HW 03 – REPORT

소속 :

학번 :

이름 :

1. 서론

실습 목표 및 이론적 배경 기술 (1~2페이지)

이번 실습 목표는 파이썬으로 Canny edge detector를 만드는 것이다.

일반적으로 사진에 나타나는 물체의 경계에는 명암, 컬러, 텍스쳐 등의 급격한 변화가 일어난다.

이러한 급격한 변화가 나타나는 곳을 확인하기 위해 미분이 사용된다. 디지털 사진은 연속 공간이 아니라 이산 공간이므로, 다음 식을 통해 도함수를 구할 수 있다.

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} (1.1)$$

만약 어떤 한 픽셀에서 그 이웃 픽셀로의 변화를 계산하려면 다음과 같이 계산할 수 있다.

$$f'(x) = \frac{f(x+1) - f(x)}{1} = f(x+1) - f(x) \ (1.2)$$

이는 이미지 각 픽셀에 (-1,1)의 마스크를 convolution하는 것과 같다.

이를 2차원 공간으로 확장하면 x방향은 마스크$[-1 \quad 0 \quad 1]^T$, y방향은 마스크 $[-1 \quad 0 \quad 1]$ 을 적용하여 급격한 변화가 일어나는 곳을 구할 수 있다.

그러나 일반적으로, 디지털 사진의 잡음 때문에 edge가 아닌 부분에서도 많은 변화가 나타나 위의 마스크로는 잘 검출할 수 없고, 아래 소벨 연산자가 많이 쓰인다.

$$s_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}, s_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \ (1.3)$$

위의 소벨 연산자를 각 픽셀에 적용시키면, 각 픽셀의 y방향, x방향의 변화값(각각 d_y, d_x)이 나온다. 이러한 변화는 magnitude와 direction이 있는 벡터로 표현될 수 있으며 다음 식으로 구할 수 있다.

$$magnitude = \sqrt{d_y^2 + d_x^2} \ (1.4)$$

$$direction = \tan^{-1}\frac{d_y}{d_x} \ (1.5)$$

이때 방향은 0-2π 의 값을 가지는데, 일반적으로 어떤 한 픽셀의 이웃 픽셀은 $(0, \frac{1}{4}\pi, \frac{1}{2}\pi, \frac{3}{4}\pi, \pi, \frac{5}{4}\pi, \frac{3}{2}\pi, \frac{7}{4}\pi, 2\pi)$ 의 8방향에 있으므로, 이들 값으로 양자화된다.

캐니 에지 연산자는 1986년에 캐니가 발표한 edge 검출 방법이며, 좋은 edge 검출 알고리즘의 3가지 조건을 제시하였다.

1. 오류율 최소 : false positive와 false negative가 최소여야된다.
2. 위치 : 검출된 edge가 실제 edge의 위치와 가까워야한다.

3. 두께 : edge는 한 두께여야 한다.

먼저 두께를 줄이는 방법을 알아보자. 일반적으로, 디지털 사진에 잡음이 있기 때문에 이를 제거하고 edge를 검출한다. 그러나 이 과정에서 edge의 폭도 넓어진다. 두께를 줄이기 위해서는 보통 어떤 픽셀의 값과 그 픽셀의 direction 방향(식 1.5 참고)및 그 반대 방향의 픽셀 값을 비교해 작으면, edge가 아닌 것으로 간주한다. 이를 non-maximum suppression이라고 한다.

그리고 edge를 검출하면, edge가 아닌 것도 많이 검출된다. 오류율을 최소로 하기 위해, 임계값 T_{low}와 T_{high}를 두어 T_{high}이상의 강도를 가진 edge를 strong edge로 하고, T_{low} 이상 T_{high}미만의 강도를 가진 edge를 weak edge로 하며, T_{low} 미만의 강도를 가진 edge는 edge로 간주하지 않는다. 그리고 strong edge와 연결되어있는 weak edge는 모두 strong edge로 바꾸고, 남아있는 weak edge를 edge로 간주하지 않으면 canny edge detect가 완료된다. 이 마지막 과정을 hysteresis thresholding이라고 한다.

2. 본론

실습 내용 및 결과 기술 (2페이지 이상)

(1) Noise reduction

이구아나 이미지를 grayscale 로 변경하고, gaussian filter를 적용하여 smoothing 연산을 수행한다. 여기서 사용한 함수는 이전 HW2에서 구현한 함수며, 이전 HW2에서 문제가 있었던 부분을 보완하였다(자료형 부분을 보완하였으며, overflow를 방지하기 위해 이미지를 numpy array로 변경할 때 타입을 np.float32로 변경하였다)

Figure 1 원본 이구아나 이미지(좌)와 gaussian filter를 적용한 이미지(우)

(2) Finding the intensity gradient of the image

Sobel filter를 (1)에서 smoothing 연산을 수행하여 나온 이미지에 적용하여, x방향, y방향 변화값을 계산한다. 그리고 식 1.4와 1.5를 이용하여 magnitude와 direction을 계산한다. 이때 magnitude는 0~255 값으로 mapping 한다.

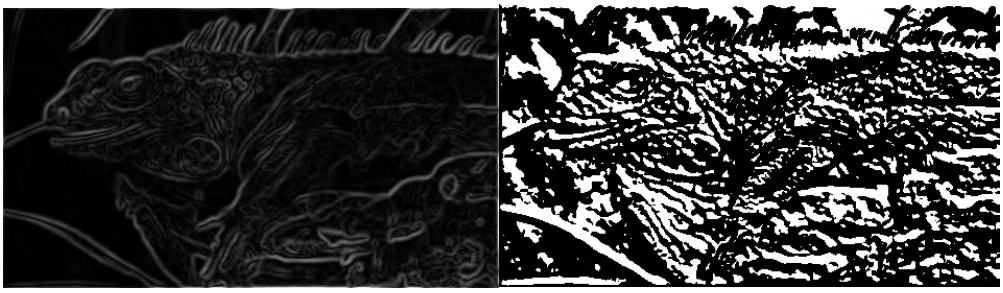

Figure 2 이미지 gradient의 magnitude(좌)와 direction(우)

(3) Non-Maximum Suppression

앞서 smoothing 연산 때문에 edge가 두꺼워졌으므로, 이를 작게 한다. 한 픽셀의 값과 그 픽셀의 gradient direction 및 그 반대 방향의 값을 비교하여 magnitude가 작으면 해당 픽셀의 magnitude를 0으로 하여, edge가 아니라고 간주한다. 이때 direction은 양자화 한다.

Figure 3 Non-Maximum Suppression을 적용한 이미지.

(4) Double thresholding

hysteresis thresholding을 위해 edge를 strong edge와 weak edge로 분류한다. 분류하는 기준인 T_{low}와 T_{high} 는 다음과 같이 계산한다.

$$diff = \max(image) - \min(image)$$

$$T_{low} = \min(image) + diff \times 0.15$$

$$T_{high} = \min(image) + diff \times 0.03$$

Visualizing을 위해 strong edge는 픽셀 값 255로, weak edge는 80으로 표시한다.

Figure 4 Double thresholding을 적용한 이미지. Strong edge는 255, weak edge는 80의 명암으로 표시된다.

(5) Edge Tracking by hysteresis

Double thresholding을 적용한 이미지에 hysteresis를 수행한다. 이를 위해 DFS를 이용하여, strong edge에서 시작하여 그 주변의 weak edge를 탐색하고 해당 weak edge를 strong edge로 바꾸고, strong edge와 연결되지 않은 weak edge는 무시한다.

Figure 5 Hystresis thresholding을 적용한 최종 이미지. Strong edge만 남았다.

3. 결론

토의 및 결론 (1페이지)

파이썬으로 canny edge detector를 만들어보았다. 이를 위해 이미지에서 edge를 검출하고, non-maximum인 부분을 없애고, edge를 strong edge와 weak edge로 나누며 strong edge와 연결된

weak edge는 strong edge로 바꾸고, 그렇지 않은 edge는 edge에서 제외하였다.

Sobel filter를 이용하여 convolution을 통해 edge를 검출할 수 있음을 알게되었으며, Canny edge detector를 이용하면 edge를 더 정확하게 검출할 수 있음을 알게되었다. 이미지에서 어떠한 것을 검출하는 것은 AI 등 복잡한 과정이 필요할 줄 알았는데 행렬 연산 등을 통하여 edge등을 검출할 수 있음을 알게되었다. 또한 HW3 수행과정에서 HW2에서의 실수(이미지를 numpy array로 변경 후 계산과정에서 overflow)를 방지하기 위해 numpy array의 타입을 np.float32로 변경하였으며, 이 과정에서 자료형의 중요성을 알게 되었다.

참고 자료

- 오일석, '컴퓨터 비전', 한빛아카데미, 2014
- https://numpy.org/doc/stable/reference/index.html