

HW 03 – REPORT

소속 : 정보컴퓨터공학부

학번 : 201924548

이름 : 이풍헌

1. 서론

실습 목표 및 이론적 배경 기술 (1~2페이지)

이번 실습에서는 canny edge detection을 실습하고자 한다. Canny edge detection은 이미지에서 밝기나 색상의 급격한 변화를 나타내는 edge를 찾아는 데 사용된다. Edge로 생성된 이미지로 이미지 대상의 윤곽이나 형태를 파악할 수 있다.

실습 순서는 다음과 같다.

1. Hw2에서 만든 gaussian filter를 통해 이미지를 blur처리한다.
2. 이미지에 sobel operator를 적용해 gradient와 theta를 구한다.
3. Non maximum suppression을 통해 gradient 방향으로 local maximum인지 체크한다.
4. Double threshold를 통해 threshold 사이에 있으면 weak, high threshold보다 크면 strong으로 표시한다.
5. Hysteresis를 통해 strong에 연결된 weak를 strong으로 변경해준다.

2. 본론

실습 내용 및 결과 기술 (2페이지 이상)

1. HW2에서 만든 Gaussian filter를 통해 이미지를 그레이 스케일로 blur처리 한다.

```
def reduce_noise(img):  
    """ Return the gray scale gaussian filtered image with sigma=1.6  
    Args:  
        img: RGB image. Numpy array of shape (H, W, 3).  
    Returns:  
        res: gray scale gaussian filtered image (H, W).  
    """  
  
    #implement  
    image = img.convert('L')#그레이스케일로 변환  
    image = np.asarray(image).astype(np.float32)  
    res = gaussconvolve2d(image, 1.6)#sigma 1.6으로 gaussian filter적용  
    res = np.clip(res, 0, 255) #0~255범위로 조정  
  
    return res
```



gray scale blurred image

2. Sobel filter를 x, y 에 각각 적용해 gradient I_x, I_y 를 구한다



x-axis gradient image



y-axis gradient image

$G(\text{magnitude}) = \sqrt{I_x^2 + I_y^2}$, $\text{Theta} = \arctan(I_y / I_x)$ 로 구할 수 있다.



gradient magnitude image



direction of gradient image

3. Non max suppression을 통해 theta에 따라 G의 local maximum을 찾아준다

```
def non_max_suppression(G, theta):
    """ Performs non-maximum suppression.
    This function performs non-maximum suppression along the direction
    of gradient (theta) on the gradient magnitude image (G).
    Args:
        G: gradient magnitude image with shape of (H, W).
        theta: direction of gradients with shape of (H, W).
    Returns:
        res: non-maxima suppressed image.
    """

    H, W = G.shape
    res = np.zeros((H, W)) #0으로 된 빈 맵
    angle = theta * 180 / np.pi # radian을 호도법으로 변경 변환
    #-22.5~22.5 오른쪽 -180~-157.5 , 157.5~180 가로방향
    #22.5~67.5 오른쪽 위 -112.5~-157.5 왼쪽 아래
    #67.5~112.5 위 -67.5~-112.5 아래
    #-22.5~-67.5 오른쪽 아래 112.5~157.5 왼쪽 위
    for i in range(1, H - 1):
        for j in range(1, W - 1):
            # 가로
            if (-22.5 < angle[i, j] < 22.5) or (-180 <= angle[i, j] <= -157.5) or (157.5 < angle[i, j] <= 180):
                left = G[i, j - 1]
                right = G[i, j + 1]

            # / 대각선
            elif 22.5 <= angle[i, j] < 67.5 or -157.5 <= angle[i, j] < -112.5:
                left = G[i - 1, j + 1]
                right = G[i + 1, j - 1]

            # 수직
            elif 67.5 <= angle[i, j] < 112.5 or -112.5 <= angle[i, j] < -67.5:
                left = G[i - 1, j] #위
                right = G[i + 1, j] #아래

            # \ 대각선
            elif 112.5 <= angle[i, j] < 157.5 or -67.5 < angle[i, j] <= -22.5 :
                left = G[i - 1, j - 1]
                right = G[i + 1, j + 1]

            if (G[i, j] > left) and (G[i, j] > right): # 좌우보다 더 크면 저장 local maxima
                res[i, j] = G[i, j]

    return res
```



NMS image

4. Double threshold를 통해 불필요한 pixel을 제거하고 weak pixel 과 strong pixel로 나눈다.

```
def double_thresholding(img):
    """
    Args:
        img: numpy array of shape (H, W) representing NMS edge response.
    Returns:
        res: double_thresholded image.
    """
    #implement
    diff = np.max(img) - np.min(img) #threshold 생성
    thresholdH = np.min(img) + diff * 0.15
    thresholdL = np.min(img) + diff * 0.03

    weak = 80
    strong = 255

    res = np.zeros(img.shape) #0으로 채워진 맵 생성
    #np.where(조건, 참일 때 반환값, 거짓일 때 반환값)
    res = np.where((thresholdL <= img) & (img < thresholdH), np.ones(img.shape) * weak, res) # thresholds 사이면 weak
    res = np.where(thresholdH <= img, np.ones(img.shape) * strong, res) # high threshold 이상이면 strong
    return res
```

Np.where 을 사용해 for문 없이 정리할 수 있다.



double threshold image

5. Hysteresis를 통해 strong pixel에 대해 dfs탐색을 하며 strong pixel에 연결된 weak pixel들을 strong pixel로 변환한다.

```
def hysteresis(img):
    """ Find weak edges connected to strong edges and link them.
    Iterate over each pixel in strong_edges and perform depth first
    search across the connected pixels in weak_edges to link them.
    Here we consider a pixel (a, b) is connected to a pixel (c, d)
    if (a, b) is one of the eight neighboring pixels of (c, d).
    Args:
        img: numpy array of shape (H, W) representing NMS edge response.
    Returns:
        res: hysteresised image.
    """
    #implement
    H, W = img.shape
    res = np.zeros((H, W))

    for i in range(1, H - 1):
        for j in range(1, W - 1): # 테두리를 제외한 모든 픽셀
            if (img[i][j] == 255): # strong pixel이면
                dfs(img, res, i, j) # dfs 수행
    return res
```



hysteresis image

3. 결론

토의 및 결론 (1페이지)

이미지에서 의미 있는 정보를 추출하기 위해 edge를 검출하였다.

Canny edge detection은

1. Noise reduction
2. Gradient, theta 얻기
3. Non maximum suppression처리
4. Double threshold 적용
5. Hysteresis을 통한 pixel linking 과정을 통해 적용할 수 있다.

