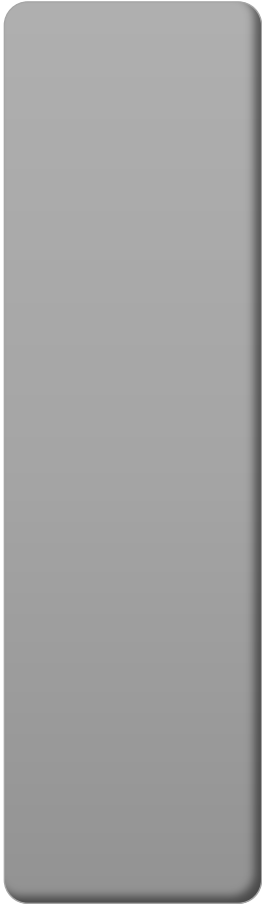


CHAPTER 07

컨볼루션(Convolution)_2

contents

- 
- 7.3 기타 필터링
 - 7.4 모폴로지(morphology)

7.2.5 캐니 에지 검출

- 잡음은 다른 부분과 경계를 이루는 경우 많음
 - 대부분의 에지 검출 방법이 이 잡음들을 에지로 검출
 - 이런 문제를 보완하는 방법 중의 하나가 캐니 에지(Canny Edge) 검출 기법
- 캐니 에지 검출 -여러 단계의 알고리즘으로 구성된 에지 검출 방법

1. 블러링을 통한 노이즈 제거 (가우시안 블러링)
2. 화소 기울기(gradiant)의 강도와 방향 검출 (소벨 마스크)
3. 비최대치 억제(non-maximum suppression)
4. 이력 임계값(hysteresis threshold)으로 에지 결정

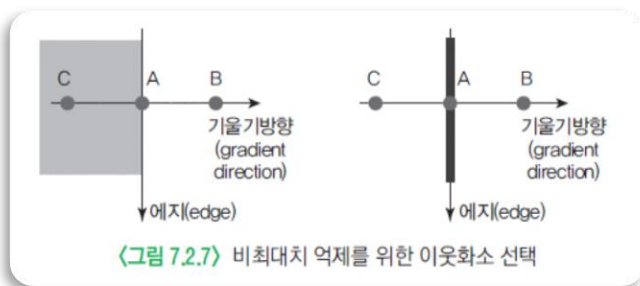
- 3. Non-maximum suppression: 중심 픽셀을 기준으로 값이 가장 큰 경우 그대로 두고, 아닌 경우 값을 제거
- 4. 이력 임계값: 검출된 edge 값의 최대/최소값 기준으로 필터링

7.2.5 캐니 에지 검출

- 1) 블러링 - 5×5 크기의 가우시안 필터 적용
 - 불필요한 잡음 제거 및 필터 크기는 변경가능
- 2) 화소 기울기(gradient) 검출
 - 가로 방향과 세로 방향의 소벨 마스크로 회선 적용
 - 회선 완료된 행렬로 화소 기울기의 크기(magnitude)와 방향(direction) 계산
 - 기울기 방향은 4개 방향(0, 45, 90, 135)으로 근사하여 단순화

7.2.5 캐니 에지 검출

- 3) 비최대치 억제(non-maximum suppression)
 - 기울기의 방향과 에지의 방향은 수직



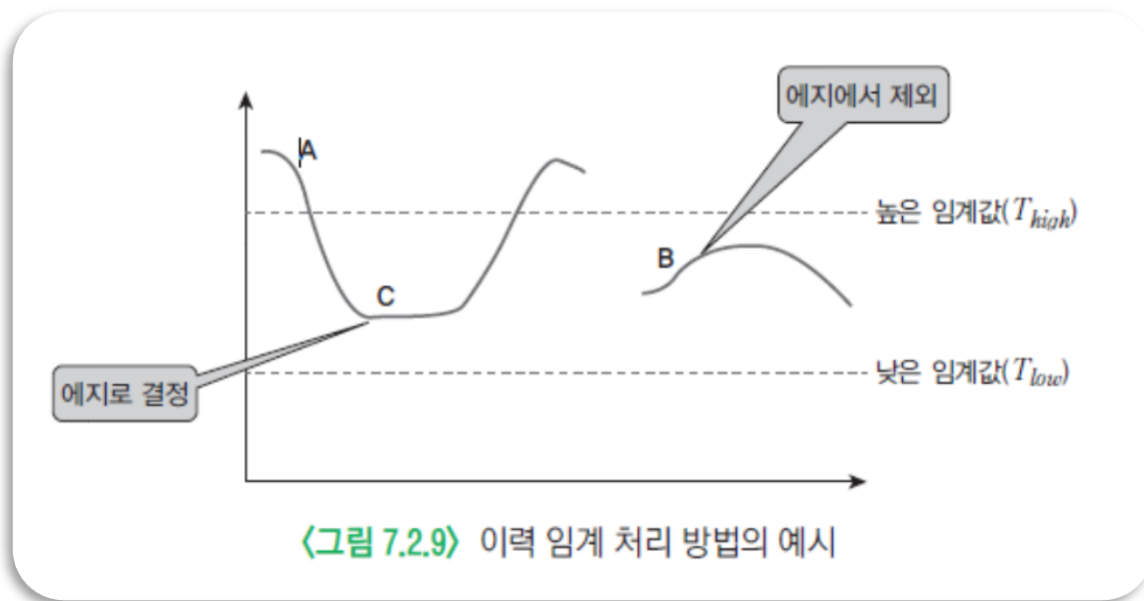
- 기울기 방향에 따른 이웃 화소 선택:
 - 중심 픽셀을 기준으로 값이 가장 큰 경우 그대로 두고, 아닌 경우 값을 제거
 - 검출된 에지들을 좀더 깨끗하게



〈그림 7.2.8〉 비최대치 억제를 위한 이웃 화소 선택

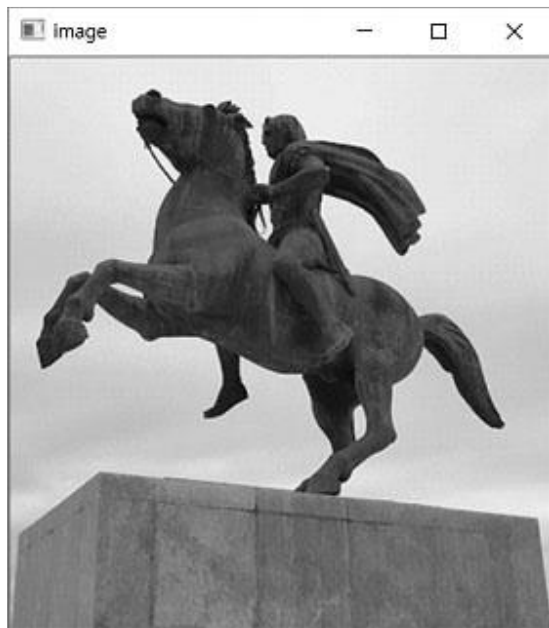
7.2.5 캐니 에지 검출

- 4) 이력 임계값 방법(hysteresis thresholding)
 - 두 개의 임계값(T_{high} , T_{low}) 사용해 에지 이력 추적으로 에지 결정
 - 각 화소에서 높은 임계값보다 크면 에지 추적 시작



7.2.5 캐니 에지 검출

- 실행결과



입력영상



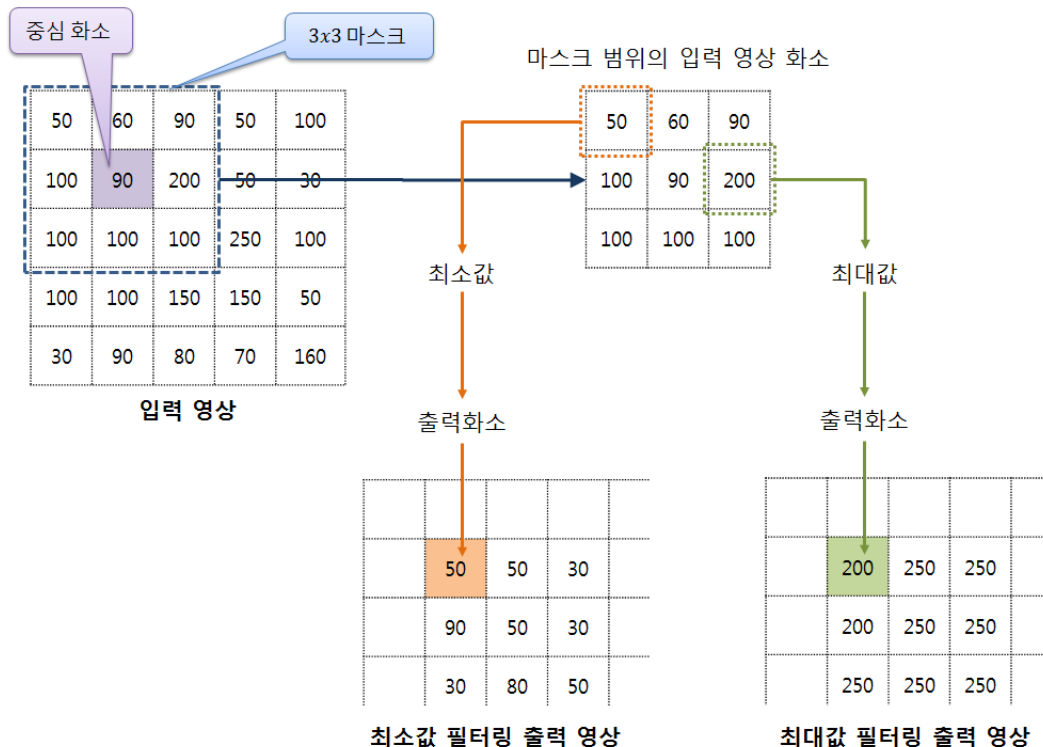
```
image_canny = cv2.Canny(image, t_low, t_high)
```

7.3 기타 필터링

- 7.3.1 최댓값/최솟값 필터링
- 7.3.2 평균값 필터링
- 7.3.3 미디언 필터링
- 7.3.4 가우시안 스무딩 필터링

7.3.1 최댓값/최솟값 필터링

- 입력 영상의 해당 화소(중심화소)에서 마스크로 씌워진 영역의 입력화소들을 가져와서 그 중에 최댓값/최솟값을 출력화소로 결정하는 방법



7.3.1 최댓값/최솟값 필터링

- 최댓값 필터링

- 가장 큰 값인 밝은 색들로 출력화소가 구성
- 돌출되는 어두운 값이 제거 전체적으로 밝은 영상이 됨

- 최솟값 필터링

- 가장 작은 값들인 어두운 색들로 출력화소가 구성
- 돌출되는 밝은 값들이 제거되며, 전체적으로 어두운 영상 됨

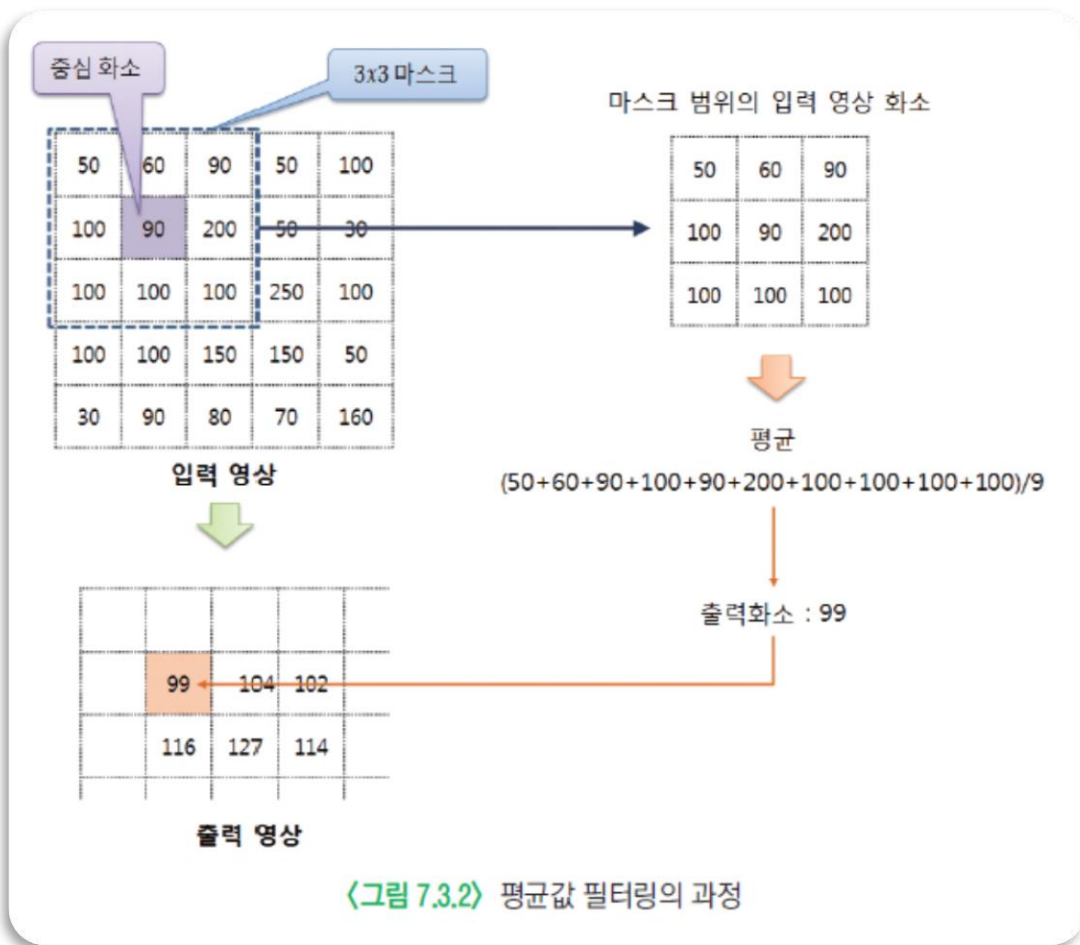
7.3.1 최댓값/최솟값 필터링

- 실행결과



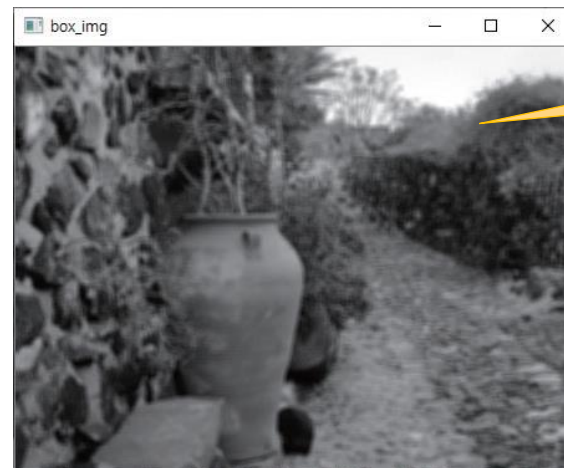
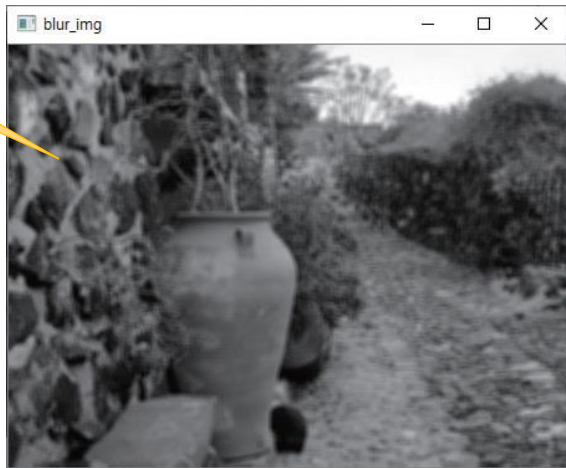
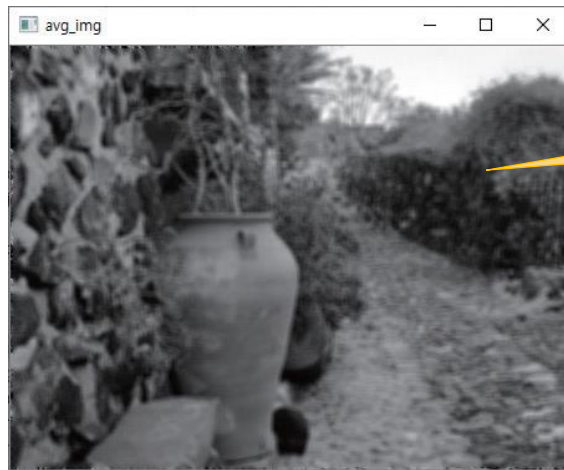
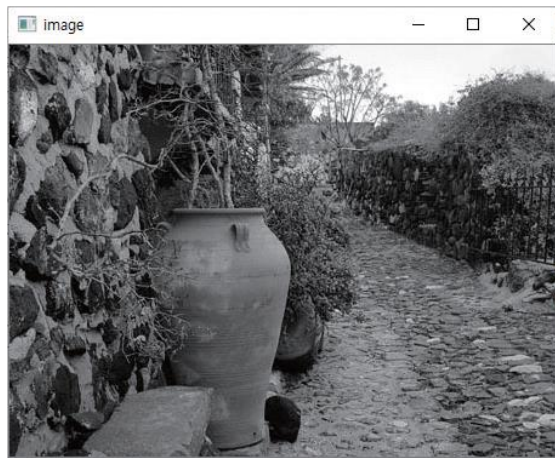
7.3.2 평균값 필터링

- 마스크 영역 입력화소들의 평균을 구하여 출력화소로 지정하는 방법



7.3.2 평균값 필터링

- 실행결과



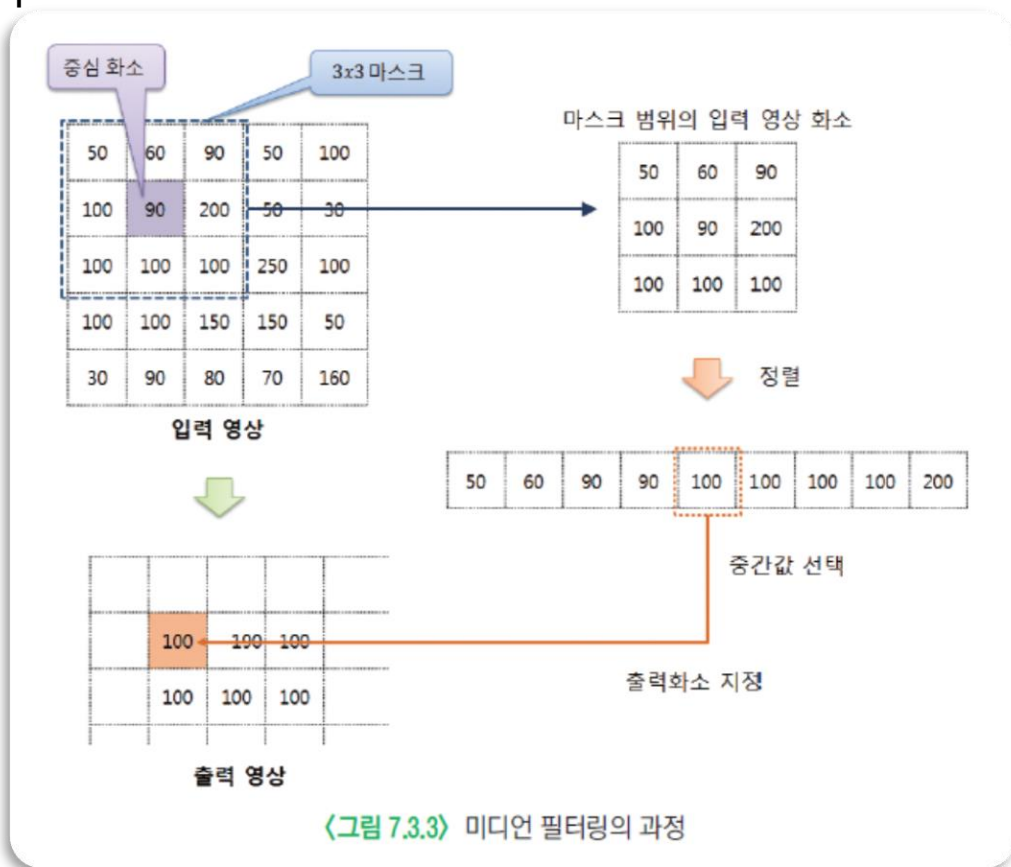
7.3.3 미디언 필터링

- 마스크 범위 원소 중 중간값 취하여 출력화소로 결정하는 방식

- 마스크 범위내에 있는 화소값 정렬 필요
- 임펄스 잡음, 소금-후추 잡음 제거

- RGB 컬러

- 3개채널 간의 상호 의존도가 커서
- 잡음이 많아 질 수 있음



7.3.3 미디언 필터링

예제 7.3.3

미디언 필터링 - 11.filter_median.py

```
01 import numpy as np, cv2
02
03 def median_filter(image, size):                # 미디언 필터링 함수
04     rows, cols = image.shape[:2]
05     dst = np.zeros((rows, cols), np.uint8)
06     center = ksize // 2                        # 마스크 절반 크기
07
08     for i in range(center, rows - center):      # 입력 영상 순회
09         for j in range(center, cols - center):
10             y1, y2 = i - center, i + center + 1 # 마스크 높이 범위
11             x1, x2 = j - center, j + center + 1 # 마스크 너비 범위
12             mask = image[y1:y2, x1:x2].flatten() # 관심 영역 지정 및 벡터 변환
13
14             sort_mask = cv2.sort(mask, cv2.SORT_EVERY_COLUMN) # 정렬 수행
15             dst[i, j] = sort_mask[sort_mask.size//2] # 출력 화소로 지정
16     return dst
17
18 def salt_pepper_noise(img, n):                  # 소금 후추 잡음 생성 함수
19     h, w = img.shape[:2]
20     x, y = np.random.randint(0, w, n), np.random.randint(0, h, n)
21     noise = img.copy()
22     for (x, y) in zip(x, y):
23         noise[y, x] = 0 if np.random.rand() < 0.5 else 255
24     return noise
```

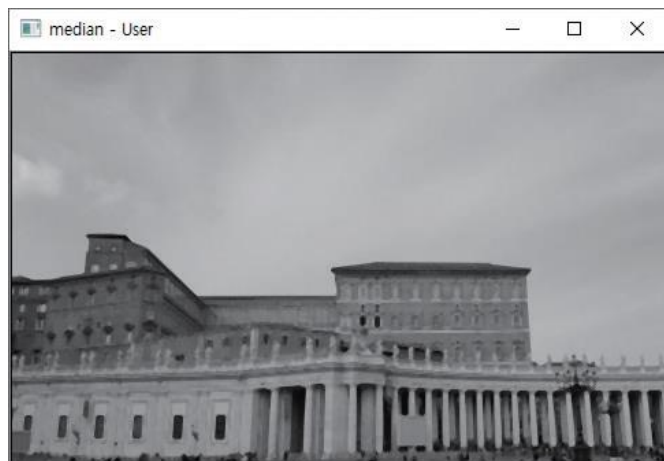
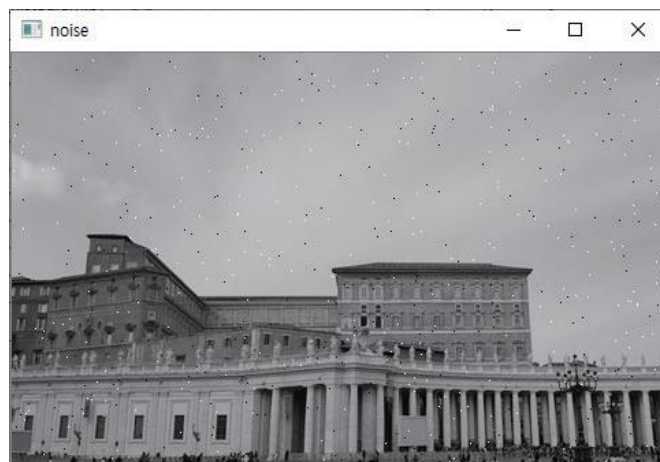
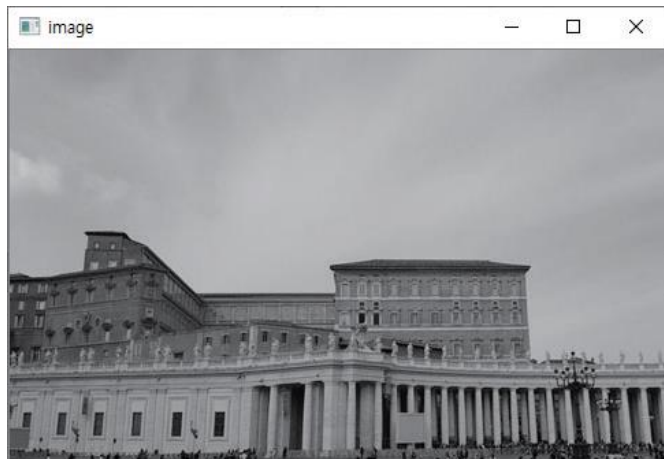
7.3.3 미디언 필터링

```
26 image = cv2.imread("images/median.jpg", cv2.IMREAD_GRAYSCALE)
27 if image is None: raise Exception("영상파일 읽기 오류")
28
29 noise = salt_pepper_noise(image, 500)           # 소금-후추 잡음 영상 생성
30 med_img1 = median_filter(noise, 5)              # 사용자 정의 함수
31 med_img2 = cv2.medianBlur(noise, 5)            # OpenCV 제공 함수
32
33 cv2.imshow("image", image),
34 cv2.imshow("noise", noise),
35 cv2.imshow("median - User", med_img1)
36 cv2.imshow("median - OpenCV", med_img2)
37 cv2.waitKey(0)
```

5x5 크기 미디언 필터링

7.3.3 미디언 필터링

- 실행결과



7.3.4 가우시안 스무딩 필터링

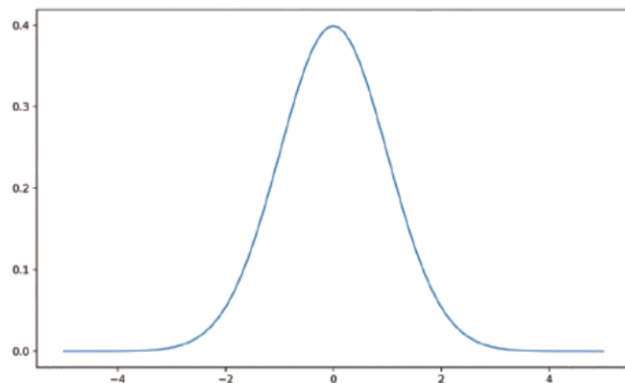
- 스무딩

- Convolution을 통해서 영상의 세세한 부분을 부드럽게 하는 기법
- 대표적인 방법 - 가우시안 필터링

$$N(\mu, \sigma)(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- 가우시안 분포(정규 분포)

- 특정 값의 출현 비율을 그래프로 그렸을 때, 평균에서 가장 큰 수치 가짐
- 평균을 기준으로 좌우 대칭 형태
- 양끝으로 갈수록 수치가 낮아지는 종 모양
 - 평균과 표준 편차로 그래프 모양 변경

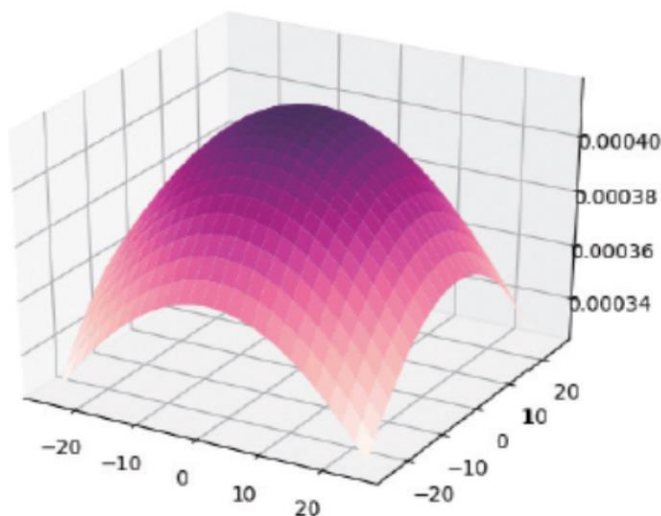


〈그림 7.3.4〉 정규 분포 그래프

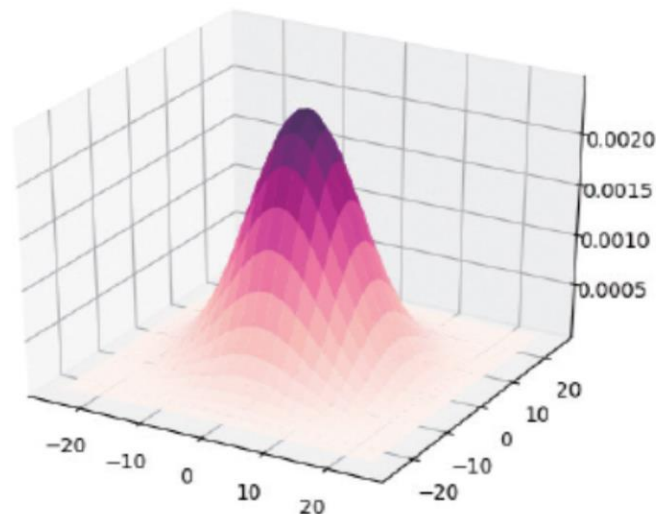
7.3.4 가우시안 스무딩 필터링

- 2차원 가우시안 분포

$$N(\mu, \sigma_x, \sigma_y)(x, y) = \frac{1}{\sigma_x \sigma_y 2\pi} \exp\left[-\left(\frac{(x-\mu)^2}{2\sigma_x^2} + \frac{(y-\mu)^2}{2\sigma_y^2}\right)\right]$$



ksize = (50, 50), $\sigma_x, \sigma_y = 50, 50$



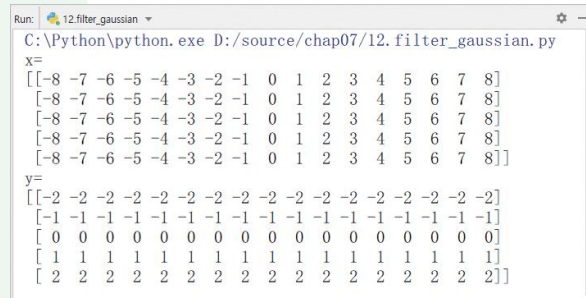
ksize = (50, 50), $\sigma_x, \sigma_y = 8.8$

〈그림 7.3.5〉 2차원 정규 분포 그래프의 예

7.3.4 가우시안 스무딩 필터링

예제 7.3.4 가우시안 필터링 - 12.filter_gaussian.py

```
01 import numpy as np, cv2
02
03 def getGaussianMask(ksize, sigmaX, sigmaY):          # 가우시안 마스크 생성 함수
04     sigma = 0.3 * ((np.array(ksize) - 1.0) * 0.5 - 1.0) + 0.8
05     if sigmaX <= 0: sigmaX = sigma[0]                # 표준편차 양수 아닐 때,
06     if sigmaY <= 0: sigmaY = sigma[1]                # ksize로 기본 표준편차 계산
07
08     u = np.array(ksize)//2                           # 커널 크기 절반
09     x = np.arange(-u[0], u[0]+1, 1)                  # x 방향 범위
10     y = np.arange(-u[1], u[1]+1, 1)                  # y 방향 범위
11     x, y = np.meshgrid(x, y)                         # 정방 행렬 생성
12
13     ratio = 1 / (sigmaX * sigmaX * 2 * np.pi)
14     v1 = x ** 2 / (2 * sigmaX ** 2)
15     v2 = y ** 2 / (2 * sigmaY ** 2)
16     mask = ratio * np.exp(-(v1+v2))                  # 2차원 정규분포 수식
17     return mask / np.sum(mask)                       # 원소 전체 합 1 유지
```



```
Run: 12.filter_gaussian
C:\Python\python.exe D:/source/chap07/12.filter_gaussian.py
x=
[[-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8]
 [-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8]
 [-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8]
 [-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8]
 [-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8]]
y=
[[-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1]
 [ 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2]]
```

7.3.4 가우시안 스무딩 필터링

```
18
19 image = cv2.imread("images/smoothing.jpg", cv2.IMREAD_GRAYSCALE)
20 if image is None: raise Exception("영상파일 읽기 오류")
21
22 ksize = (17, 5) # 커널 크기: 가로×세로
23 gaussian_2d = getGaussianMask(ksize, 0, 0)
24 gaussian_1dX = cv2.getGaussianKernel(ksize[0], 0, cv2.CV_32F) # 가로 방향 마스크
25 gaussian_1dY = cv2.getGaussianKernel(ksize[1], 0, cv2.CV_32F) # 세로 방향 마스크
26
27 gauss_img1 = cv2.filter2D(image, -1, gaussian_2d) # 사용자 생성 마스크 적용
28 gauss_img2 = cv2.GaussianBlur(image, size, 0) # OepnCV 제공1-가우시안 블러링
29 gauss_img3 = cv2.sepFilter2D(image, -1, gaussian_1dX, gaussian_1dY) # OpenCV 제공2
30
31 titles = ['image', 'gauss_img1', 'gauss_img2', 'gauss_img3']
32 for t in titles: cv2.imshow(t, eval(t)) # 문자열 리스트로 행렬들을 윈도우 표시
33 cv2.waitKey(0)
```

가로 방향이 큰 값

2차원 가우시안 마스크 생성

커널 크기: 가로×세로

가로 방향 마스크

세로 방향 마스크

1차원 가우시안 마스크 생성
- x, y 방향

가우시안 블러링
으로 수행

사용자 생성 마스크 적용

OepnCV 제공1-가우시안 블러링

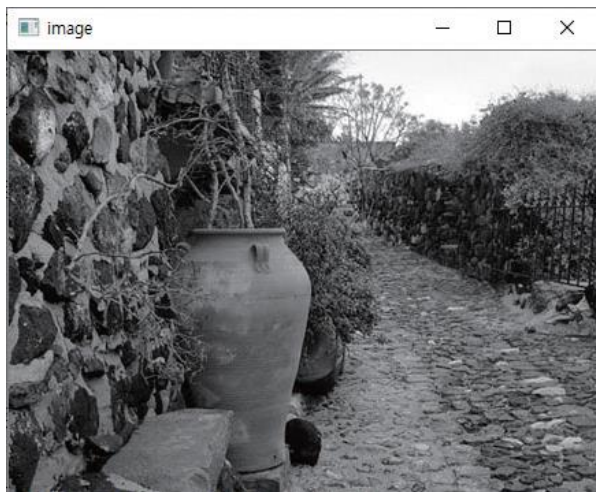
OpenCV 제공2

1차원 가우시안 마스크로
cv2.setFilter2D() 함수에 적용

문자열 리스트로 행렬들을 윈도우 표시

7.3.4 가우시안 스무딩 필터링

- 실행결과



가로 방향으로 흐림



심화 예제 – 블러링 & 캐니에지

심화예제 7.3.5

블러링과 캐니 에지를 이용한 컬러 에지 검출 - 13.edge_color_canny.py

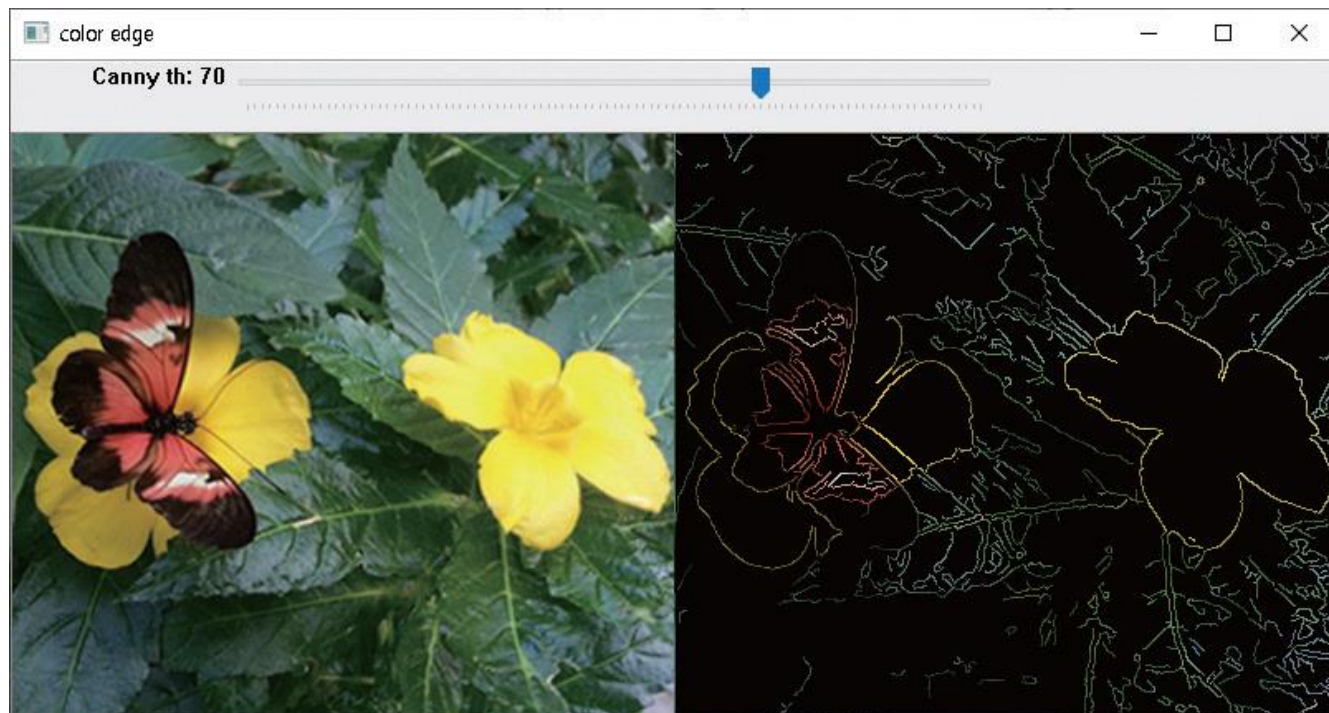
```
01 import cv2
02
03 def onTrackbar(th):                                # 트랙바 콜백 함수
04     rep_edge = cv2.GaussianBlur(rep_gray, (5, 5), 0)    # 가우시안 블러링
05     rep_edge = cv2.Canny(rep_edge, th, th*2, 5)        # 캐니 에지 검출
06     h, w = image.shape[:2]
07     cv2.rectangle(rep_edge, (0, 0, w, h), 255, -1)      # 흰색 사각형 그리기
08     color_edge = cv2.bitwise_and(rep_image, rep_image, mask=rep_edge)
09     cv2.imshow("color edge", color_edge)
10
11 image = cv2.imread("images/color_edge.jpg", cv2.IMREAD_COLOR)
12 if image is None: raise Exception("영상파일 읽기 오류")
13
14 th = 50
15 rep_image = cv2.repeat(image, 1, 2)                  # 가로 반복 복사
16 rep_gray = cv2.cvtColor(rep_image, cv2.COLOR_BGR2GRAY)    # 명암도 영상 변환
17
18 cv2.namedWindow("color edge", cv2.WINDOW_AUTOSIZE)      # 윈도우 생성
19 cv2.createTrackbar("Canny th", "color edge", th, 100, onTrackbar) # 콜백 함수 등록
20 onTrackbar(th)                                          # 콜백 함수 첫 실행
21 cv2.waitKey(0)
```

에지만 검출위해
명암도 영상에서
부드러운 부분 제거

복사시 에지 영상을 마스크로 사용
- 컬러 영상의 에지 부분만 복사

심화예제

- 실행결과



7.4 모폴로지(morphology)

- 7.4.1 침식 연산
- 7.4.2 팽창 연산
- 7.4.3 열림 연산과 닫힘 연산

7.4 모폴로지(morphology)

- 모폴로지 – 형태학

- 생물학

- 생물형태의 기술과 그 법칙성의 탐구를 목적으로 일반적으로 해부학과 발생학을 합쳐서 형태학이라 부름

- 의학

- 인체 형태학이란 의미로 사용

- 체육학

- 스포츠 운동 형태학이란 개념으로 사용

- 영상 처리에서 형태학

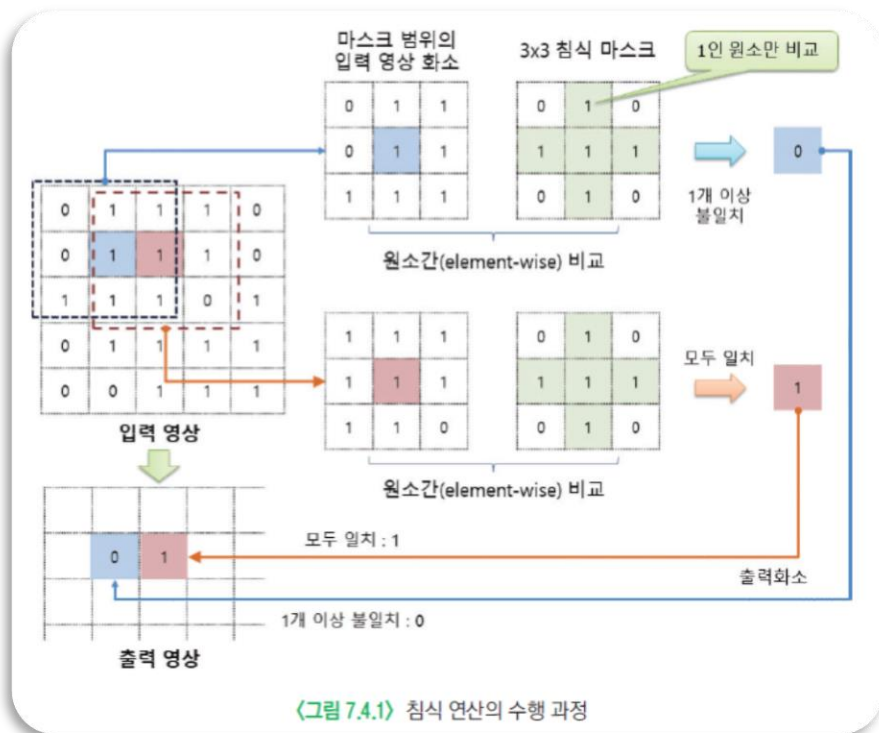
- 영상의 객체들의 형태(shape)를 분석하고 처리하는 기법

- 영상의 경계, 골격, 블록 등의 형태를 표현하는데 필요한 요소 추출

- 영상 내에 존재하는 객체의 형태를 조금씩 변형시킴으로써 영상 내에서 불필요한 잡음 제거하거나 객체를 뚜렷하게 함

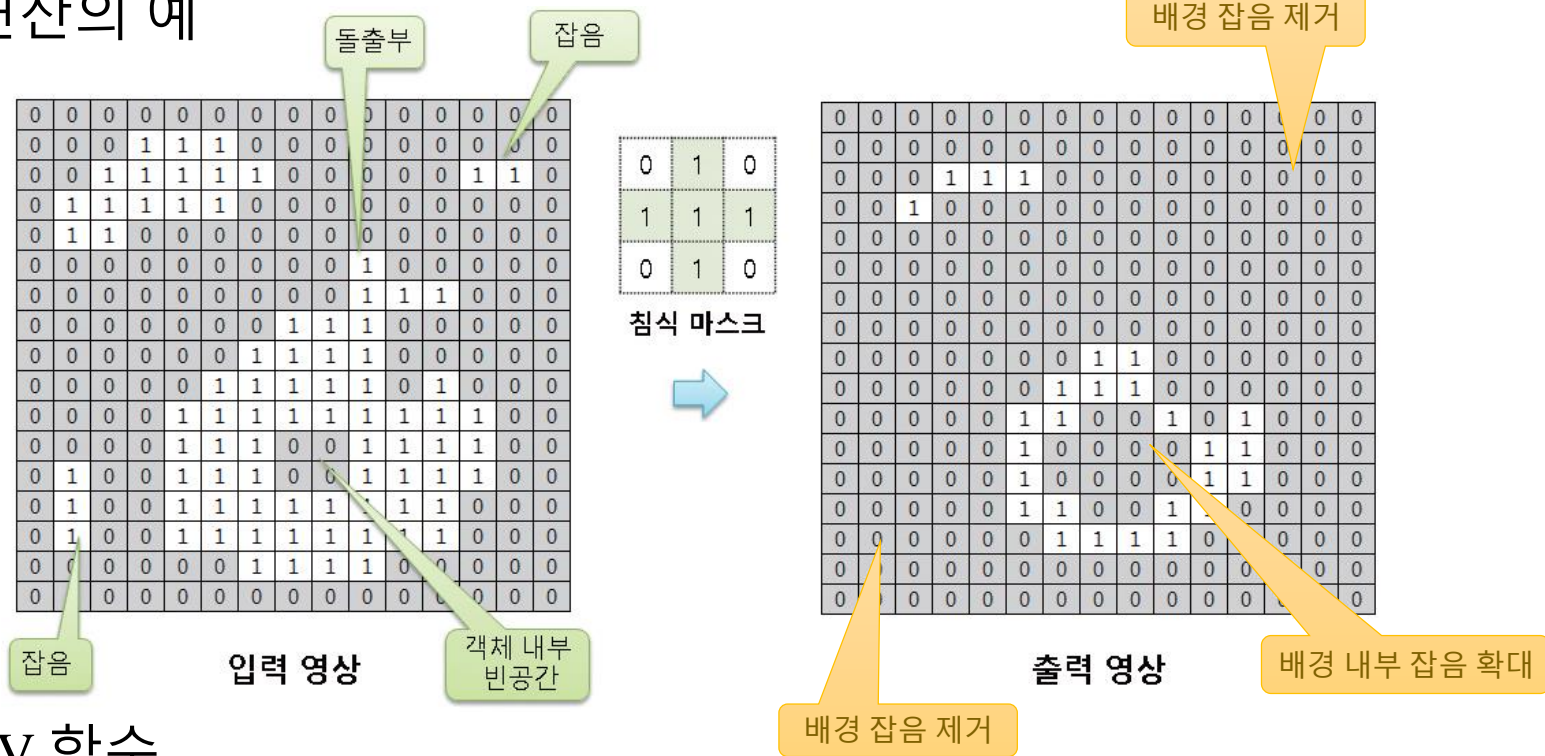
7.4.1 침식 연산

- 객체를 침식시키는 연산
 - 객체의 크기 축소, 배경 확장
 - 영상 내에 존재하는 잡음 같은 작은 크기의 객체 제거 가능
 - 소금-후추 잡음과 같은 임펄스 잡음 제거



7.4.1 침식 연산

• 침식 연산의 예



• OpenCV 함수

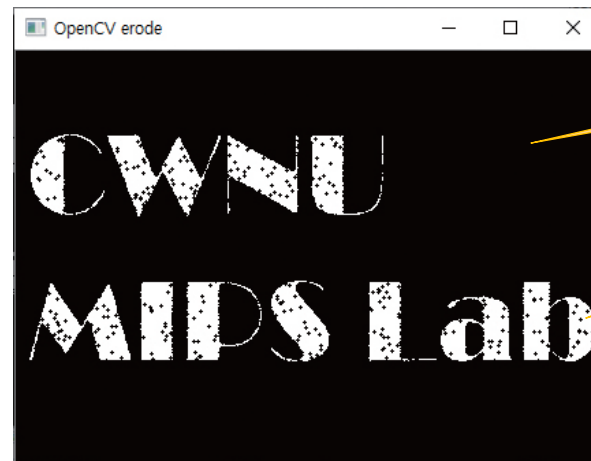
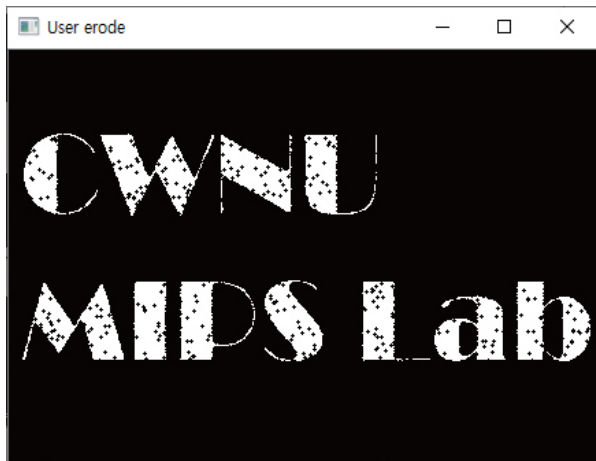
- cv2.morphologyEx() - 옵션 : MORPH_ERODE
- cv2.erode()

7.4.1 침식 연산

- 실행 결과



입력영상

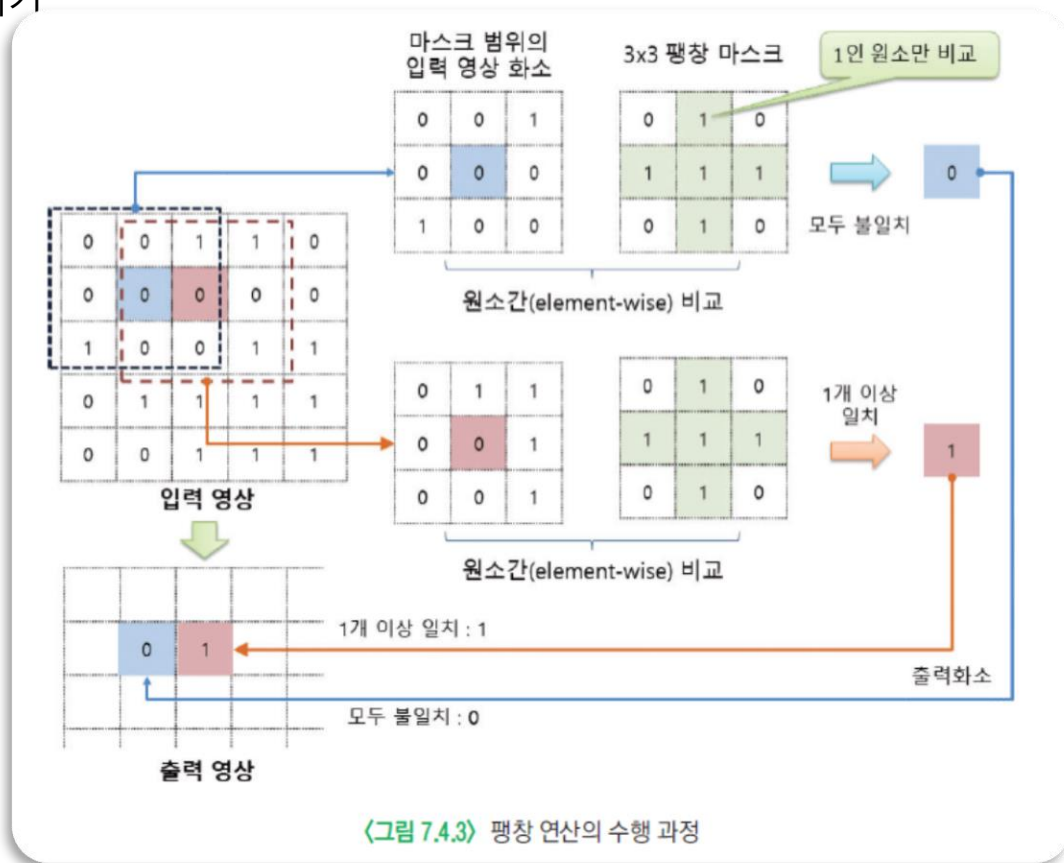


배경 잡음 제거

객체 내부 잡음 확대

7.4.2 팽창 연산

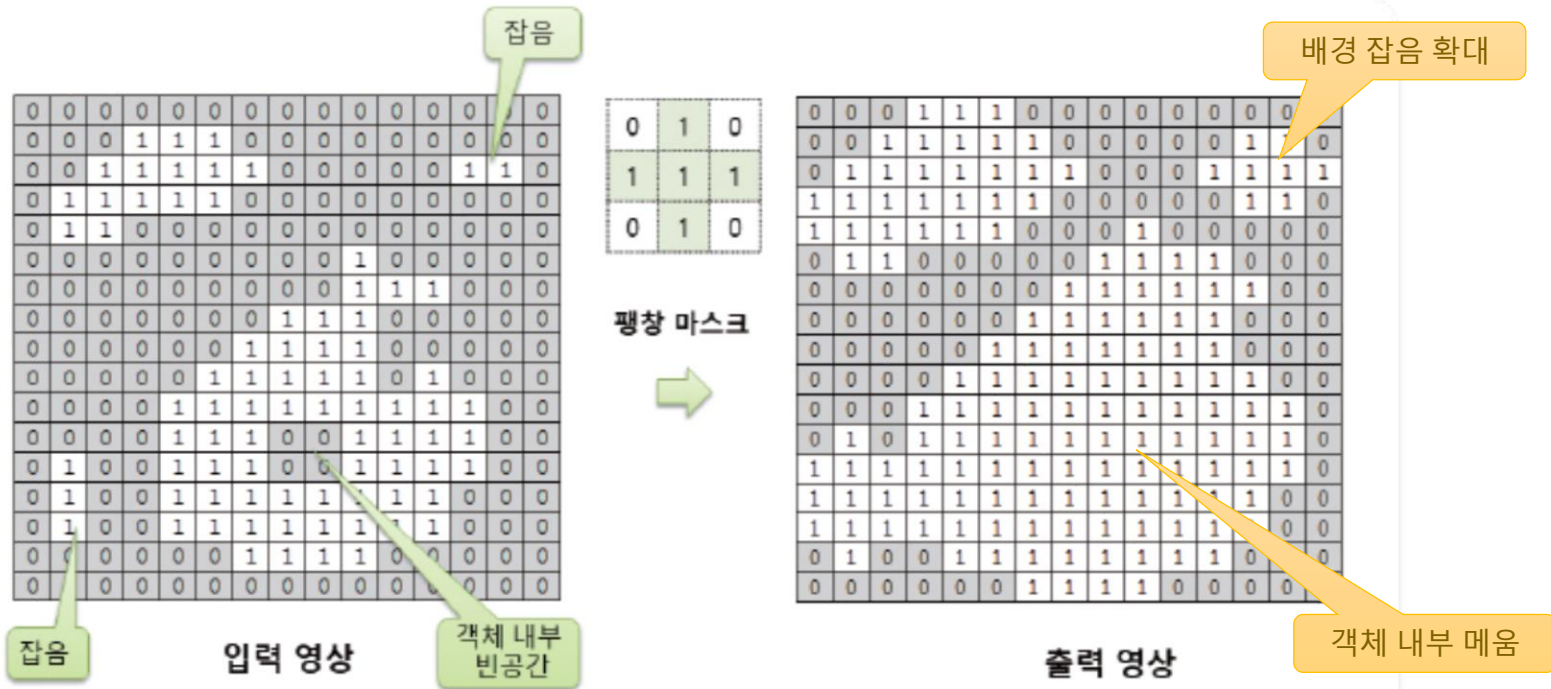
- 객체를 팽창시키는 연산
 - 객체의 최외곽 화소를 확장시키는 기능 → 객체 크기 확대, 배경 축소
 - 객체 팽창으로 객체 내부의 빈 공간도 메워짐
 - 객체 내부 잡음 제거



〈그림 7.4.3〉 팽창 연산의 수행 과정

7.4.2 팽창 연산

• 팽창 연산의 예



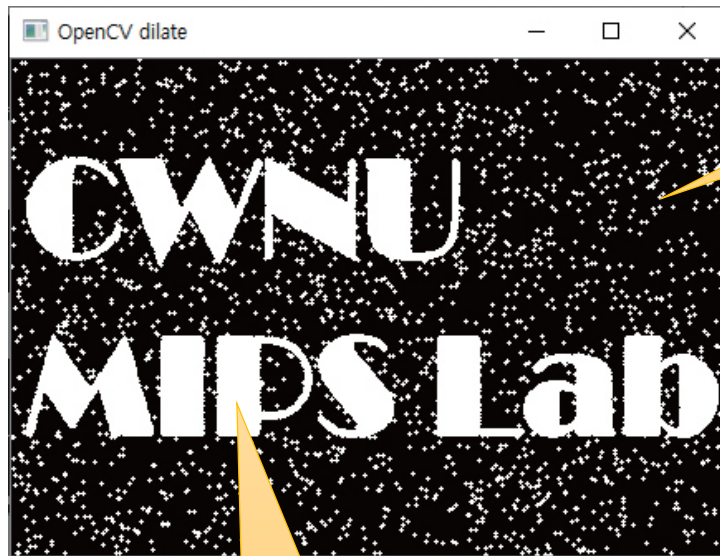
〈그림 7.4.4〉 팽창 연산의 예

7.4.2 팽창 연산

- 실행결과



입력영상



배경 잡음 증가

객체 내부 잡음 제거

단원 요약

- 캐니 에지 검출 방법은 John F. Canny에 의해 개발된 것으로서 다음과 같은 여러 단계의 알고리즘으로 구성된 에지 검출 방법이다.
- 비선형 공간 필터링의 방법으로 최솟값, 최댓값 필터링, 미디언 필터링 등이 있다.
- 모폴로지는 행태학적 방법을 영상 처리에 적용한 것으로서 침식과 팽창 연산이 있다. 침식연산은 객체의 크기가 축소되기 때문에 영상 내에 존재하는 작은 크기의 잡음을 제거하는 효과적이다. 팽창 연산은 객체의 크기가 확대되어 객체 내부의 빈 공간을 메우는 역할을 한다.

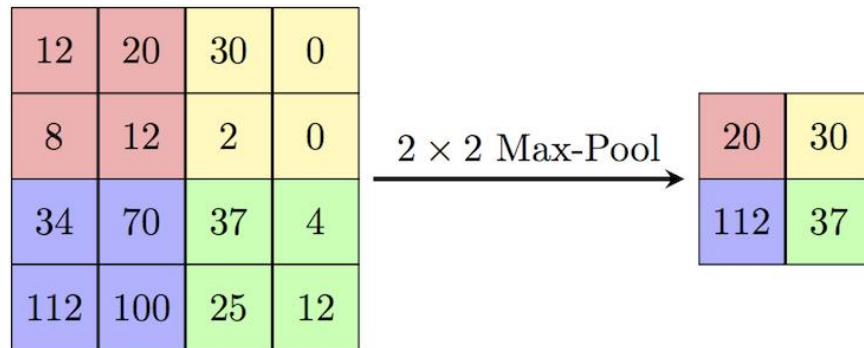
7. 실습 과제

- (과제) 연습문제 14. (p.354)

- 캐니 에지 알고리즘에서 이중 임계값을 트랙바로 만들어서 두개의 임계값을 조절하여 에지를 검출하도록 프로그램을 구현하시오.
- 다음 함수 이용:
 - `image_canny = cv2.Canny(image, t_low, t_high)`

- (보너스)

- Maxpooling은 딥러닝에서 널리 쓰이는 기능중 하나이다. 아래와 같은 2x2 Maxpooling은 필터 사이즈 영역 안의 최대 값을 추출하는 필터링으로 구현할 수 있다. Grayscale 이미지에 대한 Maxpooling을 직접 구현하고 결과를 출력하시오.



7. 실습 규칙

- 실습 과제는 실습 시간내로 해결해야 합니다.
 - 해결 못한경우 실습 포인트를 얻지 못합니다.
 - -> 집에서 미리 연습하고 오길 권장합니다.
- 코드 공유/보여주기 금지. 의논 가능.
- 보너스문제까지 해결한 학생은 조기 퇴실 가능