01 Matlab Code

1. Reverse Function

설명 : 행렬의 좌우 반전을 도와주는 함수

```
function y = reverse(x) %Make the right and left side of matrix reversed
[m, n] = size(x);
for i = 1:m
    for j = 1:fix(n/2)
        S1 = x(i, j);
        S2 = x(i, n-j+1);
        y(i, j) = S2;
        y(i, n-j+1) = S1;
    end
end
end
```

2. Right Shift Function

설명 : 행렬의 값을 a 만큼 오른쪽으로 이동하도록 도와주는 함수

```
function y = right_shift(x, a) %Shift the Matrix x a space to the right
[m, n] = size(x);
for i = 1:m
    for j = 1:n-a
      y(i, j+a) = x(i, j);
    end
end
end
```

3. Main Code

```
%%% Robot Vision%%%
%%% Dept. of Electronic Engineering
%%% 201314651 Lee Wonjai



F = im2double(imread('C:\Users\user\OneDrive\¹ÙÅÁ È-¸é\4ÇÐ³â
2ÇÐ±â\·Î°¿°ñÀü\Original Images\dipum_images_ch02\Fig0206(a)(rose-
original).tif')); % read the targeted image

imdata = imfinfo('C:\Users\user\OneDrive\¹ÙÅÁ È-¸é\4ÇÐ³â
```

```
2ÇÐ±â\·Î°¿°ñÀü\Original Images\dipum_images_ch02\Fig0206(a)(rose-
original).tif'); % read the precise imformation of image
% extract the information of image from imdata
fprintf('Format of the image = %s\n',imdata.Format)
fprintf('Size of the image = %d\n',imdata.FileSize)
fprintf('Compression of the image = %s\n',imdata.Compression)
fprintf('BPP of the image = %d\n',imdata.BitsPerSample)

S = F(300:340, 300:340); % cut the image
S = im2uint8(S);
figure('Name', 'Cutted','NumberTitle','off'), imshow(S);

F = im2uint8(F); % change a matrix F into an uint8 format

R = reverse(F); % reverse the left and right side of image
figure('Name', 'Reversed','NumberTitle','off'), imshow(R)

SD = F.*0.8; % blur the image
figure('Name', 'Blurred','NumberTitle','off'), imshow(SD)

SR = right_shift(F, 3); % shift the image to the right side,
right_shift(Matrix, number wanted to be shifted)
SB = F-SR; % Subtract an original image F to shifted image SR
figure('Name', 'Subtracted','NumberTitle','off'), imshow(SB)
```

02 Result 1

1. Information of the image

Format of the image = tif

Size of the image = 1049798

Compression of the image = Uncompressed

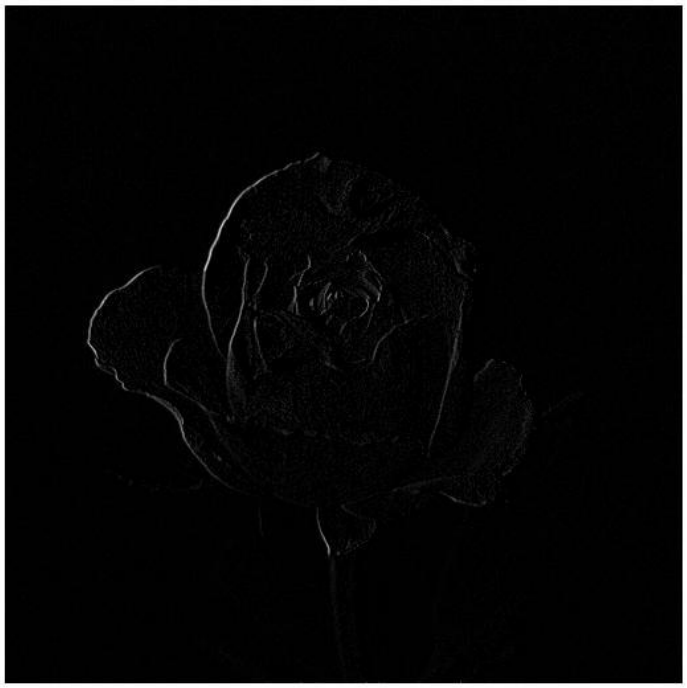
BPP of the image = 8

2. Cutted

3. Reversed

4. Fainted

5. Outline



02 심화학습(python 구현)

```python
import numpy as np
import cv2
from PIL import Image

def reverse(Matrix1):
    M1, N1 = Matrix1.shape
    A1 = np.uint8(np.zeros((M1,N1)))
    for i in range(M1):
        for j in range(N1//2):
            S1 = Matrix1[i, j]
            S2 = Matrix1[i, N1-j-1]
            A1[i, N1-j-1] = S1
            A1[i, j] = S2
    return A1

def  right_shift(Matrix2, num):
    M2, N2 = Matrix2.shape
    A2 = np.uint8(np.zeros((M2,N2)))
    for k in range(M2):
        for l in range(N2-num-1,-1,-1):
            A2[k, l+num] = Matrix2[k, l]
    return A2

def minus_flow(Matrix3,Matrix4): #because if some value suffer overflow or underflow, it designate worng value, I use function the classify the case
    M3, N3 = Matrix3.shape
    A3 = np.uint8(np.zeros((M3,N3)))
    for m in range(M3):
        for n in range(N3):
            if Matrix3[m, n] < Matrix4[m, n]:
                A3[m, n] = 0
                continue
            A3[m, n] = Matrix3[m, n] - Matrix4[m, n]
    return A3

F = cv2.imread('Fig0206(a)(rose-original).tif',cv2.IMREAD_GRAYSCALE)
```

```
S = F[299:340, 299:340] #also can be done by using or operation between F and filter imgae(299~339 white, others black)
R = reverse(F)
SD = cv2.multiply(F,0.8) #If i just muliply it, type of array and its element is not matched. That's why I use cvfunction
SR = right_shift(F,3)
SB = minus_flow(F,SR)

row, col = F.shape

img = Image.open('Fig0206(a)(rose-original).tif') #image information
print(img.format)
print(img.size)
print(img.filename)


cv2.imshow('Rose', F)
cv2.imshow('Cutted', S)
```

```
cv2.imshow('Reversed',R)
cv2.imshow('Blurred', SD)
cv2.imshow('Outline', SB)

cv2.imwrite('Rose.tif', F) #store the chaged image
cv2.imwrite('Cutted.tif', S)
cv2.imwrite('Reversed.tif',R)
cv2.imwrite('Blurred.tif', SD)
cv2.imwrite('Outline.tif', SB)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

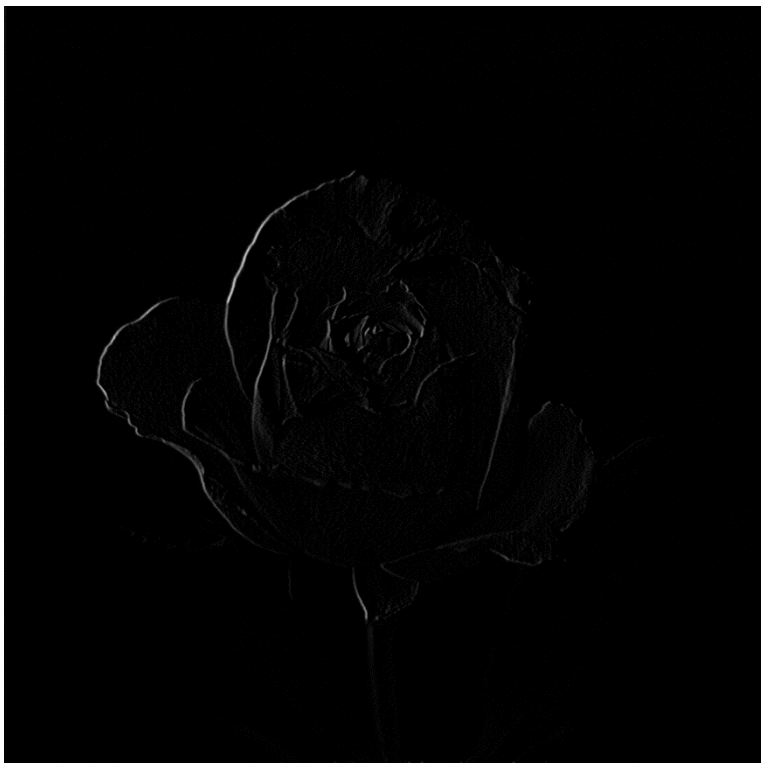
Result 2

1. Cutted

2. Reversed

3. Fainted

4. Outlilne



Conclusion

1. Matlab

- Cutting의 경우 행렬의 요소 일부분을 추출하여 구현하는 것 뿐 아니라, cutting 하려는 부분만 white 처리 나머지 부분은 black으로 처리된 행렬과 or 연산을 하여 수행하는 것도 가능 하다.,(Filtering)

  Ex) A = zeros(1024,1024); A(300:340,300:340) = 255; S = A|F

- 실제 카메라 어플리케이션 사용시 좌우 반전하면 이미지가 깨지는 현상이 있기에, matlab에서도 이 현상을 예상했다. 그러나 예상과 달리 별 문제없이 좌우 반전이 잘 되었다.

- SR의 경우 3pixel 정도밖에 이동했기에 육안으로 이동 여부를 확인하지 못했으나, 이를 SB를 통해 동작 여부 확인 가능

- SB의 경우 물체의 outline을 따는데 도움이 됨 -> convolution 영상 처리 기법과 관련이 있을 것으로 예상

2. Python

- Python의 경우 영상처리 library를 import 해야함. 대표적으로 사용하는 library는 opencv와 pillow이다.

- 일반적으로 pillow가 많이 쓰이는 이유는 함수들이 사용하기 간편하기 때문이다. 그러나 opencv의 영상처리 속도가 pillow보다 빠르며, 이는 많은 이미지 데이터를 처리할 때 큰 차이를 보인다. 따라서 개발 툴로는 opencv를 더 많이 쓴다. (Numpy의 경우 행렬 계산 편리를 위해 사용)

- 이번 과제에서 pillow library를 사용한 이유는 opencv에는 이미지 정보를 불러오는 내장 함수가 따로 존재하지 않기 때문이다. -> 파이썬 툴의 약점이며, 내 지식의 한계이기도 하다.

- SD에서 cv2.multipier 함수를 사용한 이유: Python에서 uint8 행렬에 단순히 float형을 곱하면 행렬 계산은 가능하나, 행렬의 데이터 타입과 요소 값의 데이터 타입이 맞지 않아 이미지 출력할 때 에러가 발생한다.

- SB에서 함수를 만들어 사용한 이유 : Python과 Matlab의 차이 때문인데, Matlab에서는 pixel 값이 0이하로 떨어지거나 255이상으로 올라가도 0 또는 255값을 잡아준다. 그러나 Python에서는 이러한 기능이 없기에 underflow와 overflow 현상이 발생한다. 따라서 if문을 써서 flow 현상을 제어했다.