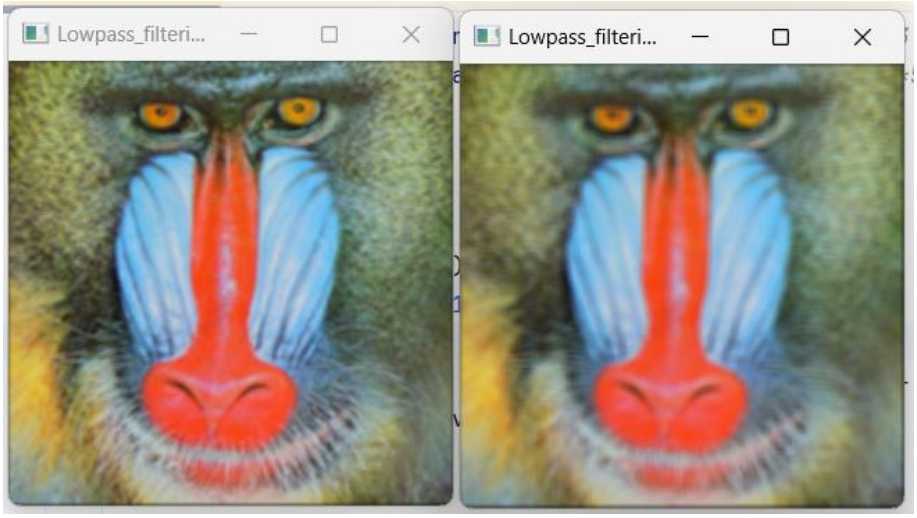
Lab2_2013755_강단이

1.1 Lowpass Filtering

세부 기능 설명

마스크가 3*3일 때, 5*5일 때 두가지를 시행하기 위해 image와 똑 같은 크기의 dst3, dst5을 선언해줬고, 1/9값을 가지는 3*3 마스크와 1/25값을 가지는 5*5 마스크를 만들었다. 모든 픽셀에 대해 3*3 필터를 적용했고, 필터의 가중치와 주변 픽셀 값을 곱해 새로 계산하며 rint로 반올림해주었다. 마찬가지로 5*5 마스크도 적용을 해주며 실행된 결과를 imshow와 imwrite로 보여주고 저장했다.

기능별 실행 화면 캡쳐



좌측부터 3*3 filtering, 5*5 filtering

소스코드

```
import cv2 as cv
import numpy as np

src = cv.imread("Mandrill.bmp",cv.IMREAD_COLOR) #Read the file
H,W,C = src.shape[:]

dst3 = np.zeros((H, W, C), src.dtype)
dst5 = np.zeros((H, W, C), src.dtype)

array_3 = [[1/9 for col in range(3)] for row in range(3)] #3*3 mask
array_5 = [[1/25 for col in range(5)] for row in range(5)] #5*5 mask

for i in range(H):
    for j in range(W):
        val = 0
        for mi in range(-1, 2): #-1 ~ 1
            for mj in range(-1, 2):
                if (-1 < i+mi < H) and (-1 < j + mj < W):
                    val = val + src[i+mi, j+mj] * array_3[mi + 1][mj + 1]
        dst3[i ,j] = np.rint(val)

        val = 0
        for mi in range(-2, 3): #-2 ~ 2
            for mj in range(-2, 3):
                if (-2 < i+mi < H) and (-2 < j + mj < W):
                    val = val + src[i+mi, j+mj] * array_5[mi + 2][mj + 2]
        dst5[i, j] = np.rint(val)


cv.imshow("Lowpass_filtering_3", dst3)
cv.imshow("Lowpass_filtering_5", dst5)
cv.imwrite("./Lowpass_filtering_3.bmp", dst3)
cv.imwrite("./Lowpass_filtering_5.bmp", dst5)
cv.waitKey(0)
```

1.2 Noise(Median) Filtering

세부 기능 설명

Pepper_noise 영상을 읽어와 결과값을 저장할 dst와 주변 픽셀 계산에 사용할 array를 선언한다. 영상 크기만큼 픽셀을 반복해 검사하며 주변 픽셀값을 array에 저장하고 정렬을 수동으로 해준 뒤에 중앙값을 찾아 num/2 위치의 값을 결과값에 저장해준다. 출력하고 저장한다.

기능별 실행 화면 캡쳐

소스코드

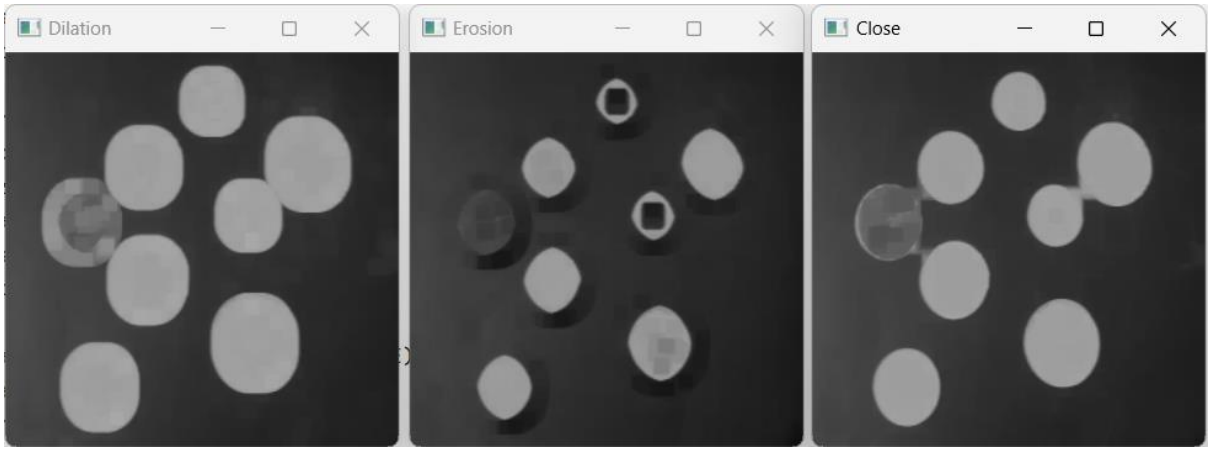
```
import cv2 as cv
import numpy as np

src = cv.imread("pepper_noise.bmp", cv.IMREAD_GRAYSCALE)
H,W = src.shape[:]
dst = np.zeros((H,W), src.dtype)
array = np.zeros(9, src.dtype)

for i in range(H):
    for j in range(W):
        num=0
        for mi in range(-1,2):
            for mj in range(-1,2):
                if(0 < i + mi < H) and (0 < j + mj < W):
                    array[num]=src[i+mi,j+mj]
                    num +=1

        for si in range(num-1):
            for sj in range(num-1-si):
                if array[sj] > array[sj+1]:
                    array[sj], array[sj+1] = array[sj+1], array[sj]
        dst[i,j] = array[int(num/2)]

cv.imwrite("./median_filtering.bmp", dst)
cv.imshow("median_filtering", dst)
cv.waitKey(0)
```

1.3 Morphology

세부 기능 설명

Coin 영상을 불러오고 픽셀마다 그 픽셀 9*9 범위를 검사해 제일 큰 값을 해당 자리로 저장하는 Dilation 함수를 만든다. 마찬가지로 9*9 범위로 검사해 제일 작은 값을 해당 픽셀 값으로 저장하는 Erosion 함수를 만든다. Closing을 통해 Dilation을 먼저 하고, 그 값을 다시 Erosion시켜 반환한다.

기능별 실행 화면 캡쳐



소스코드

```
from nt import scandir
from sys import deactivate_stack_trampoline

import cv2 as cv
import numpy as np

#Dilation
def Dilation(src):
    H,W = src.shape[:]
    dst = np.zeros((H, W), src.dtype)

    for i in range(H):
        for j in range(W):
            max = -1
            for mi in range(-4, 5):
                for mj in range(-4, 5):
                    if (0 < i + mi < H) and (0 < j + mj < W):
                        k = src[i + mi, j + mj]
                        if (k > max):
                            max = k

            dst[i, j] = max
    return dst

#Erosion
def Erosion(src):
    H,W = src.shape[:]
    dst = np.zeros((H, W), src.dtype)

    for i in range(H):
        for j in range(W):
            min = 999
            for mi in range(-4, 5):
                for mj in range(-4, 5):
                    if (0 < i + mi < H) and (0 < j + mj < W):
```

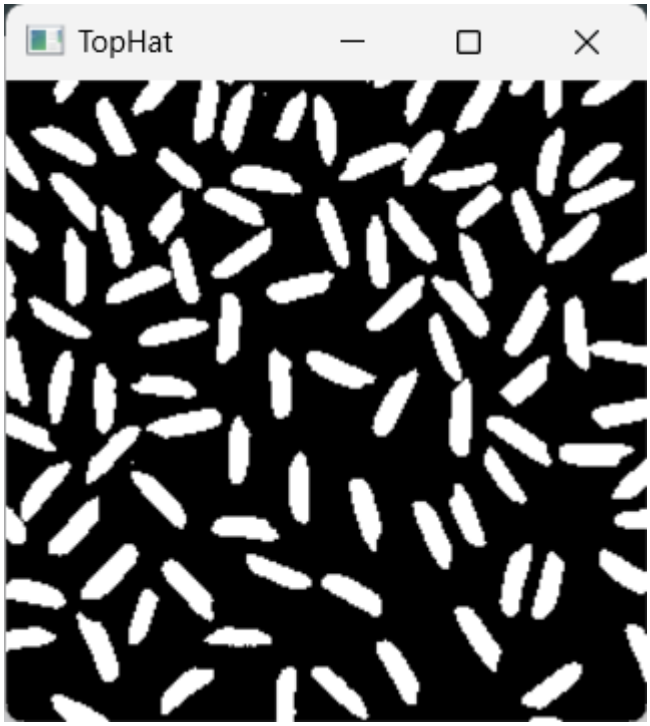
```
                        k = src[i + mi, j + mj]
                        if (k < min):
                            min = k

            dst[i, j] = min
    return dst

def Closing(src):
    dst = Dilation(src)
    dst = Erosion(dst)
    return dst

src = cv.imread("coin.bmp",cv.IMREAD_GRAYSCALE) #Read the file
dilated = Dilation(src)
erosioned = Erosion(src)
closed = Closing(src)

cv.imshow("Dilation", dilated)
cv.imshow("Erosion", erosioned)
cv.imshow("Close", closed)
cv.imwrite("./Dilation.bmp", dilated)
cv.imwrite("./Erosion.bmp", erosioned)
cv.imwrite("./Close.bmp", closed)
cv.waitKey(0)
```

1.4 Morphology Implementation

세부 기능 설명

위에서 만든 Erosion, Dilation 함수를 그대로 사용했고, 지난 실습 때 사용한 Otsu Algorithm도 그대로 사용했다. 다만 Opening 함수를 새로 만들어 Erosion을 먼저하고, 그 후에 Dilation을 하게 만들었으며 검사하는 범위를 Size로 받을 수 있게 만들어 실습 결과가 동일하게 나올 수 있는 size값을 여러 번 찾아봤고, Opening에 매개변수 7을 넣어 쌀을 제외한 것만 잡게 size값을 조정해주었다. 이렇게 나온 값을 본래의 이미지 픽셀에서 빼주었으며, 이 과정에서 runtime error로 범위 오류가 날 수 있기 때문에 np.int32를 붙여주었다. 그렇게해서 나온 결과를 otsu 알고리즘을 적용해 binary 함수에 넣어서 결과값을 내었다.

기능별 실행 화면 캡쳐



소스코드

```
import cv2 as cv
import numpy as np

#Dilation
def Dilation(src,size):
    H,W = src.shape[:]
    dst = np.zeros((H, W), src.dtype)

    for i in range(H):
        for j in range(W):
            max = -1
            for mi in range(-size, size+1):
                for mj in range(-size, size+1):
                    if (0 < i + mi < H) and (0 < j + mj < W):
                        k = src[i + mi, j + mj]
                        if (k > max):
                            max = k

            dst[i, j] = max
    return dst

#Erosion
def Erosion(src,size):
    H,W = src.shape[:]
    dst = np.zeros((H, W), src.dtype)

    for i in range(H):
        for j in range(W):
            min = 999
```

```
            for mi in range(-size, size+1):
                for mj in range(-size, size+1):
                    if (0 < i + mi < H) and (0 < j + mj < W):
                        k = src[i + mi, j + mj]
                        if (k < min):
                            min = k

            dst[i, j] = min
    return dst

#Opening
def Opening(src, size):
    dst = Erosion(src,size)
    dst = Dilation(dst,size)
    return dst

#Otsu
def binary(dst):
    output = np.zeros((H, W), dst.dtype)
    P = np.zeros(256)
    hist = np.zeros(256)
    q1 = np.zeros(256)
    q2 = np.zeros(256)
    mu1 = np.zeros(256)
    mu2 = np.zeros(256)
    sigma1 = np.zeros(256)
    sigma2 = np.zeros(256)
    sigmamu = np.zeros(256)

    for y in range(H):
        for x in range(W):
            k = dst[y, x]
            hist[k] = hist[k] + 1

    for i in range(256):
        P[i] = hist[i] / (H * W)   # 명암값이 i일 확률 P[i]

    for t in range(256):
        for i in range(t + 1):
            q1[t] = q1[t] + P[i]
        for i in range(t + 1, 256):
            q2[t] = q2[t] + P[i]
        for i in range(t + 1):
            if q1[t] > 0:
                mu1[t] = mu1[t] + i * P[i] / q1[t]
        for i in range(t + 1, 256):
            if q2[t] > 0:
                mu2[t] = mu2[t] + i * P[i] / q2[t]
        for i in range(t + 1):
            if q1[t] > 0:
                sigma1[t] = sigma1[t] + (i - mu1[t]) ** 2 * P[i] / q1[t]
        for i in range(t + 1, 256):
            if q2[t] > 0:
                sigma2[t] = sigma2[t] + (i - mu2[t]) ** 2 * P[i] / q2[t]

    for t in range(256):
        sigmamu[t] = q1[t] * sigma1[t] + q2[t] * sigma2[t]
```

```
    minSigma = np.inf
    minT = 0
    for i in range(256):
        if minSigma > sigmamu[i]:
            minT = i
            minSigma = sigmamu[i]

    for i in range(H):
        for j in range(W):
            if dst[i, j] > minT:
                output[i, j] = 255
            else:
                output[i, j] = 0
    return output

src = cv.imread("rice.bmp",cv.IMREAD_GRAYSCALE) #Read the file
opened = Opening(src, 7)

H,W = src.shape[:]
dst = np.zeros((H, W), src.dtype)

for i in range(H):
    for j in range(W):
        diff = np.int32(src[i, j]) - np.int32(opened[i, j])
        if   diff < 0:
            diff = 0
        dst[i, j] = diff

output = binary(dst)
cv.imshow("TopHat", output)
cv.imwrite("./TopHat.bmp", output)
cv.waitKey(0)
```