▪ Homework #4: 2021학년 1학기

교과목명	디지털영상처리II	수업주차	7주차
이름	황진주	학번	20193148

▣ 주제

원과 선을 포함한 사진을 하나 선택한다. 에지 검출(LoG, DoG, 케니 에지 등) 또는 이진화 과정(임계화, 오츄 알고리즘 등)을 거친 후,

하프 변환 기법을 이용하여 선과 원을 검출하는 결과를 만들어 보시오.

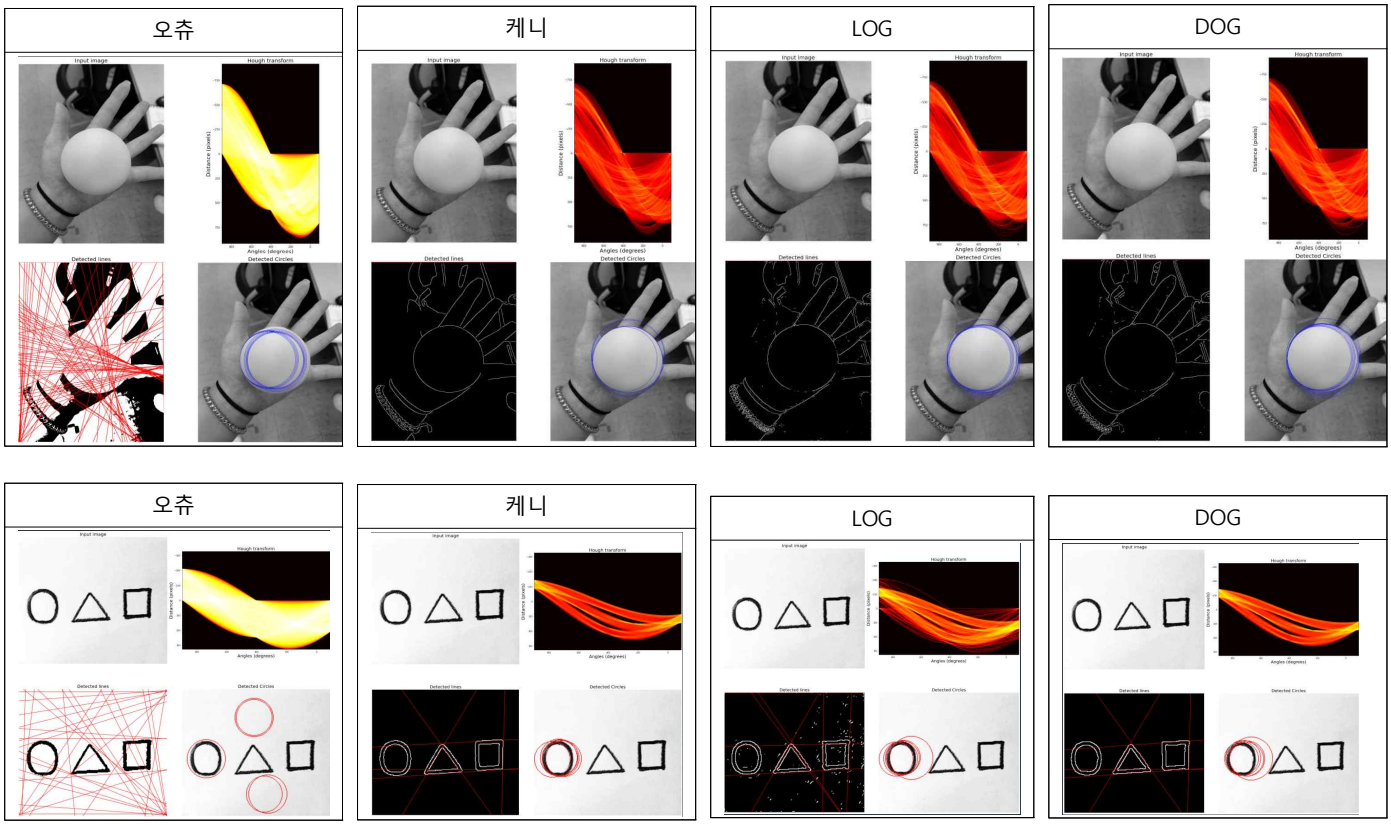
결과 1) 실행 파이썬 코드

```
import cv2
from matplotlib import pylab as pylab
import numpy as np
from skimage.transform import (hough_line, hough_line_peaks, hough_circle,
         hough_circle_peaks)
from skimage.draw import circle_perimeter
from skimage.io import imread, imsave
from skimage.filters import sobel, threshold_otsu
from skimage.color import rgb2gray, gray2rgb, label2rgb
from skimage import filters, feature, img_as_float, exposure, img_as_ubyte



import matplotlib.pyplot as plt
from matplotlib import cm

#%%
def zero_cross(image, T=0):
    zimg = np.zeros(image.shape)
    # 커널이 넘치지 않게 하려고 -1만큼 까지만 돌게됨
    for i in range(0, image.shape[0]-1):
        for j in range(0, image.shape[1]-1):
            # if문 끝에 \는 라인 엔터치려고 넣은거임

            # 오른쪽
            if image[i][j]*image[i+1][j] < 0 \
            and np.abs( image[i+1][j] - image[i][j]) > T :
                    zimg[i,j] = 1
            # 우상단
            elif image[i][j]*image[i+1][j+1] < 0 \
            and np.abs( image[i+1][j+1] - image[i][j]) > T :
                    zimg[i,j] = 1
            # 위
            elif image[i][j]*image[i][j+1] < 0 \
            and np.abs( image[i][j+1] - image[i][j]) > T :
                    zimg[i,j] = 1
    return zimg
#%% 오츄 알고리즘



image = rgb2gray(imread('../images/8.jpg'))

# 오츄 알고리즘 적용
# thresh = threshold_otsu(image)
# binary = image > (thresh * 1.2)

# LoG
#im_g = filters.gaussian(image, 2)
binary = binary.astype('bool')

# Classic straight-line Hough transform
h, theta, d = hough_line(binary)
print(h)
print(theta)
print(d)
#print(*hough_line_peaks(h, theta, d))

# Generating figure 1
fig, axes = plt.subplots(2, 2, figsize=(20, 20))
ax = axes.ravel()



hough_radii = np.arange(30, 80, 1)
hough_res = hough_circle(binary, hough_radii)

ax[0].imshow(image, cmap=cm.gray)
ax[0].set_title('Input image', size=20)
ax[0].set_axis_off()

ax[1].imshow(np.log(1 + h),
             extent=[10*np.rad2deg(theta[-1]),          np.rad2deg(theta[0]),
          d[-1], d[0]],
             cmap=cm.hot, aspect=1/1.5)
ax[1].set_title('Hough transform', size=20)
ax[1].set_xlabel('Angles (degrees)', size=20)
ax[1].set_ylabel('Distance (pixels)', size=20)
ax[1].axis('image')

ax[2].imshow(binary, cmap=cm.gray)
for _, angle, dist in zip(*hough_line_peaks(h, theta, d)):
    y0 = (dist - 0 * np.cos(angle)) / np.sin(angle)
    y1 = (dist - binary.shape[1] * np.cos(angle)) / np.sin(angle)
    ax[2].plot((0, image.shape[1]), (y0, y1), '-r')
ax[2].set_xlim((0, image.shape[1]))
ax[2].set_ylim((image.shape[0], 0))
ax[2].set_axis_off()
ax[2].set_title('Detected lines', size=20)



# Select the most prominent 5 circles
accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii,
         total_num_peaks=2)
```

```
#im_l = filters.laplace(im_g)
#binary = zero_cross(im_l, T=np.max(im_l)*0.015)

# DoG
im_dog = filters.difference_of_gaussians(image, 2.0)
binary = zero_cross(im_dog, T=np.max(im_dog)*0.04)

# 캐니
#binary = feature.canny(image, sigma=2)
```

```
image = gray2rgb(image)
for center_y, center_x, radius in zip(cy, cx, radii):
    circy, circx = circle_perimeter(center_y, center_x, radius)
    image[circy, circx] = (0.9, 0.1, 0.1)

ax[3].imshow(image, cmap=plt.cm.gray)
ax[3].set_axis_off()
ax[3].set_title('Detected Circles', size=20)

print("end")
plt.tight_layout()
plt.show()
```

결과 2) 선 또는 원 검출 결과



결과 3) 결과를 얻기 위한 방법 및 어려웠던 점 서술

실현상황

하나의 이미지를 이용하는 것이었지만 결과검출에 문제가 있어, 여러 가지 경우를 적용하였다.

오류상황

1) 오츄에 한해 **라인 검출**이 잘 이루어지지 않았다.
2) 케니, LOG, DOG를 적용한 특정 이미지는 **라인 검출 시 결과가 나오지 않았**다.
   - hough_line_peaks의 결과가 오츄에서는 여러 값이 나왔지만 나머지는 값이 하나만 도출됨
   - 오츄와의 차이점 : 이미지 변환값이 오츄는 bool값이지만 나머지는 int값이어서 bool로 통일
   - 오츄와의 유사점 : 연산 후의 값의 범위 및 형태
3) ㅇ△ㅁ 이미지의 원 검출 시 arange 변화량 값을 1로 두지 않으면 **인덱스 초과 오류**가 발생한다.

1. 오츄 결과도출법

변경값)

- 바이너리 판단을 위한 **thresh 값**을 기존값에서 1.2를 곱한 값으로 변경
  => 노이즈 없는 더욱 깔끔한 변화
- 다른 기법과 달리 **arange(15, 35, 1)**의 작은 범위의 설정
- 결과가 매끄럽지 않아 total_num_peaks의 속성을 **5가지의 결과**로 상향 조정

결과)

- 허프 변환 : 다른 기법에 비해 더욱 많고 넓은 범위의 선 발생
- 직선 검출 : 직선의 구분이 잘 이루어지지 않고, 난잡한 선의 발생
- 원 검출 : 다른 기법에 비해 더 낮은 검출 정확도를 보임

2. 케니 결과도출법

변경값)

- 시그마값을 2로 설정

결과)

- 허프 변환 : 3가지의 **명확**하고 진한 선의 영역으로 구분됨
- 직선 검출 : 삼각형 **세 면**에 대한 직선 검출 사각형의 상/하/우 **세 면**의 직선 검출
- 원 검출 : 5개의 원의 범위가 크게 벗어나지 않음

3. LOG 결과도출법

변경값)

- T의 기준값에 **0.015**를 곱함 : 선이 명확하면서 노이즈가 제일 적은 결과 도출

결과)

- 허프 변환 : 3가지의 **명확**하고 진한 선의 영역에 **잡음**이 들어간 듯한 모습을 보임
- 직선 검출 : 삼각형의 **세 면**, 사각형의 **네 면**이 모두 검출됨
- 원 검출 : 원 중앙을 기준으로 원이 형성, 추출을 2개 할 시 명확하지만 5개로 늘인 후 튀는 값 발생

4. DOG 결과도출법

변경값)

- T의 기준값에 **0.04**를 곱함 : 노이즈가 없는 명확한 라인 검출

결과)

- 허프 변환 : 3가지의 **명확**하고 진한 선의 영역으로 구분됨
- 직선 검출 : 삼각형 **세 면**에 대한 직선 검출 사각형의 상/하/우 **세 면**의 직선 검출
- 원 검출 : 원의 우측으로 원이 형성, 추출을 2개 할 시 명확하지만 점점 우측으로 커지는 원 발생