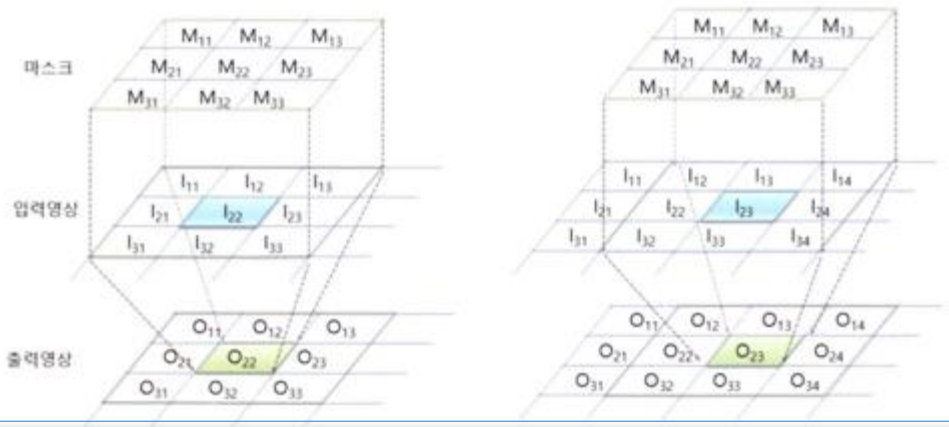
영역 처리

회선(Convolution)

- ❖ 공간 영역의 개념과 회선(Convolution)
  - ✓ 영상 처리에서 영역에 대한 하나의 의미는 두 개의 다른 범위(domain)의 구분으로 공간 영역(spatial domain)이며 다른 하나는 주파수 영역(frequency domain)
  - ✓ 영역에 대한 다른 의미는 영역 기반 처리(area based processing)라는 표현에서 사용하는 영역인데 픽셀이 모인 특정 범위(영역)의 픽셀 배열을 의미
  - ✓ 픽셀 기반 처리가 픽셀 값 각각에 대해 여러 가지 연산을 수행하는 것이라면 영역 기반 처리는 마스크(mask)라 불리는 규정된 영역을 기반으로 연산이 수행되기 때문에 영역 기반 처리를 마스크 기반 처리라고도 함
  - ✓ 마스크 기반 처리는 마스크 내의 픽셀 값과 공간 영역에 있는 입력 영상의 픽셀 값들을 대응되게 곱하여 출력 픽셀 값을 계산하는 것을 의미하는데 이러한 처리를 모든 출력 픽셀 값에 대해 이동하면서 수행하는 것을 회선(convolution)이라고 함
  - ✓ 입력 영상에 곱해 지는 이 마스크는 커널(kernel), 윈도우(window), 필터(filter) 등의 이름으로도 부름



회선(Convolution)

- 커널(Kernel)
  - ✓ 주변 픽셀에 수행되는 연산을 수학적으로는 커널(Kernel)이라고 표현
  - ✓ 커널의 크기는 흐림의 정도를 결정하는데 커널이 클수록 이미지가 더 부드러워짐
  - ✓ 커널은 이미지를 선명하게 만드는 것과 경계선 감지 등에 널리 사용
  - ✓ 이미지 커널 관련 문서

    https://setosa.io/ev/image-kernels/

    https://en.wikipedia.org/wiki/Kernel_(image_processing)

회선(Convolution)

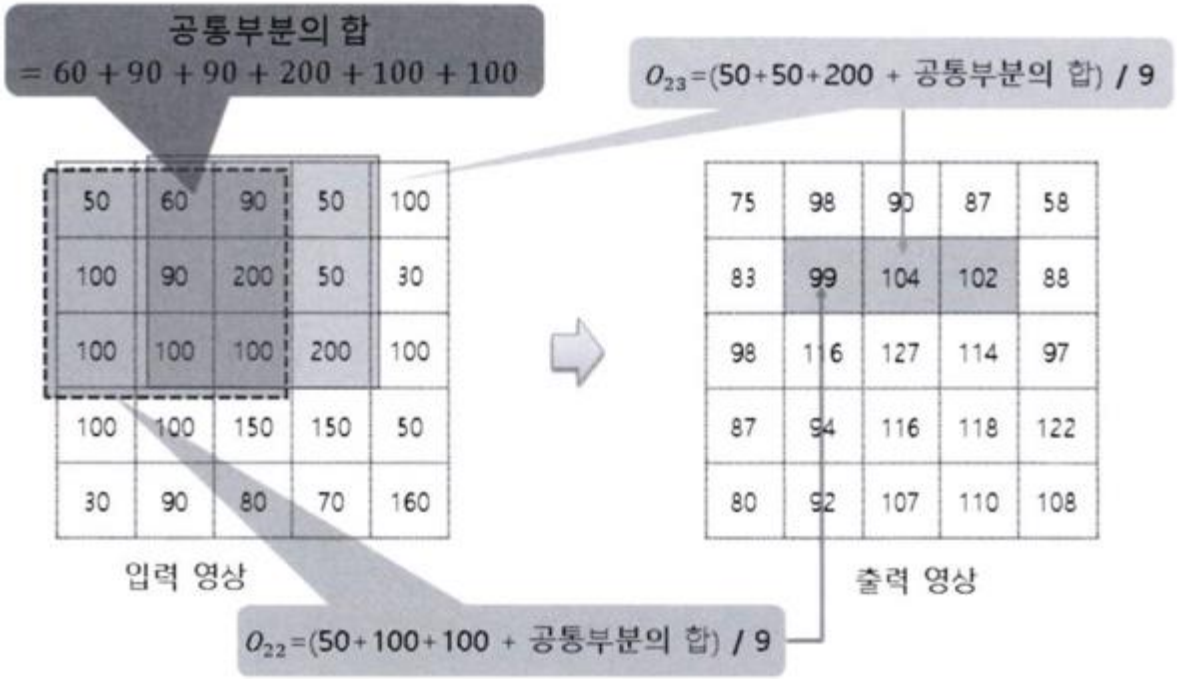
- ❖ 블러링(blurring)
  - ✓ 블러링은 영상에서 픽셀값이 급격하게 변하는 부분들을 감소시켜 점진적으로 변하게 함으로써 영상이 전체적으로 부드러운 느낌이 나게 하는 기술인데 스무딩(smoothing)이라 하기도 함
  - ✓ 급격히 변화하는 것을 점진적으로 변하게 하려면 커널 마스크를 이용해서 회선을 수행하면 됨

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$

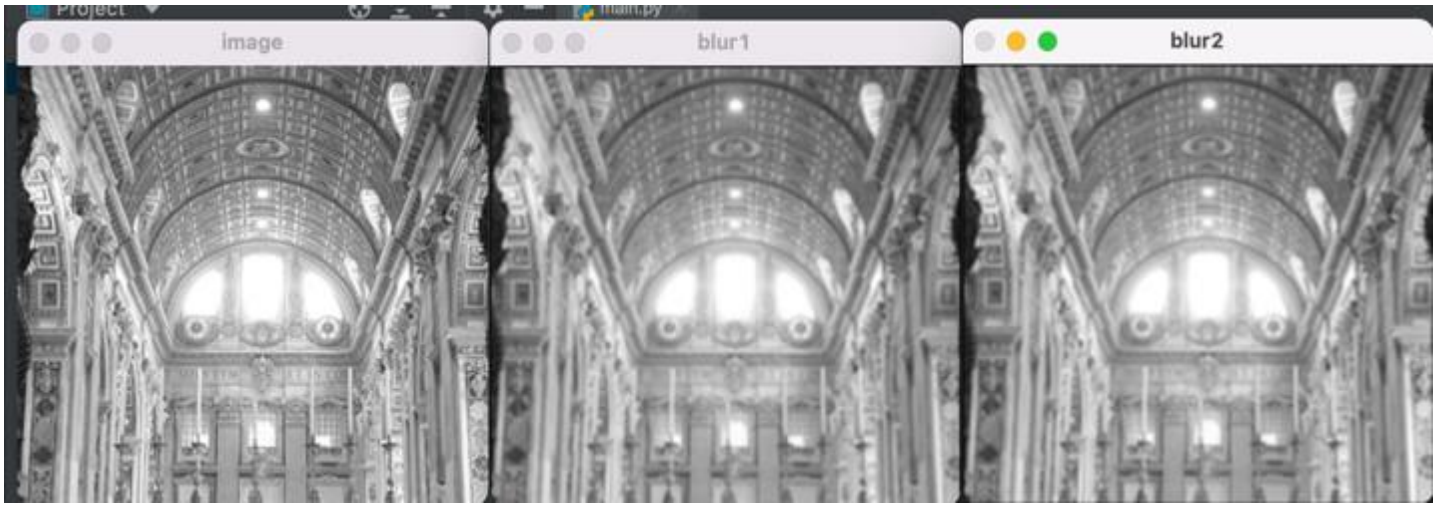
회선(Convolution)

- 블러링
  - ✓ 3 X 3 마스크로 회선



회선(Convolution)

- ❖ 블러링
  - ✓ 회선 이용 블러링



회선(Convolution)

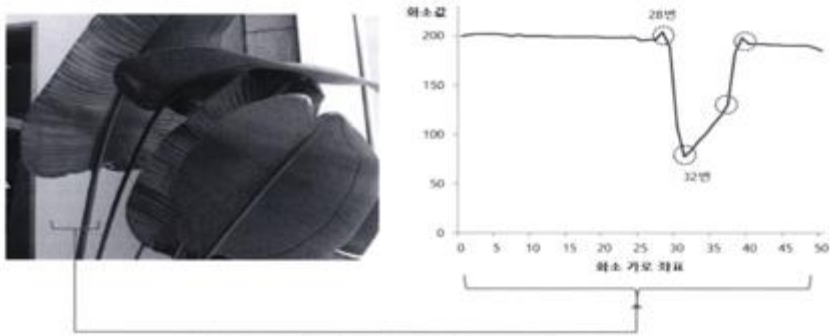
- 샤프닝(Sharpening)
  - ✓ 입력 픽셀에서 이웃 픽셀끼리 차이를 크게 되도록 출력 픽셀를 만들어서 날카로운 느낌이 나게 만드는 것으로 영상의 세세한 부분을 강조할 수 있으며 경계 부분에서 명암 대비가 증가되는 효과를 낼 수 있음
  - ✓ 샤프닝에서는 마스크 원소들의 값 차이가 커지도록 구성하는 것
  - ✓ 입력 영상의 픽셀와 출력 영상의 픽셀가 마스크의 중심 위치에서 대응되는데 이 마스크의 중심 위치의 계수를 중심 계수라 하는데 마스크 중심 계수의 비중이 크면 출력 영상은 입력 영상의 형태를 유지하게 되므로 주변 계수들을 중심 계수와 값의 차이를 크게 만들면 샤프닝이 수행됨
  - ✓ 마스크 원소의 전체 합이 1이 되어야 입력 영상의 밝기가 손실 없이 출력 영상의 밝기로 유지됨
  - ✓ 마스크 원소의 합이 1보다 작으면 출력 영상의 밝기가 입력 영상보다 어두워지며 1보다 크면 입력 영상보다 더 밝아짐
  - ✓ 전체 합이 0이 되면 대부분의 출력 픽셀가 0이 되어 출력 영상이 검은색을 나타내게 되는데 에지 검출에서 주로 사용

회선(Convolution)

❖ 샤프닝(Sharpening)

에지 검출

- 에지 검출
  - ✓ 영상 처리의 관점에서 가장 자리, 모서리 라는 뜻 에서 어떤 물체의 윤곽선 혹은 경계선이라는 의미를 가질 수 있는데 이 윤곽선은 객체에서 크기, 위치, 모양을 인지할 수 있으며 그 방향성을 탐지할 수 있기 때문에 에지 검출은 영상처리에서 아주 중요하며 기본적인 처리 분야로 널리 다루어 짐
  - ✓ 영상 처리에서 에지는 픽셀값이 급격하게 변화하는 부분으로 정의할 수 있는데 픽셀값이 높은 값에서 낮은 값으로 변하거나 혹은 낮은 값에서 높은 값으로 변하는 부분을 에지라 함
  - ✓ 에지 검출(edge detection)이란 에지에 해당하는 픽셀들을 찾는 과정
  - ✓ 에지 검출 방법
    - ❑ 픽셀의 차분을 이용하여 그 차분이 특정 임계값 이상인 곳을 에지로 지정
    - ❑ 마스크를 이용하여 에지를 계산할 수도 있는데 1차 미분 마스크나 2차 미분 마스크를 사용하여 회선을 수행



범위만 한 행의 화소값을 그래프로 표현

에지 검출

- 미분 필터
  - ✓ 영상은 2차원 공간 상에 나열된 픽셀값의 집합이라 할 수 있는데 앞의 그림의 그래프를 보면 왼쪽 영상의 특정 좌표에서 가로 방향(혹은 세로 방향)으로 픽셀값들을 구성했을 때 오른쪽 그래프에서 밝기의 변화를 파악할 수 있는데 여기서 미분(differentiation)이라는 수학 용어가 등장함
  - ✓ 미분은 함수의 순간 변화율을 구하는 계산 과정을 의미하는데 에지가 픽셀의 밝기가 급격히 변하는 부분이기 때문에 함수의 변화율을 취하는 미분 연산을 이용해서 에지를 검출할 수 있음
  - ✓ 영상에서 밝기의 변화율을 검출하는 방법은 밝기에 대한 기울기(gradient)를 계산하는 것으로 현재 픽셀에서 밝기의 기울기를 계산하고 이 기울기의 크기를 구하면 에지가 됨
  - ✓ 디지털 영상은 연속된 데이터가 아닌 이산된 데이터가 나열되어 있기 때문에 엄밀한 의미에서 미분 연산을 할 수 없기 때문에 이산된 데이터인 픽셀에 대한 기울기는 다음의 수식과 같은 방법으로 근사하여 계산

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\delta x} \\ \frac{\partial f(x,y)}{\delta y} \end{bmatrix}$$

$$G_x = \frac{f(x+dx,y)-f(x,y)}{dx} \fallingdotseq f(x+1,y)-f(x,y), \quad dx = 1$$

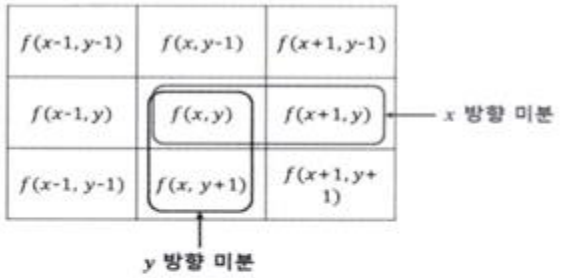
$$G_y = \frac{f(x,y+dy)-f(x,y)}{dy} \fallingdotseq f(x,y+1)-f(x,y), \quad dy = 1$$

$$G[f(x,y)] \fallingdotseq \sqrt{G_x^2+G_y^2} \approx |G_x|+|G_y|$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

에지 검출

- 미분 필터
  - ✓ 2차원 공간상의 한 픽셀에서 수평 방향과 수직 방향으로 각각 미분하는데 이것을 보통 편미분(partial derivative)이라 하고 각 방향의 편미분을 한 픽셀 단위 (dx = 1, dy = 1)의 차분으로 근사
  - ✓ 다음으로 각 방향의 차분을 이용해서 기울기의 크기를 계산하는데 이 크기(magnitude)가 에지의 강도가 되는데 계산 복잡도를 줄이기 위해서 제곱과 제곱근 대신에 절댓값을 사용하기도 하며 또한 역탄젠트(arctan) 함수에 가로 방향과 세로 방향 차분을 적용하면 에지의 방향을 계산할 수도 있음
  - ✓ 1차 미분 공식을 영상에 구현하는 쉬운 방법이 1차 미분 마스크로 회선을 적용하는 것
  - ✓ 3 X 3 마스크에서 입력 픽셀의 위치를 생각해 보면 마스크의 중심 위치의 입력 픽셀가 f(x, y)일 때 주변 픽셀의 위치를 보면 다음 그림과 같음



0	0	0
0	−1	0
0	1	0

(a) y 방향 마스크

0	0	0
0	−1	1
0	0	0

(b) x 방향 마스크

에지 검출

- 미분 필터
  - ✓ 마스크 원소를 (a) 와 같이 f(x, y), f(x, y+1) 위치에 -1 과 1을 구성하여 회선을 수행하면 회선의 내부 계산 수식이 f(x, y+1) - f(x, y) 이 되어서 y 방향 미분인 G_y 와 같은 결과가 되고 오른쪽의 (b)와 같이 f(x, y), f(x+1, y) 위치에 -1과 1을 구성하여 회선을 수행하면 회선 수식이 f(x+1, y) - f(x, y)이 되어서 t 방향 미분인 G_x가 적용됨

에지 검출

- 1차 미분 마스크
  - ✓ 회선의 수식을 이용해서 픽셀간 차분을 계산할 수 있도록 마스크의 원소를 구성하면 1차 미분 마스크가 되는데 이 마스크를 적용해서 입력 영상에 회선을 수행하면 에지 검출이 가능하고 이때 마스크 계수의 합은 0이 되어야 함
  - ✓ 이 외에도 다양한 1차 미분 마스크가 있으며 대표적으로 로버츠(Roberts), 프리윗(Prewitt), 소벨(Sobel) 등이 있음
  - ✓ Roberts Mask
    - ❑ 로버트 마스크는 대각선 방향으로 1 과 -1을 배치하여 구성
    - ❑ 나머지 원소의 값이 모두 0이어서 다른 1차 미분 마스크에 비해서 계산이 단순
    - ❑ 한 번만 차분을 계산하기 때문에 차분의 크기가 작고 이로 인해서 경계가 확실한 에지만을 추출하며 잡음에 매우 민감

$G_x =$

−1	0	0
0	1	0
0	0	0

대각방향 마스크 1

$G_y =$

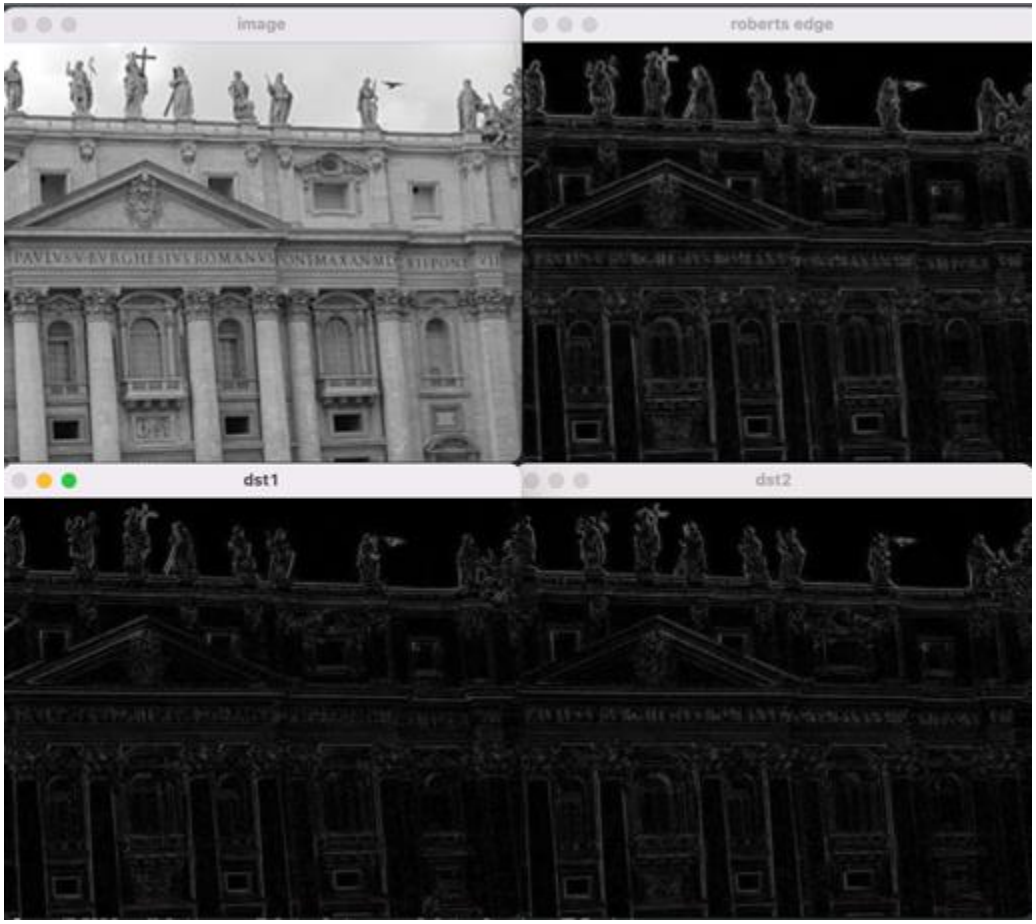
0	0	−1
0	1	0
0	0	0

대각방향 마스크 2

$$O(i, j) = \sqrt{G_x^2 + G_y^2}$$

에지 검출

- 1차 미분 마스크
  - ✓ Roberts Mask



에지 검출

- 1차 미분 마스크
  - ✓ Prewitt Mask
    - 프리윗 마스크는 로버츠 마스크의 단점을 보완하기 위해 고안
    - 그림의 수직 마스크를 보면 원소의 배치가 수직 방향으로 구성되어서 수직 마스크라고 하며 결과 영상에서 에지의 방향도 수직으로 나타남
    - 중심 계수의 앞과 뒤 픽셀로 x 방향 차분을 3번 계산
    - 수평 마스크는 중심 계수에서 위와 아래의 픽셀로 y 방향의 차분을 3번 계산
    - 수직 마스크의 회선 결과와 수평 마스크의 회선 결과에 대해서 크기(magnitude)로 영상(에지 강도)을 생성
    - 세 번의 차분을 합하기 때문에 로버츠 연산자에 비해 에지의 강도가 강하며 수직과 수평 에지를 동등하게 찾는 데 효과적

$G_x =$

−1	0	1
−1	0	1
−1	0	1

수직 마스크

$G_y =$

−1	−1	−1
0	0	0
1	1	1

수평 마스크

$$O(i, j) = \sqrt{G_x^2 + G_y^2}$$

에지 검출

- 1차 미분 마스크
  - ✓ Prewitt Mask

에지 검출

- 1차 미분 마스크
  - ✓ Sobel Mask
    - ❑ 소벨 마스크는 에지 추출을 위한 가장 대표적인 1차 미분 연산자
    - ❑ 마스크의 구성은 프리윗 마스크와 유사하지만 그림과 같이 중심 계수의 차분에 대한 비중을 2배로 키운 것이 특징
    - ❑ 소벨 마스크도 수직 마스크의 회선 결과와 수평 마스크의 회선 결과의 크기로 구성되는데 수직, 수평 방향의 에지도 잘 추출하며 특히 중심 계수의 차분 비중을 높였기 때문에 대각선 방향의 에지도 잘 검출

$G_x =$

−1	0	1
−2	0	2
−1	0	1

수직마스크

$G_y =$

1	−2	1
0	0	0
1	2	1

수평마스크

$$O(i, j) = \sqrt{G_x^2 + G_y^2}$$

에지 검출

- 1차 미분 마스크
  - ✓ Sobel Mask

    cv2.Sobel(src, ddepth. dx, dy[, dst[, ksize[. scale[, delta[, borderType]]]]]) -> dst

    - ❑ 입력 src의 Sobel 연산자를 확장한 dx, dy에 따라 1, 2, 3차 미분을 dst

$$dst(x,y) = \frac{\partial^{dx+dy} src(x,y)}{\partial x^{dx} \partial y^{dy}}$$

      - dst는 src와 같은 채널 같은 크기
      - ddepth는 출력 dst의 픽셀 구조이며 ddepth = - 1이면 src 와 같은 픽셀 구조
      - src가 cv2_CV_8U이면 dst의 ddepth는 -1, cv2.CV_16S, cvaCV_32F, cv2_CV_64F가 가능
      - dx는 x축 미분 차수, dy는 y축 미분 차수
      - ksize는 Sobel 윈도우 필터의 크기로 1, 3, 5, 7인데 예를 들어 ksize = 3이면 3 X 3 윈도우 필터(커널)이고 dx = 1, dy = 0이면, x-축 편미분 g_x dx = 0, dy = 1이면 y-축 편미분 g_y 필터를 적용
      - ksize = 1이면 dx, dy에 따라 3 X 1 또는 1 X 3 커널이 적용되고 x축, y축으로의 1차 또는 2차 미분 만을 위해 사용

에지 검출

- 1차 미분 마스크
  - ✓ Sobel Mask
    - #1 : 입력 영상 sre의 그래디언트 gx, gy를 cv2.Sobel() 함수를 계산
    - #2 : gx의 절대값의 제곱근을 계산하고, cv2mormdizeO로 최소값을 0, 죄대값을 255로 dstX에 정규화
    - #3 : gy의 절대값의 제곱근을 계산하고, cv2.normafce()로 최소값을 0, 최대값을 255로 dstY에 정규화
    - #4 : cv2.magnitude(gx, gy)로 그래디언트의 크기를 mag에 계산한다. mag의 값이 큰 픽셀가 에지
    - cv2momialize()로 최소값을 0, 최대값을 255로 dstM에 정규화
    - 첫번째 그림은 입력 영상이고 두번째 그림은 gx를 정규화한 dstX로 x-방향으로 변화가 있는 세로 에지에서 높은 값(흰색)을 갖으며 세번째 그림은 gy를 정규화한 dstY로 y-방향으로 변화가 있는 가로 에지에서 높은 값(흰색)을 갖으며 마지막 그림은 mag 를 정규화한 dstM로 사각형의 테두리 에지에서 높은 값을 갖음
    - cv2.normalize() 대신 cv2.threshold()를 사용하면 에지와 배경을 갖는 이진 영상을 얻을 수 있음

에지 검출

- 2차 미분 마스크
  - ✓ 1차 미분 연산자는 밝기가 급격하게 변화하는 영역뿐 아니라 점진적으로 변화하는 부분까지 민감하게 에지를 검출하여 너무 많은 에지가 나타날 수 있는데 이를 보완하는 방법으로 1차 미분에서 한 번 더 미분하는 방법인 2차 미분(second order derivative) 연산이 있음
  - ✓ 2차 미분 연산자는 변화하는 영역의 중심에 위치한 에지만을 검출하며 밝기가 점진적으로 변화되는 영역에 대해서는 반응을 보이지 않음
  - ✓ 대표적인 방법으로는 라플라시안(Laplacian), LoG(Laplacian of Gaussian), DoG(Difference of Gaussian) 등 이 있음

에지 검출

- 2차 미분 마스크
  - ✓ Laplacian Mask
    - ❑ 라플라시안은 피에르시몽 라플라스(Pierre-Simon Laplace)라는 프랑스의 수학자 이름을 따서 지은 것
    - ❑ 라플라시안은 함수 f에 대한 그래디언트의 발산으로 정의되며 수식으로 표현하면 다음과 같음

$$\triangle f = \nabla^2 f = \nabla \cdot \nabla f$$

    - ❑ 영상은 2차원 좌표계이기 때문에 2차원 직교좌표계에서 라플라시안의 수식은 다음과 같음

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

에지 검출

❖ 2차 미분 마스크

✓ Laplacian Mask

❑ 각 항을 디지털 영상의 픽셀로 근사하여 1차 미분한 결과에 한 번 더 미분을 수행하면 다음과 같이 정리할 수 있음

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial f(x+1, y)}{\partial x} - \frac{\partial f(x, y)}{\partial x} \\ &= [f(x+1, y) - f(x, y)] - [f(x, y) - f(x-1, y)] \\ &= f(x+1, y) - 2 \cdot f(x, y) + f(x-1, y) \\ \frac{\partial^2 f}{\partial y^2} &= \frac{\partial f(x, y+1)}{\partial y} - \frac{\partial f(x, y)}{\partial y} \\ &= [f(x, y+1) - f(x, y)] - [f(x, y) - f(x, y-1)] \\ &= f(x, y+1) - 2 \cdot f(x, y) + f(x, y-1)\end{aligned}$$

❑ 두 항을 더하면 다음과 같이 라플라시안 마스크의 공식이 완성

$$\nabla^2 f(x, y) = f(x-1, y) + f(x+1, y) + f(x, y-1) + f(, y+1) - 4 \cdot f(x, y)$$

에지 검출

- 2차 미분 마스크
  - ✓ Laplacian Mask
    - ❑ 3 X 3 크기의 마스크를 예로 라플라시안 마스크 공식에 적용하면 중심 계수를 4배로 하고 상하좌우 픽셀를 중심 계수와 반대 부호를 갖게 구성하고 마스크 원소의 전체 합은 0이 되어야 함
    - ❑ 이런 방법으로 그림 (a)와 같이 두 개의 마스크를 구성할 수 있으며 4방향을 가지는 마스크가 되고 경우에 따라서는 그림 (b)와 같이 8방향으로 늘려 보면 모든 방향의 에지를 검출하고자 할 때도 있는데 이 경우에는 중심 계수의 값을 더 크게 하고 8방향의 모든 값을 반대 부호가 되게 하면 됨

0	−1	0
−1	4	−1
0	−1	0

0	1	0
1	−4	1
0	1	0

(a) 4방향 마스크

−1	−1	−1
−1	8	−1
−1	−1	−1

1	1	1
1	−8	1
1	1	1

(b) 8방향 마스크

- ❑ 라플라시안은 중심 계수와 4방향 혹은 8방향의 주변 픽셀와 차분을 합하여 에지를 검출하기 때문에 주변 픽셀에 잡음 성분이 있으면 잡음 성분에 매우 민감하여 실제보다 더 많은 에지를 검줄하는 경향이 있음

에지 검출

- 2차 미분 마스크
  - ✓ Laplacian Mask

    cv2.Laplacian(src, ddepth[, dst[, ksize[, scale[, delta.[, borderType]]]]])-> dst

    - src에 대하여 2차 미분을 이용한 Laplacian을 적용한 후에 scale로 스케일링 하고 delta 값을 더해 ddepth 깊이의 dst에 저장
    - ksize는 필터 크기를 결정하며 에지는 라플라시안 필터링 결과에서 0-교차(zero-crossing)하는 위치를 찾음

  - ❑ 라플라시안 필터링은 2차 미분을 사용하므로 잡음 (noise)에 민감
  - ❑ 잡음을 줄이는 방법은 입력 영상에 가우시안 필터를 사용하여 영상을 부드럽게 하여 잡음을 제거하여 미분 오차를 줄이고 라플라시안 필터링을 사용하는 것

에지 검출

- 2차 미분 마스크
  - ✓ Laplacian Mask
    - ❑ #1 : 입력 영상 src를 부드럽게 하여 미분 오차를 줄이기 위하여 ksize = (7, 7) 크기의 필터를 사용한 가우시안 블러링으로 blur를 생성
    - ❑ #2 : 입력 영상 src에 라플라시안 필터링하여 lap을 생성하고 minVal - -239.0, maxVal = 189.0 인데 lap의 절대값을 dst에 저장하고 범위 [0, 255]로 dst를 정규화
    - ❑ #3 : 가우시안 블러링 영상 blur에 라플라시안 필터링하여 lap을 생성하면 minVal = -35.0, maxVal = 3.0인데 lap2의 절대값을 dst에 저장하고 범위 [0, 255]로 dst2를 정규화

에지 검출

- 2차 미분 마스크
  - ✓ LoG & DoG
    - ❑ 라플라시안은 잡음에 민감한 단점이 있어서 잡음을 먼저 제거하고 라플라시안을 수행한다면 잡음에 강한 에지 검출이 가능
    - ❑ 잡음 제거 방법으로는 미디언 필터링 혹은 최대값/최소값 필터링 등을 수행할 수 있는데 이런 방법들은 비선형 공간 필터링이기 때문에 마스크가 큰 경우 속도 문제가 있음
    - ❑ 수행 속도를 생각한다면, 잡음을 제거해 주는 선형 공간 필터를 선택하여 회선을 하고 그 후에 라플라시안 마스크로 회선하는 방법을 생각할 수 있는데 두 가지 모두 선형 필터링이기 때문에 다음 수식과 같이 하나로 합쳐서 단일 마스크로 계산할 수 있는데 여기서 G_σ는 가우시안 스무딩 마스크이며 *는 회선을 의미

$$\Delta[G_\sigma(x,y)*f(x,y)] = [\Delta G_\sigma(x,y)]*f(x,y) = LoG*f(x,y)$$

    - ❑ 이렇게 구성한 마스크가 LoG(Laplacian of Gaussian)
    - ❑ LoG 마스크를 수식에 따라서 풀면 다음 수식과 같음

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

에지 검출

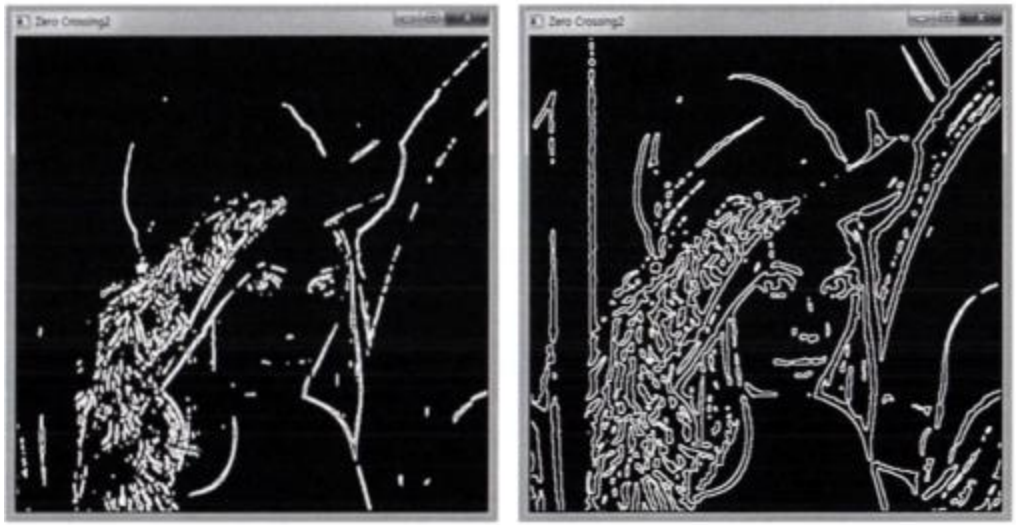
- 2차 미분 마스크
  - ✓ LoG & DoG
    - ❑ 이전 수식으로 마스크의 계수를 구성하고 회선을 수행하면 잡음에 강한 에지를 검출할 수 있지만 수식에 따라 마스크 계수를 생성할 때 값의 범위가 너무 작은 관계로 전체 계수의 합이 0에 가까워지도록 스케일 조정이 필요
    - ❑ LoG는 복잡한 공식에 의해 마스크를 생성해야 하며 그에 따라서 수행 시간도 많이 걸리게 되는데 이런 단점을 보안하여 LoG와 유사한 기능을 하면서 단순한 방법으로 구현하는 알고리즘이 있는데 이것이 바로 DoG(Difference of Gaussian)
    - ❑ DoG는 가우시안 스무딩 필터링의 차를 이용해서 에지를 검출하는 방법

$$DoG(x,y)=\left(\frac{1}{2\pi\sigma_1^2}\cdot e^{\frac{-(x^2+y^2)}{2\sigma_1^2}}\right)-\left(\frac{1}{2\pi\sigma_2^2}\cdot e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}\right) \quad (\text{단, } \sigma_1<\sigma_2)$$

    - ❑ 두 개의 표준편차를 이용해서 가우시안 마스크를 만들고 그 차이가 DoG 마스크가 되고 이 마스크로 회선을 수행하면 에지 검줄이 가능
    - ❑ 여기서 각 표준편차의 값을 조절함으로써 검출할 에지의 넓이를 조절할 수 있음
    - ❑ 쉽게 DoG를 구현하는 방법은 두 개의 표준편차로 가우시안 마스크를 생성하여 회선을 수행하고 그 결과 행렬들의 차분을 계산하는 것

에지 검출

- 2차 미분 마스크
  - ✓ LoG & DoG



에지 검출

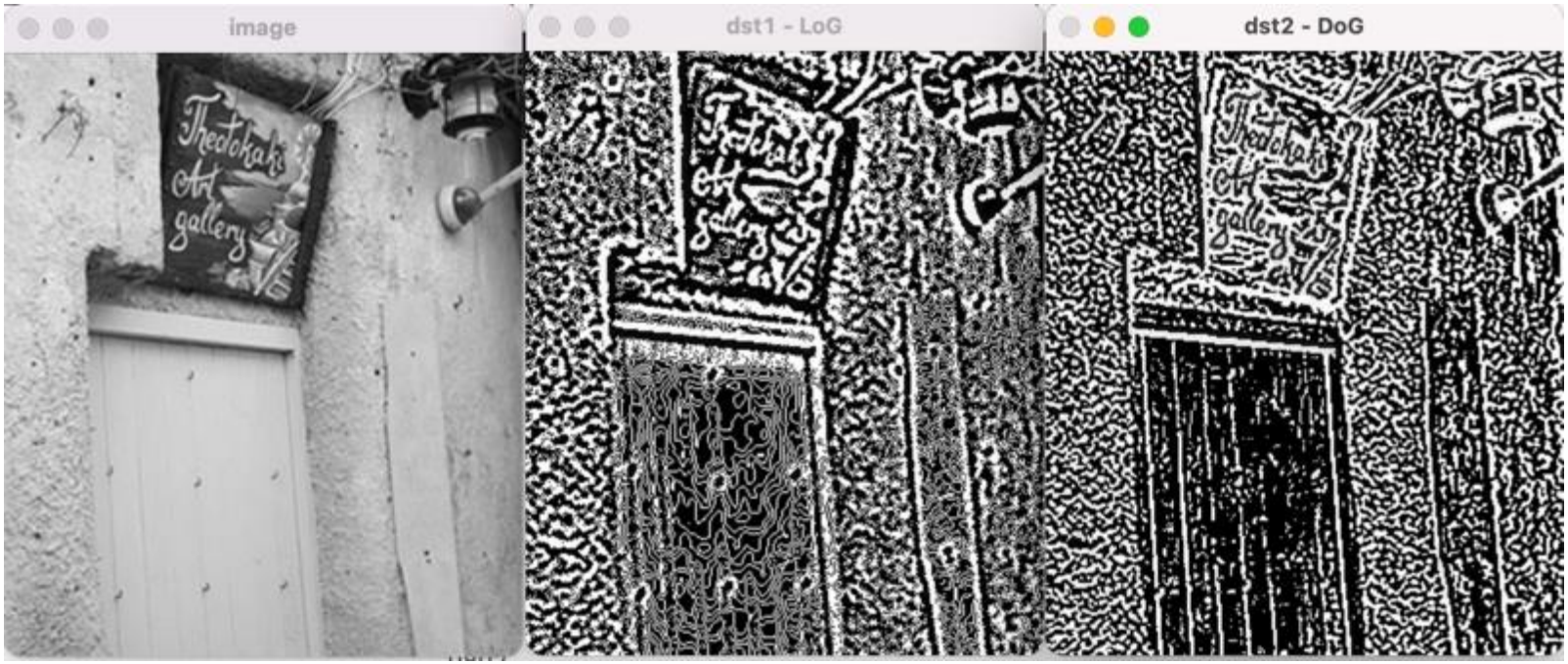
- 2차 미분 마스크
  - ✓ LoG & DoG
    - ❑ #1 : logFilter() 함수는 ksize의 LoG(Laplacian of Gaussian) 필터를 생성하여 반환하는데 라플라시안 필터링은 2차 미분을 사용하여 잡음(noise)에 민감하므로 잡음을 줄이는 방법으로 입력 영상을 가우시안 필터링하여 잡음을 제거한 후에 라플라시안을 적용하는 방법을 사용할 수 있고 가우시안 함수에 대한 2차 미분에 의한 라플라시안을 계산하여 윈도우 필터를 생성하여 필터링할 수도 있는데 이러한 필터링을 LoG(Laplacian of Gaussian)라 함
    - ❑ 윈도우 필터의 크기는 n = 2 X 3 X σ + 1 또는 σ = 0.3(n / 2 - 1) + 0.8로 계산
    - ❑ 에지는 LoG 필터링된 결과에서 0-교차 (zero crossing)하는 위치

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

    - ❑ #2 : logFilter()로 ksize의 가우시안의 라플라시안 필터 kernel을 생성하는데 cv2.filter2D()로 src에 kernel 필터를 적용하여 LoG를 생성
    - ❑ #3 : zeroCrossing2() 함수는 lap[y, x]의 8-이웃 픽셀 neighbors에서 임계값 thresh를 이용하여 value > thresh인 개수 pos, value < -thresh는 neg에 카운트하여, pos > 0 and neg > 0이면 영-교차점으로 판단
    - ❑ 0 대신 thresh를 사용하는 것은 계산에서 실수에 따른 오차 때문

에지 검출

- 2차 미분 마스크
  - ✓ LoG & DoG

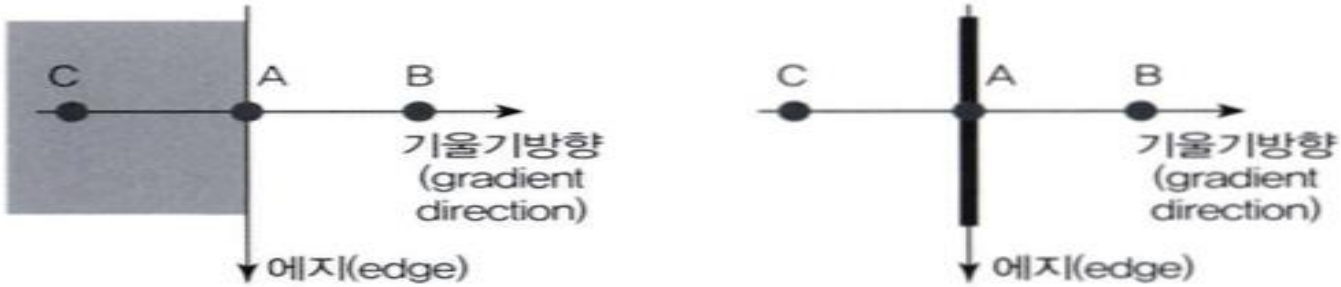


에지 검출

- 2차 미분 마스크
  - ✓ Canny Edge
    - ❑ 영상 내에서 잡음은 다른 부분과 경계를 이루는 경우가 많다 보니 대부분의 에지 검출 방법이 이 잡음들을 에지로 검출하게 되는데 이 문제를 해결하는 방법으로 캐니 에지(Canny edge) 검출 기법이 있음
    - ❑ 캐니 에지 검출 방법은 1986년에 John F. Canny에 의해 개발된 것으로서 여러 단계의 알고리즘으로 구성된 에지 검출 방법
    - ❑ 캐니 에지 알고리즘은 일반적으로 다음의 네 단계의 알고리즘으로 구성
      - 블러링을 통한 노이즈 제거(가우시안 블러링)
      - 픽셀 기울기(gradiant)의 강도와 방향 검출(소벨 마스크)
      - 비최대치 억제(non-maximum suppression)
      - 이력 임계값(hysteresis threshold)으로 에지 결정

에지 검출

- 2차 미분 마스크
  - ✓ Canny Edge
    - ❑ 세부적으로 첫 단계에서 블러링은 5 X 5 크기의 가우시안 필터를 적용해서 수행하는데 블러링은 불필요한 잡음을 제거하기 위해서 수행하는 것이기 때문에 마스크의 크기를 다르게 하든지 혹은 다른 블러링 필터를 적용해도 무관
    - ❑ 다음으로 픽셀 기울기(gradient) 검출에는 가로 방향과 세로 방향의 소벨 마스크로 회선을 적용하고 회선이 완료된 행렬(G_x , G_y) 를 이용해서 픽셀 기울기의 크기(magnitude)와 방향(direction)을 계산하고 기울기의 방향은 4개 방향(0, 45, 90, 135)으로 근사하여 단순화하는데 여기서 그림에서 보는 바와 같이 기울기의 방향과 에지의 방향은 수직을 이루는 것에 유의
    - ❑ 비최대치 억제를 위한 이웃 픽셀 선택



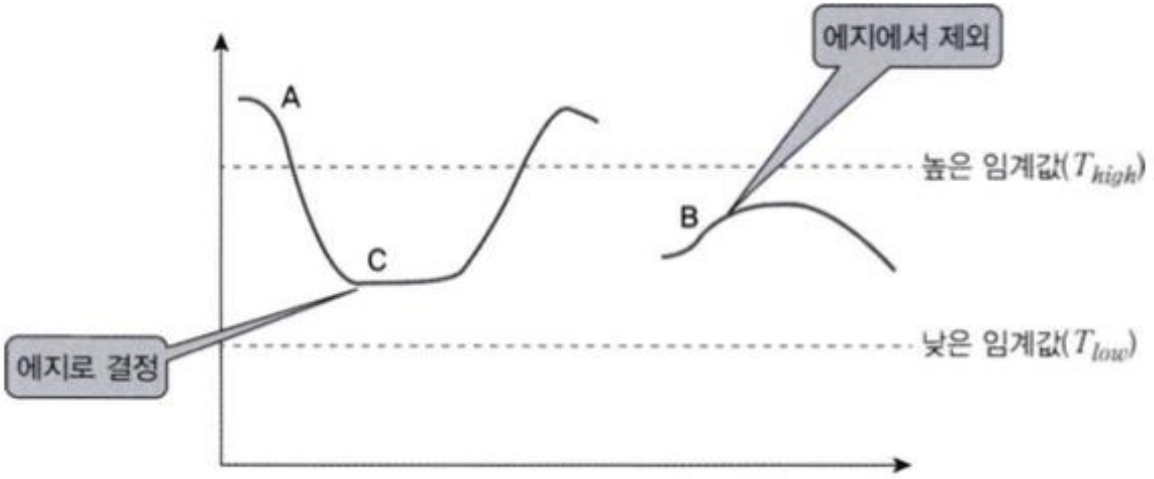
    - ❑ 비최대치 억제(non-maximum suppression)라는 것은 현재 픽셀가 이웃하는 픽셀들보다 크면 에지로 보존하고 그렇지 않으면 에지가 아닌 것으로 간주해서 제거하는 것

에지 검출

- 2차 미분 마스크
  - ✓ Canny Edge
    - ❑ 에지의 방향에 있는 이웃 픽셀는 비교할 필요가 없기 때문에 그림과 같이 기울기의 방향에 있는 두 개의 픽셀를 비교 대상으로 선택하고 현재 픽셀와 선택된 두 픽셀의 에지 강도를 비교하여 최대치가 아니면 억제되고 최대치인 것만 에지로 결정

에지 검출

- 2차 미분 마스크
  - ✓ Canny Edge
    - ❑ 비최대치를 억제하여도 에지가 아닌 것이 에지로 결정된 경우가 많이 존재하는데 잘못된 에지를 제거하는 쉬운 방법 중에 하나가 임계값을 설정하고 에지의 강도가 이 임계값보다 작으면 에지에서 제외하는 것이지만 이 방법은 임계값이 높으면 실제 에지도 제거될 수 있으며 임계값이 낮으면 잘못된 에지를 제거하지 못하는 문제가 생길 수 있음



에지 검출

- 2차 미분 마스크
  - ✓ Canny Edge
    - ❑ 캐니 알고리즘은 잘못된 에지를 제거하고 정확한 에지만을 검출하여 에지가 끊어지는 것을 방지하는 방법으로 이력 임계 처리(hysteresis thresholding) 방법을 사용
    - ❑ 이것은 두 개의 임계값(T_{high} , T_{low})을 사용해서 에지의 이력을 추적하여 에지를 결정하는 방법
    - ❑ 이 방법은 각 픽셀에서 높은 임계값(T_{high})보다 크면 에지 추적을 시작하고 추적을 시작하면 추적하지 않은 이웃 픽셀들을 대상으로 낮은 임계값(T_{low})보다 큰 픽셀를 에지로 결정하는 방식

에지 검출

❖ 에지 검출

cv2.getDerivKernels(dx, dy, ksize[, kx[, ky[, normalize[, ktype]]]])-> kx, ky

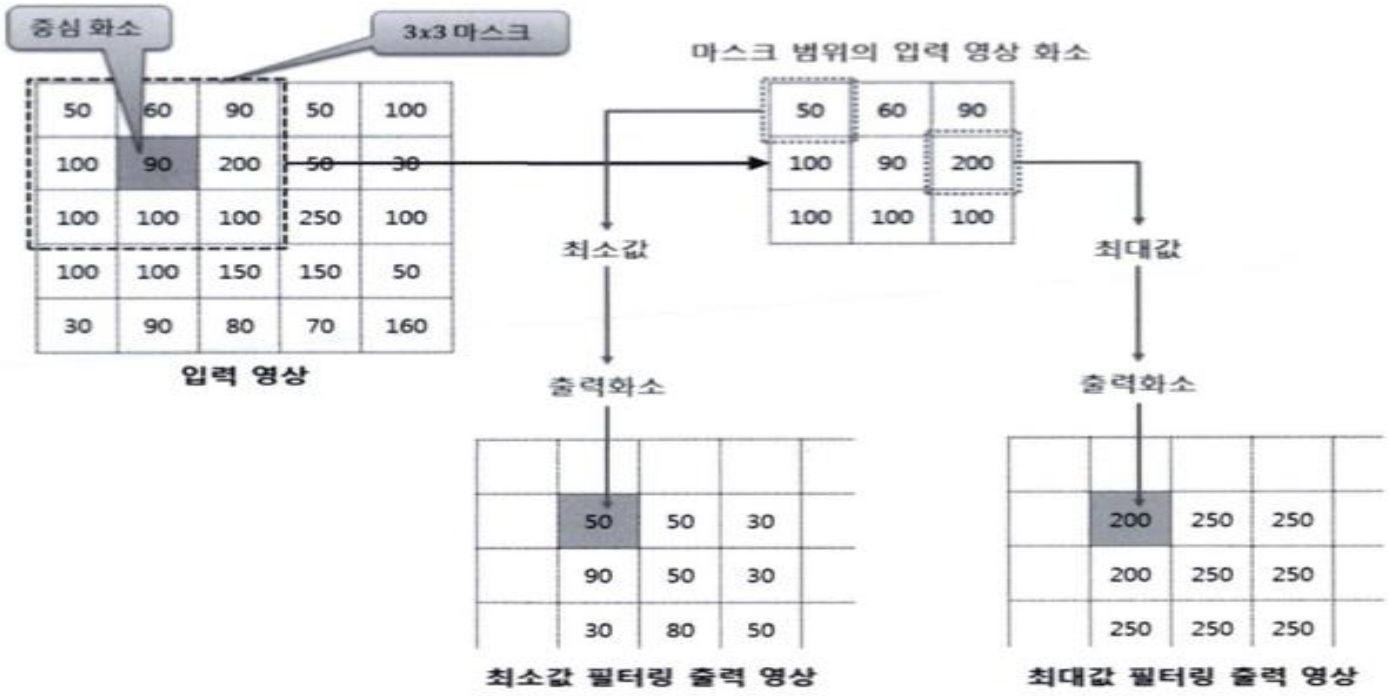
- ❑ cv2.getDerivKernels() 함수는 영상에서 미분을 계산하기 위한 1D 선형 필터를 반환
- ❑ kx, ky는 행과 열의 dx, dy 미분 필터 계수를 위한 출력 행렬
- ❑ normalize = True이면 정규화 수행
- ❑ ktype은 kx, ky의 자료형으로 32비트 실수 또는 64비트 실수이고 ksize는 커널의 크기인데 ksize = 1, 3, 5, 또는 7이면 Sobel 커널이 생성되고 생성된 커널을 사용하여 cv2.sepFilter2D() 함수로 필터링하여 미분을 계산
- ❑ 2D 필터는 ky.dot(kx.T) 행렬 곱셈으로 얻을 수 있음

에지 검출

- 에지 검출
  - ✓ #1 : cv2.getDerivKernels()에서 dx = 1, dy = 0, ksize = 3으로 x-축 방향 미분 선형 필터 kx, ky를 생성하고 sobelX 필터는 dx = 1, dy = 0의 Sobel에서 필터이고 cv2.filter2D()로 src에 sobelX 필터를 적용하여 gx를 생성
  - ✓ #2 : cv2.getDerivKernels()에서dx=0, dy=1, ksize=3으로 y-축 방향 미분 선형 필터 kx, ky를 생성하고 sobelY 필터는 dx = 0, dy = 1의 Sobel에서 필터이며 cv2.filter2D()로 src에 sobelY 필터를 적용하여 gy를 생성
  - ✓ #3 : cv2.magnitude()로 그래디언트(gx, gy)의 크기를 mag에 계산하고 임계값 100을 사용하여 에지 이진 영상 edge를 생성

최대값/최소값 필터링

- 최대값/최소값 필터링
  - ✓ 입력 영상의 중심 픽셀에서 마스크로 씌워진 영역의 입력 픽셀들을 가져와서 계수를 구성하고 그 중에서 최대값 혹은 최소값을 출력 픽셀로 결정하는 방법
  - ✓ 최대값 필터링은 마스크 계수 중에서 최대값을 통과시켜 출력 픽셀이 되고 최소값 필터링은 최소값을 통과시켜 출력 픽셀이 됨



최대값/최소값 필터링

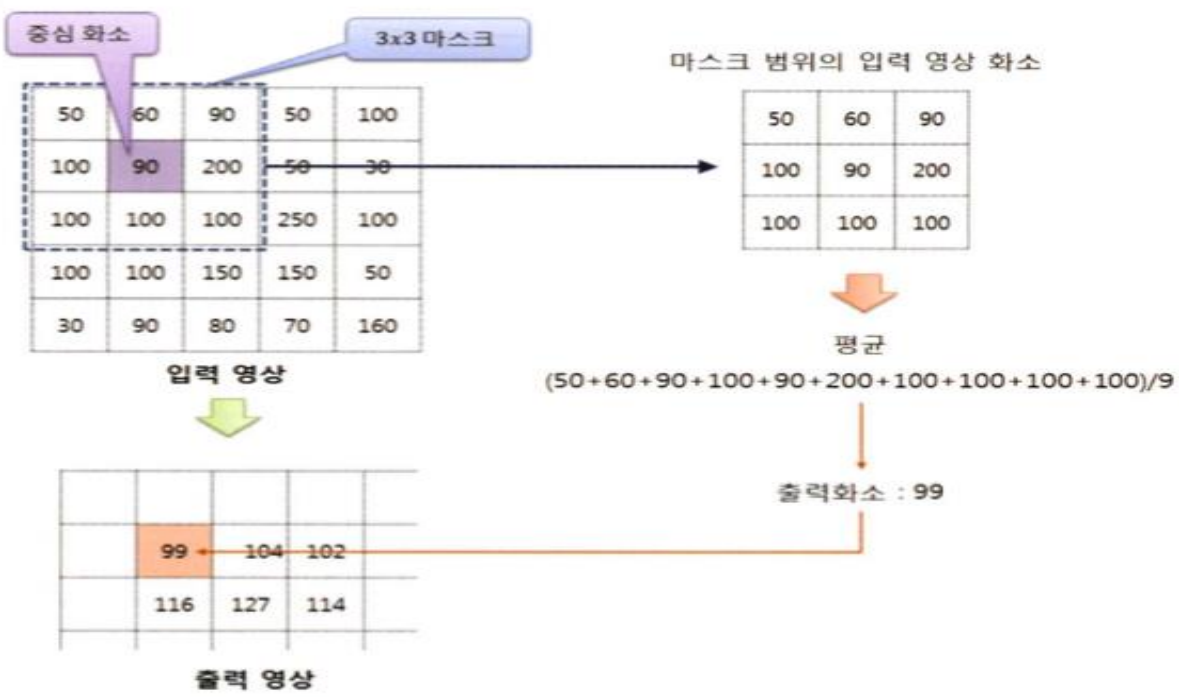
- 최대값/최소값 필터링
  - ✓ 최대값 필터링은 가장 큰 값인 밝은 색들로 출력 픽셀가 구성되기 때문에 돌출되는 어두운 값이 제거되며 전체적으로 밝은 영상이 됨
  - ✓ 최소값 필터링은 가장 작은 값들인 어두운 색들로 출력 픽셀가 구성되기 때문에 돌출되는 밝은 값들이 제거되며 전체적으로 어두운 영상이 됨
  - ✓ 최대값 필터링은 밝은 임펄스 잡음이 강조되며 최소값 필터링은 어두운 임플스 잡음이 강조되므로 높은 대조를 가진 영상에서 특징을 확대시키기 위한 기법으로 이용될 수 있음

최대값/최소값 필터링

❖ 최대값/최소값 필터링

평균값 필터링

- 평균값 필터링
  - ✓ 마스크로 씌워진 영역의 입력 픽셀들을 가져와서 그 픽셀들의 평균을 구하여 출력 픽셀로 지정하는 방법
  - ✓ 마스크 영역의 픽셀값들을 평균하기 때문에 블러링의 효과가 나타남
  - ✓ 회선에서 블러링 마스크를 적용한 것과 결과가 같음



평균값 필터링

- 평균값 필터링

평균값 필터링

- 평균값 필터링
  - ✓ 실행 결과
    - ❑ 모든 결과 영상이 비슷하게 흐려짐
    - ❑ 이전 예제까지는 영상의 상하좌우 끝 부분에 마스크 크기의 절반만큼 검은색으로 출력되었는데 이번에는 정상적으로 처리된 결과를 보이는데 이것은 마스크의 범위가 입력 영상을 벗어나는 픽셀에 대해서 입력 영상의 중심 픽셀를 출력 픽셀로 지정했기 때문
    - ❑ 입력 영상의 상하좌우 끝부분에 있는 픽셀들은 마스크를 씌웠을 때 입력 픽셀가 존재하지 않는 픽셀들이 있음
    - ❑ 3 X 3 마스크일 경우에는 상하좌우로 한 픽셀씩이 해당하며 5 X 5 마스크일 경우에는 상하좌우 두 픽셀씩이 해당되는데 이 경우 배열 참조가 잘못되어 오류가 발생하기 때문에 추가적인 방법을 적용해서 출력 픽셀를 결정해야 함
    - ❑ OpenCV에서도 cv2.filter2D(), cv2.blur(), cv2.boxFilter(), cv2.sepFilter2D()와 같은 필터링을 수행하는 함수들에서 상하좌우 경계 부분의 픽셀값을 결정하는 방법으로 borderType 이라는 옵션 상수를 다음과 같이 정의해 둠

평균값 필터링

❖ 평균값 필터링

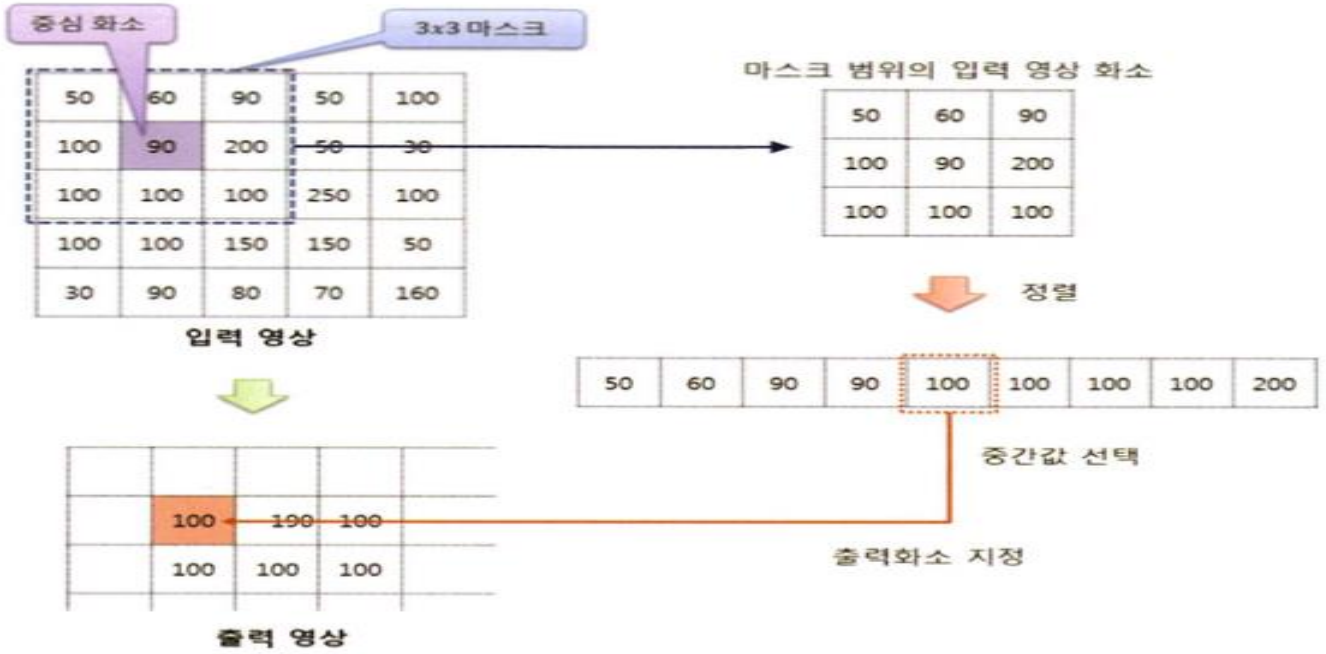
✓ 실행 결과

❑ borderType

옵션 상수	값	설명
cv2.BORDER_CONSTANT	0	특정 상수값으로 대체 70 70 70 20 25 35 45 50 60 70 70 70
cv2.BORDER_REPLICATE	1	계산 가능한 경계의 출력 화소 하나만으로 대체 대체 화소 대체 화소 20 20 20 20 25 35 45 50 60 60 60 60
cv2.BORDER_REFLECT	2	계산 가능한 경계의 출력 화소로부터 대칭되게 한 화소씩 지정 대체 화소 대체 화소 35 25 20 20 25 35 45 50 60 60 50 45
cv2.BORDER_WRAP	3	영상의 왼쪽 끝과 오른쪽 끝이 연결되어 있다고 가정하여 한 화소씩 가져와서 지정 대체 화소 45 50 60 20 25 35 45 50 60 20 25 35 대체 화소

미디언 필터링

- 미디언 필터링
  - ✓ 입력 영상에서 해당 입력 픽셀를 중심으로 마스크를 씌워 마스크 크기 내에 있는 입력 픽셀들을 가져옴
  - ✓ 회선과는 다르게 마스크 계수는 필요하지 않고 마스크의 크기만 필요
  - ✓ 마스크 범 위 내에 있는 픽셀값들을 크기순으로 정렬한 후 정렬된 픽셀값 중에서 중간값을 취하여 출력 픽셀로 지정



미디언 필터링

- 미디언 필터링
  - ✓ 일정 영역에서 다른 픽셀들과 밝기가 심하게 차이가 나는 픽셀들은 임펄스 잡음(impluse noise)이나 소금-후추 잡음(salt & pepper noise)일 가능성이 높음
  - ✓ 미디언 필터링 과정에서 마스크 영역 내의 심하게 차이가 나는 픽셀들은 정렬로 인해서 최하위 값이나 최상위 값이 되는데 미디언 필터링에서 중간값 이외의 다른 나머지 값들은 출력 픽셀로 지정되지 않고 제거되기 때문에 미디언 필터링은 임펄스 잡음이나 소금-후추 잡음을 잘 제거해 주므로 평균값 필터링에 비하면 블러링 현상이 적음
  - ✓ 마스크의 크기가 커지면 잡음 제거 성능은 향상되지만 모든 픽셀를 순회하며 마스크 범위에 대해 정렬 알고리즘을 수행해야 하는 부담 때문에 수행 시간이 기하급수적으로 증가
  - ✓ 미디언 필터링은 보통 명암도 영상에서 효과적으로 수행됨
  - ✓ RGB 컬러 공간에서는 3개(빨강색, 녹색, 파랑색) 채널 간의 상호 의존도가 매우 큰데 예를 들어서 한 채널에서는 특정 픽셀가 주위와 심한 차이를 보여서 제거되어도 다른 채널에서는 같은 픽셀가 주위와 차이가 작아서 제거되지 않는 경우들이 생겨서 RGB 조합이 맞지 않아서 오히려 잡음이 더 많아질 수도 있음

미디언 필터링

❖ 미디언 필터링

cv2.medianBlur(src, ksize[, dst]) -> dst

- ❑ medianBlur() 함수는 src에서 정수 커널 크기 ksize에 의해 ksize X ksize의 필터를 사용하여 미디언(중위수)을 계산하여 dst에 저장
- ❑ src는 1, 3, 4 채널 영상이고 ksize = 3 또는 5일 때는 src의 깊이가 8비트, 16비트 부호 없는 정수, 32비트 실수가 가능하고 ksize가 크면 8비트만 가능
- ❑ dst는 src와 같은 크기이며 같은 자료형
- ❑ 미디언 필터링의 결과값은 원본 src에 있는 값이며 소금이나 후추가 뿌려져 있는 듯한 잡음(salt and pepper noise)의 경우에 평균 필터 또는 가우시안 필터보다 효과적

모폴로지(Morphology)

- 모폴리지 연산
  - ✓ 모폴로지 연산(morphological operation)은 구조 요소(structuring element)를 이용하여 반복적으로 영역을 확장시켜 떨어진 부분 또는 구멍을 채우거나 잡음을 축소하여 제거하는 등의 연산으로 침식(erode), 팽창(dilate), 열기(opening), 닫기 (closing) 등 이 있음
  - ✓ OpenCV는 cv2.getStructuringElement(). cv2.erode(), cv2.dilate(), cv2. morphologyEx() 등의 모폴로지 연산 함수가 있음

    cv2.getStructuringElement(shape, ksize[, anchor]) -> retval

    - ❑ 모폴로지 연산에 사용하는 사각형 (cv2.MORPH_ RECT), 타원 (cv2.MORPH_ELLIPSE), 십자(cv2.MORPH_CROSS) 등의 shape 모양의 ksize 크기 구조 요소를 반환
    - ❑ anchor = (-1, -1)이면 중심점이 앵커점

모폴로지(Morphology)

- 모폴리지 연산
  - ✓ 침식
    - ❑ 객체의 크기는 축소되고 배경은 확장됨
    - ❑ 객체의 크기가 축소되기 때문에 영상 내에 존재하는 잡음 같은 작은 크기의 객체들은 사라질 수도 있음
    - ❑ 소금-후추(salt & papper) 잡음과 같은 임펄스(impulse) 잡음 들을 제거할 수 있으며 영상 내에서 객체의 돌출부를 감소시키기 때문에 서로 닿는 물체를 분리할 때에도 유용하게 사용할 수 있음

    cv2.erode(src, kernel[, dst[, anchor[, iterations[, borderType[. borderValue]]]]]) -> dst

    - 입력 영상 src에 kernel를 적용하여 모폴로지 침식(erode) 연산을 iterations 만큼 반복하여 dst에 저장
    - kernel = None이면 3 X 3 사각형 커널을 사용
    - 디폴트 anchor 는 kernel의 중심점이 앵커점
    - 입력 영상 src는 cv2.CV_8U, cv2.CV_16U, cv2. CV_16S, cv2.CV_32F, cv2.CV_64F의 픽셀 깊이를 가지며 채널수는 제한이 없으며 그레이스케일 영상에서는 반복적인 min 필터링과 같은데 cv2.erode()로 지역 극소(local minima) 값을 계산할 수 있음

      $$dst(x,y) = \min_{(x',y')} src(x+x', y+y'),$$

      $$where\ (x',y')은\ kernel에서\ 0이\ 아닌\ 위치의\ 이웃$$

모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 침식

모폴로지(Morphology)

- 모폴리지 연산
  - ✓ 팽창 연산
    - 객체의 가장 외곽 픽셀를 확장시키기 때문에 객체의 크기는 확대되고 배경은 축소되며 객체의 팽창으로 인해 서 객체 내부에 있는 빈 공간도 메워지게 됨

cv2.dilate(src, kernel[. dst[, anchor[, iterations[, borderTypet. borderValue]]]]) -> dst

- 입력 영상 src에 kernel를 적용하여 모폴로지 팽창(dilate) 연산을 iterations 만큼 반복하여 dst에 저장
- kernel = None이면 3 X 3 사각형 커널을 사용

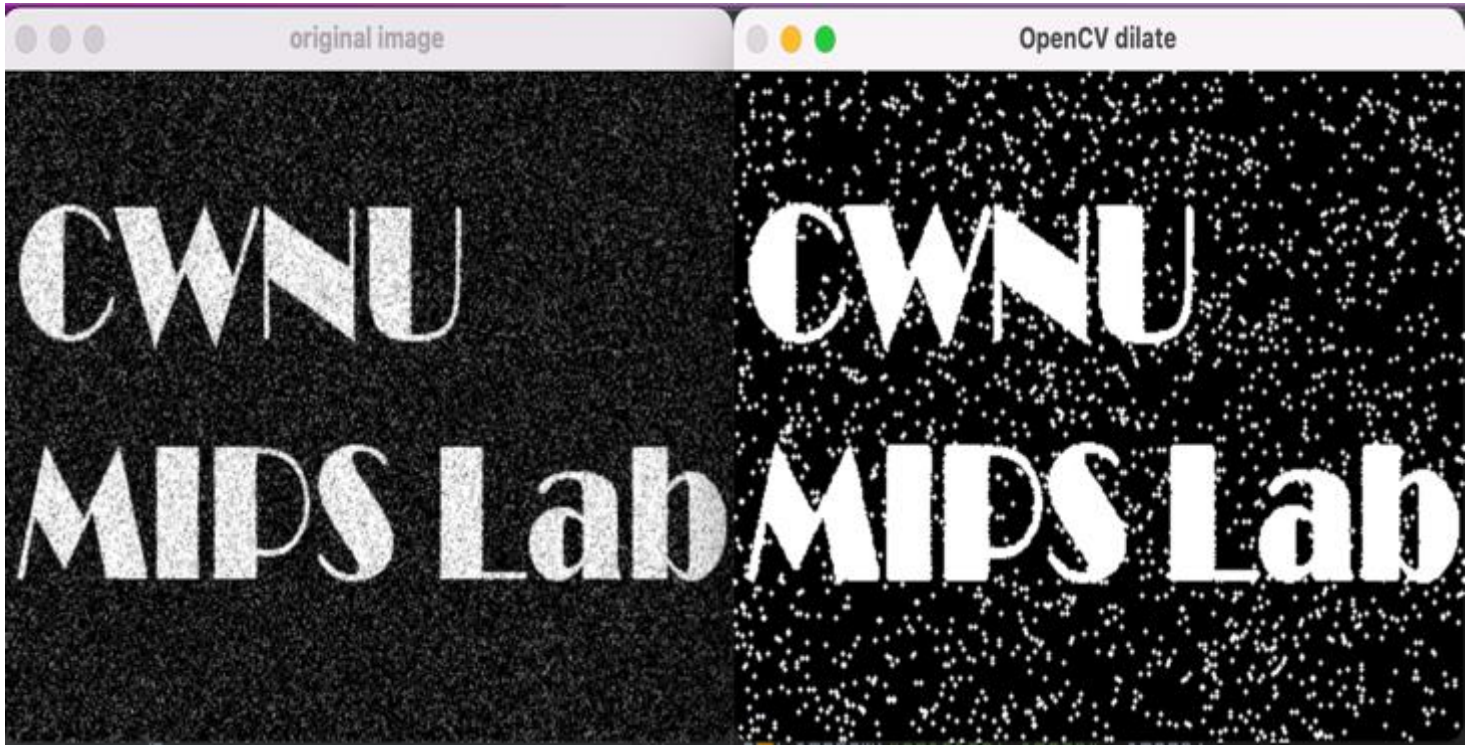
- 그레이스케일 영상에서는 반복적인 max 필터링과 같음
- cv2.dilate()로 지역 극대(local maxima) 값을 계산할 수 있음

$$dst(x,y) = \max\ src(x+x', y+y'),$$

$$where\ (x', y')\text{은 } kernel\text{에서 0이 아닌 위치의 이웃}$$

모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 팽창



모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 열림 연산 과 닫힘 연산
    - ❑ 열림 연산과 닫힘 연산은 모폴로지의 기본 연산인 침식 연산과 팽창 연산의 순서를 조합하여 수행
    - ❑ 열림 연산(opening operation)
      - ○ 침식 연산을 먼저 수행하고 바로 팽창 연산을 수행하는데 침식 연산으로 인해서 객체는 축소되고 배경 부분의 미세한 잡음들은 제거되고 다음으로 축소되었던 객체들이 팽창 연산으로 인해서 다시 원래 크기로 돌아감
      - ○ 그림은 열림 연산의 과정을 예시한 것
      - ○ 침식 연산으로 배경 부분의 잡음을 제거하면서 다시 팽창 연산으로 인해 객체 크기의 축소를 방지할 수 있는데 돌출된 부분은 제거된 후 다시 원래 크기로 돌아가지 않음

모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 열림 연산 과 닫힘 연산
    - ❑ 닫힘 연산(cloning operation)
      - ○ 팽창 연산을 먼저 수행하고 다음으로 침식 연산을 수행
      - ○ 팽창 연산으로 객체가 확장되어서 객체 내부의 빈 공간이 메워짐
      - ○ 다음으로 침식 연산으로 확장되었던 객체의 크기가 원래대로 축소됨

- ❑ 최종 결과 영상을 보면 객체 내부의 비어있던 공간이 채워지며 인접한 객체를 이어지게 하는 효과도 있음

모폴로지(Morphology)

❖ 모폴리지 연산

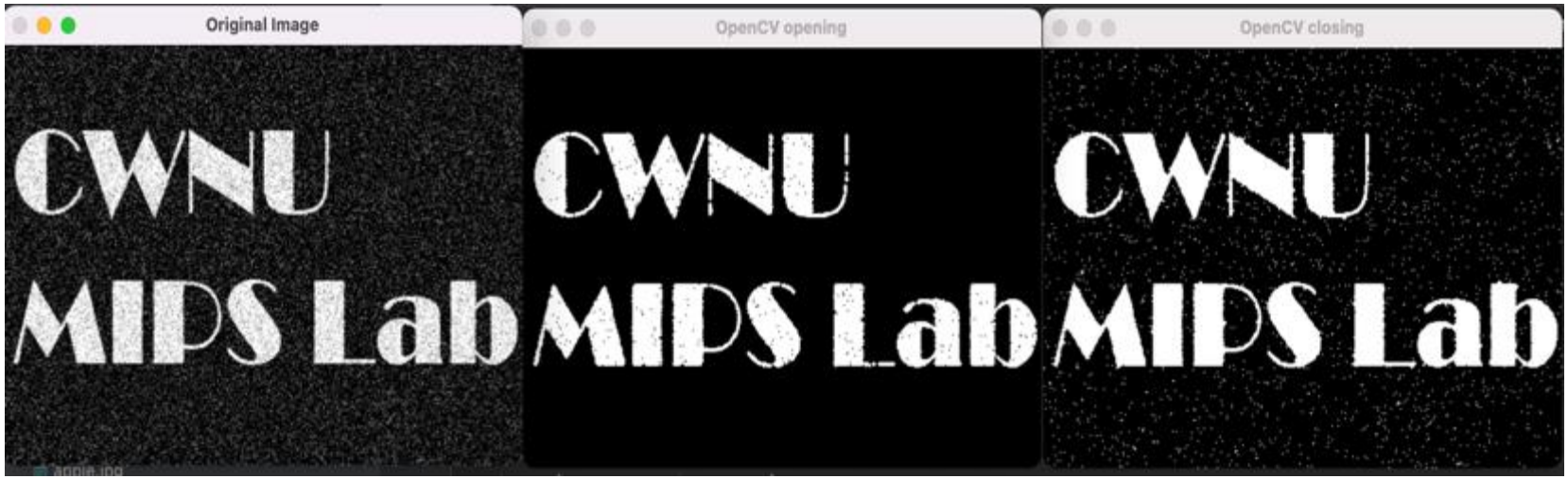
cv2.morphologyEx(src, op, kernelt, dst[, anchor[, iterations[, borderTypet, borderValue]]]]) -> dst

- ❑ 입력 src에 kernel을 적용하여 모폴로지 op 연산을 iterations만큼 반복하여 dst에 저장
- ❑ 모폴로지 연산(op)을 표시

op	설명
cv2.MORPH_OPEN	$dst = dilate(erode(src, kernel), kernel)$
cv2.MORPH_CLOSE	$dst = erode(dilate(src, kernel), kernel)$
cv2.MORPH_GRADIENT	$dst = dilate(src, kernel) - erode(src, kernel)$
cv2.MORPH_TOPHAT	$dst = src - open(src, kernel)$
cv2.MORPH_BLACKHAT	$dst = close(src, kernel) - src$

모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 열림 과 닫힘 연산



모폴로지(Morphology)

- 모폴리지 연산
  - ✓ 열림 과 닫힘 연산
    - 열림 연산은 침식으로 인해서 배경의 잡음(흰색)이 제거되지만 객체 내의 잡음은 커지고 다시 팽창 연산을 수행함으로써 객체 내부의 잡음을 원래 크기로 줄이게 되는데 이때 배경 잡음이 제거되었기에 팽창을 수행함에도 배경 잡음이 커지지 않게 되는데 객체 내부에 대한 변화없이 배경의 잡음을 제거한다.
    - 닫힘 연산은 팽창 연산으로 인해서 객체 내부의 잡음(검은색)이 제거되고 배경의 잡음(흰색)이 증가하고 다시 침식 연산의 수행함으로써 배경의 잡음 만을 원래 크기로 줄이게 되는데 이때 객체 내부의 잡음은 제거되었기에 영향이 없게 되는데 배경에 대한 변화 없이 객체 내부의 잡음을 제거
  - ✓ 모폴로지 연산은 한 번의 수행으로 결과 영상이 미흡할 경우에는 iterations 인수를 통해서 여러 번 반복적으로 수행할 수 있음

모폴로지(Morphology)

- ❖ 모폴리지 연산
  - ✓ 번호판 후보 검출

모폴로지(Morphology)

- 모폴리지 연산
  - ✓ 번호판 검출
    - 콘솔 창에서 입력 영상 번호를 입력하는데 이때 0을 입력하면 프로그램을 종료
    - 번호가 입력되면 data/test_car 디렉토리에 있는 입력 번호의 차량 영상이 image 행렬에 저장됨
    - 영상 파일이 존재하지 않으면 00번 영상파일이 없습니다. 라는 메시지를 출력하고 다음 입력을 받기 위해 대기
    - 영상 파일이 존재하면 블러링과 소벨 에지 검출을 수행한 후 모폴로지 열림 연산을 수행하고 그 결과를 영상으로 표시
    - 단순히 블러링과 소벨 에지 검출에 열림 연산만을 수행했는데 결과 영상에는 번호판과 비슷한 영역들이 검출된 것을 볼 수 있음