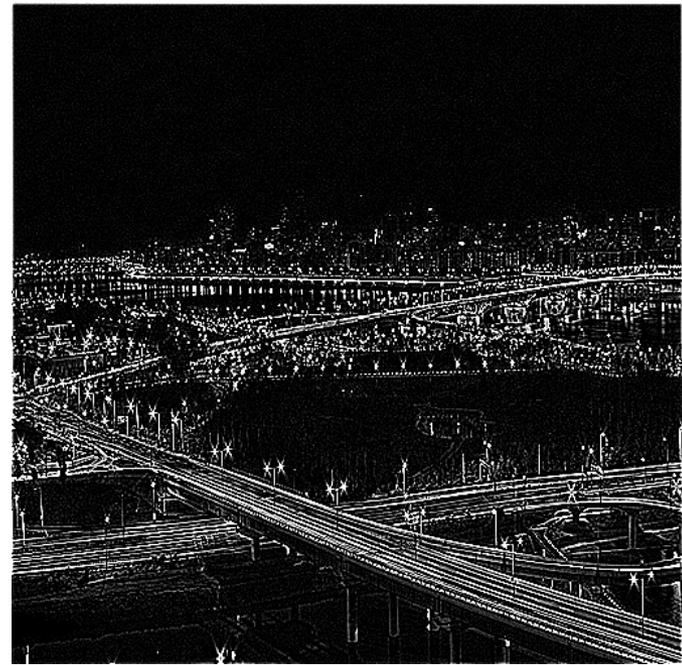
(b)

[그림 7-17] LoG의 멕시코 모자 필터와 회선 마스크

LoG(Laplacian of Gaussian) 연산자(계속)

LoG를 수행하는 두 가지 방법

- 가우시안 스무딩 필터링을 먼저 수행하고 그 결과 값에 라플라시안을 수행하는 방법
- LoG 필터에 해당하는 선형 필터링을 한꺼번에 수행하는 방법



[그림 7-18] LoG로 검출된 에지 영상

DoG(Difference of Gaussians) 연산자

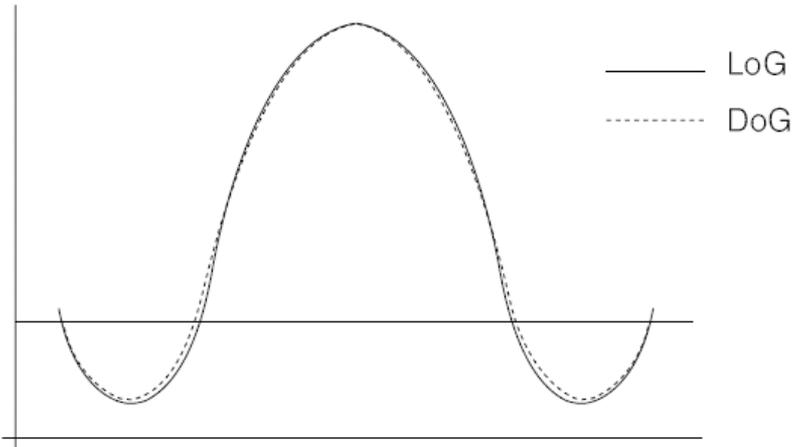
- 계산 시간이 많이 소요되는 LoG 연산자의 단점을 보완하기 위해 등장
- 각 가우시안 연산에 분산 값을 서로 다르게 주어 이 차를 이용해 에지 맵을 구함
- DoG 공식

$$DoG(x, y) = \frac{e^{-\frac{(x^2+y^2)}{2\sigma_1^2}}}{2\pi\sigma_1^2} - \frac{e^{-\frac{(x^2+y^2)}{2\sigma_2^2}}}{2\pi\sigma_2^2}$$

장점

- σ_1 과 σ_2 의 값을 변화시켜 검출할 에지의 넓이를 조절할 수 있음.
- $\sigma_1 / \sigma_2 = 1.6$ 의 값이 있는 비율이 LoG와 가장 비슷한 결과를 나타냄.

DoG(Difference of Gaussians) 연산자(계속)



[그림 7-19] LoG와 DoG 그래프 비교

0	0	-1	-1	-1	0	0
0	-2	-3	-3	-3	-2	0
-1	-3	5	5	5	-3	-1
-1	-3	5	16	5	-3	-1
-1	-3	5	5	5	-3	-1
0	-2	-3	-3	-3	-2	0
0	0	-1	-1	-1	0	0

(a) 7×7 DoG 마스크

0	0	0	-1	-1	-1	0	0	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	-3	-2	-1	-1	-1	-2	-3	0
-1	-3	-1	9	9	9	-1	-3	-1
-1	-3	-1	9	19	9	-1	-3	-1
-1	-3	-1	9	9	9	-1	-3	-1
0	-3	-2	-1	-1	-1	-2	-3	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	0	0	-1	-1	-1	0	0	0

(b) 9×9 DoG 마스크

[그림 7-20] DoG의 회선 마스크

DoG(Difference of Gaussians) 연산자(계속)



[그림 7-21] DoG로 검출된 에지 영상

컬러 영상에서의 에지 검출

👤 RGB 컬러 모델을 사용할 경우

- R, G, B 각각에서 에지 검출을 위한 회선을 수행 후 검출된 에지를 다시 합해 컬러 영상의 에지를 얻음.
- 검출된 각 성분의 에지를 합치는 공식

$$E(x, y) = \sqrt{E_{red}^2(x, y) + E_{green}^2(x, y) + E_{blue}^2(x, y)}$$

👤 HSI 컬러 모델을 사용할 경우

- RGB 컬러 모델을 HSI 컬러 모델로 변환하여 명도값 (I)에서만 회선을 적용한 뒤 RGB 컬러 모델로 변경하여 컬러 영상의 에지를 얻음.

컬러 영상에서의 에지 검출(계속)



[그림 7-22] 컬러 영상에서의 에지 검출

👤 에지

- 디지털 영상의 밝기가 낮은 값에서 높은 값으로 또는 높은 값에서 낮은 값으로 변하는 지점(=경계선)

👤 간단한 에지 추출 기법

- 연산 자체가 간단하고 빠름.
- 유사 연산자와 차 연산자, 임계 값 처리 방법이 대표적

👤 유사 연산자

- 가장 단순한 방법으로, 일련의 화소를 감산한 값에서 최대값을 결정하여 에지를 검출

👤 차 연산자

- 계산 시간이 오래 걸리는 유사 연산자의 단점 해결 위해 제시
- 화소당 뺄셈연산이 네 개만 사용되어서 빠른 연산 수행 가능

👤 임계 값을 이용한 에지 처리

- 보통 에지 추출기와 함께 사용되어 강한 에지는 강하게, 약한 에지는 약화시키는 역할 수행

👤 1차 미분 회선 마스크

- 종류가 다양
- 로버츠, 소벨, 프리윗 마스크가 대표적

👤 Compass Gradient Operator

- 에지를 좀더 정확하게 검출하려고 다른 방향의 마스크 여덟 개를 이용하여 에지를 검출하는 방법

👤 2차 미분 에지 검출기

- 라플라시안, LoG, DoG 등이 대표적

👤 라플라시안 에지 검출기

- 에지 검출 성능이 우수하여 다른 연산자보다 더욱 더 두드러지게 에지 추출
- 에지의 방향은 검출하지 못하고, 잡음 성분에 매우 민감하여 실제보다 많은 에지를 검출.

👤 LoG

- 잡음에 민감한 라플라시안의 문제를 해결하기 위해 만듦.
- 라플라시안을 적용하기 전에 가우시안 스무딩을 수행하여 잡음을 제거한 뒤 에지를 강조하는 데 라플라시안을 이용
- 계산 시간이 많이 소요됨.

👤 LoG

- 계산 시간이 많이 소요되는 LoG의 단점 보완 위해 등장

컬러 영상에서의 에지 검출

- RGB 컬러 모델 사용시: R, G, B 각각에서 에지 검출을 위한 회선을 수행 → 검출된 에지를 다시 합침.
- HSI 컬러 모델 사용시: RGB 모델을 HSI 모델로 변환하여 명도값 (I)에만 회선을 적용 → RGB 모델로 변경해서 컬러 영상의 에지를 구함.



Thank you
