2020년 컴퓨터그래픽스

- HW 03 -

제출일자	2020.10.06.
이 름	장수훈
학 번	201402414
분 반	00

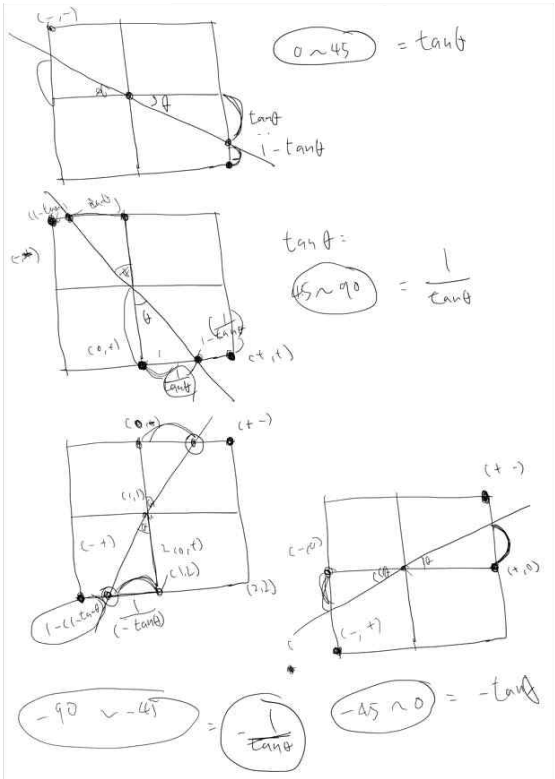
구현코드

```python
def calcMagnitude(Ix, Iy):
    ########################################
    # TODO                                 #
    # calcMagnitude 완성                    #
    # magnitude : ix와 iy의 magnitude       #
    ########################################
    # Ix와 Iy의 magnitude를 계산
    temp = Ix**2 + Iy**2
    magnitude = np.sqrt(temp)

    return magnitude


# Ix와 Iy의 angle을 구함
def calcAngle(Ix, Iy):
    ######################################
    # TODO                               #
    # calcAngle 완성                      #
    # angle     : ix와 iy의 angle         #
    ######################################
    radian_angle = np.arctan(Iy/Ix)
    angle = np.rad2deg(radian_angle)

    return angle
```

이론 ppt에 나온 그대로 구현을 하였다. angle은 밑에 함수를 보아하니 각도로 리턴을 해야할 것 같아서 rad2deg로 변환하여 리턴하였다.

```python
if 0 <= degree and degree < 45:
    rate = np.tan(np.deg2rad(degree))
    # ********************
    left_magnitude = (rate) * magnitude[row - 1, col - 1] + (1 - rate) * magnitude[row, col - 1]
    right_magnitude = (rate) * magnitude[row + 1, col + 1] + (1 - rate) * magnitude[row, col + 1]
    if magnitude[row, col] == max(left_magnitude, magnitude[row, col], right_magnitude):
        larger_magnitude[row, col] = magnitude[row, col]

elif -45 > degree and degree >= -90:
    rate = -(1/np.tan(np.deg2rad(degree)))
    up_magnitude = (rate) * magnitude[row - 1, col + 1] + (1 - rate) * magnitude[row - 1, col]
    down_magnitude = (rate) * magnitude[row + 1, col - 1] + (1 - rate) * magnitude[row + 1, col]
    if magnitude[row, col] == max(up_magnitude, magnitude[row, col], down_magnitude):
        larger_magnitude[row, col] = magnitude[row, col]

elif -45 <= degree and degree < 0:
    rate = -np.tan(np.deg2rad(degree))
    left_magnitude = (rate) * magnitude[row + 1, col - 1] + (1 - rate) * magnitude[row, col - 1]
    right_magnitude = (rate) * magnitude[row - 1, col + 1] + (1 - rate) * magnitude[row, col + 1]
    if magnitude[row, col] == max(left_magnitude, magnitude[row, col], right_magnitude):
        larger_magnitude[row, col] = magnitude[row, col]

elif 90 >= degree and degree >= 45:
    rate = 1/np.tan(np.deg2rad(degree))
    up_magnitude = (rate) * magnitude[row - 1, col - 1] + (1 - rate) * magnitude[row - 1, col]
    down_magnitude = (rate) * magnitude[row + 1, col + 1] + (1 - rate) * magnitude[row + 1, col]
    if magnitude[row, col] == max(up_magnitude, magnitude[row, col], down_magnitude):
        larger_magnitude[row, col] = magnitude[row, col]
```

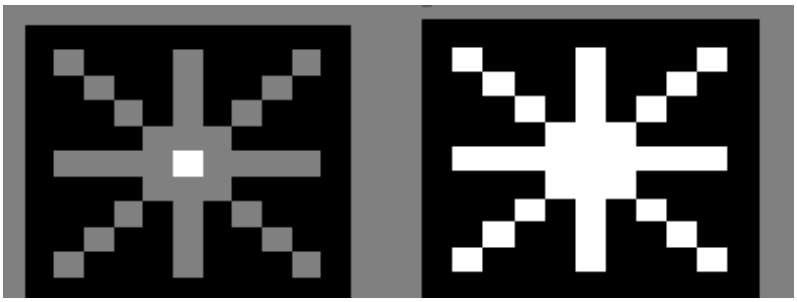
non_max 함수이다. 이건 말로 설명하기가 매우 힘들어서 사진을 첨부하겠다.



```
            else:
                ################################################
                # TODO                                         #
                # high 보다는 작고 low보다는 큰 경우              #
                ################################################
                dst[row, col] = 127

    high_edge = np.sum(_ == 255)

    while (1):
        mask(dst)
        if (high_edge == np.sum(dst == 255)):
            break
        high_edge = np.sum(dst == 255)

    for row in range(h):
        for col in range(w):
            if (dst[row, col] == 127):
                dst[row, col] = 0

    return dst

def mask(src) :
    (h, w) = src.shape
    for i in range(h) :
        for j in range(w) :
            if(src[i, j] == 127) :
                t_max = max(src[i-1, j-1], src[i-1, j], src[i-1, j+1], src[i, j-1],
                            src[i, j+1], src[i+1, j-1], src[i+1, j], src[i+1, j+1])
                if(t_max == 255) :
                    src[i, j] = 255
```

애매한 앳지일 경우 127로 선언해두고 mask함수를 돌아 강한앳지쪽에 연결된 애매한 앳지를 강하게 만들어주고, 약한 앳지를 지우는 형식으로 구현하였다.

결과값




original
after non maximum supression
before non maximum supression
tx
ty
my canny edge detection

느낀점

애매한 앳지일 때 버려야하는가 강하게 만들어야하는가를 정하는 알고리즘을 생각하는게 매우 어려웠다.
또한 기본적으로 주어진 코드의 설명을 조금 더 자세히 해주었으면 좋겠다고 생각했다.

난이도

이번과제 또한 저번과제랑 같이 어려웠다...