REPORT

Computer Vision HW#1
Sobel Edge Detector

2020.03.30

1. 과제 목표

- 본 과제의 목표는 Sobel Edge Detector를 통해 이미지의 edge를 검출하는 방법과 원리를 학습하고, Python라이브러리 OpenCV를 사용하여 실제 이미지에서 edge를 검출해 보는 것입니다.

2. 과제 내용 (코드 및 알고리즘 설명)

- 전체 코드

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os

path = os.getcwd()
img = cv2.imread(path + '/lane_image.jpg', cv2.IMREAD_GRAYSCALE)
sobelx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) #sobelx
sobely = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]]) #sobely

img_sobelx = cv2.filter2D(img, cv2.CV_64F, sobelx)
img_sobely = cv2.filter2D(img, cv2.CV_64F, sobely)

img_mag = np.sqrt(img_sobelx**2 + img_sobely**2)
img_ang = np.arctan2(img_sobely, img_sobelx)

plt.subplot(2, 2, 1)
plt.imshow(img_sobelx, cmap='gray')
plt.title('sobelx')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 2)
plt.imshow(img_sobely, cmap='gray')
plt.title('sobely')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 3)
plt.imshow(img_mag, cmap='gray')
plt.title('magnitude')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 4)
plt.imshow(img_ang, cmap='gray')
plt.title('angle')
```

```
plt.xticks([]), plt.yticks([])

plt.show()
```

- 코드에 대한 설명

  코드는 이미지 load & 파라미터 선언부, 이미지 처리, plot 생성 세 부분으로 작성되어 있습니다.

  - 이미지 load & 파라미터 선언 파트

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os

path = os.getcwd()
img = cv2.imread(path + '/lane_image.jpg', cv2.IMREAD_GRAYSCALE)
sobelx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) #sobelx
sobely = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]]) #sobely
```

  이미지 처리 라이브러리로 유명한 OpenCV, 이미지 plot 작성에 편리한 matplotlib.pyplot, numpy array 편집에 유용한 함수를 지원하는 numpy, 그리고 파일의 경로 작성을 위한 os 라이브러리를 import 해 주었습니다. os.getcwd()함수를 이용해 이미지의 경로가 자동으로 같은 폴더 내의 lane_image.jpg 파일을 load하도록 작성했습니다.

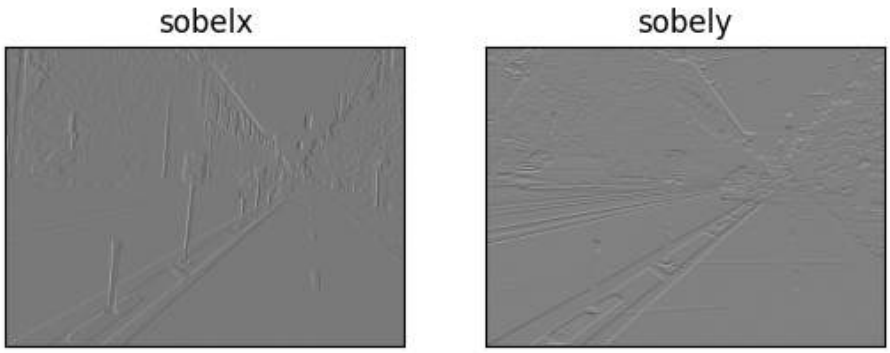
  sobelx와 sobely는 Sobel Edge Detector의 x성분과 y성분 필터입니다. Sobel 필터란 인접 픽셀값들의 차이를 통해 기울기, 즉 미분값을 나타내는 미분 연산자입니다. 추후 cv2.filter2D 함수에 사용하기 위해 각각 $\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}, \begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$을 저장했습니다.

  - 이미지 처리 파트

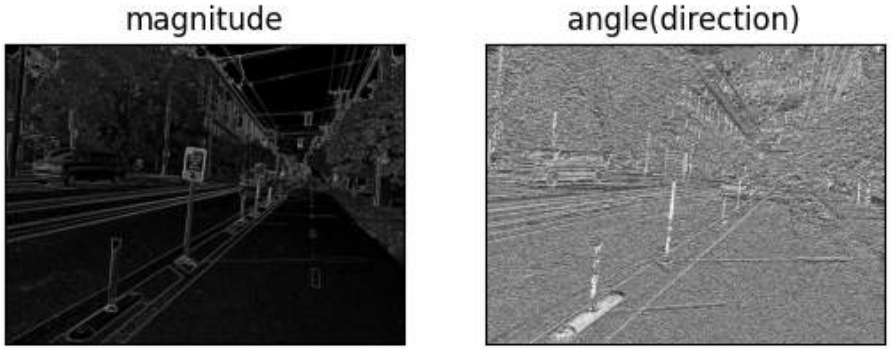
```
img_sobelx = cv2.filter2D(img, cv2.CV_64F, sobelx)
img_sobely = cv2.filter2D(img, cv2.CV_64F, sobely)

img_mag = np.sqrt(img_sobelx**2 + img_sobely**2)
img_ang = np.arctan2(img_sobely, img_sobelx)
```

  cv2.filter2D 함수를 사용하여 손쉽게 sobelx, sobely 필터를 적용할 수 있습니다. 또한 필터를 적용한 이미지의 데이터 타입을 CV_64F, 즉 float64로 변환합니다. 그 이유는 데이터 타입으로 인한 데이터 손실을 막기 위함입니다. 원본 이미지의 데이터 타입은 uint8로, 0~255 사이의 양수값을 가집니다. 이때 filter2D의 출력은 양의 기울기를 취할 때 양수가, 음의 기울기를 취할 때 음수가 출력되는데, 데이터 타입이 uint8이라면 음수값을 저장할 수 없으므로 음수를 0으로 저장합니다. 따라서 데이터 타입을 음수도 저장할 수

있는 float형으로 변환하여 저장합니다. float32와 float64 둘 모두 상관없습니다. 이때의 출력인 img_sobelx, img_sobely는 다음과 같습니다.



출력된 이미지를 관찰하면 img_sobelx는 세로 edge가 강조된 것을, img_sobely는 가로 edge가 강조된 것을 볼 수 있습니다. img_sobelx와 img_sobely를 구했다면, 이 두 이미지값을 적용한 sobel magnitude 값과 angle(direction)값을 구할 수 있습니다. Magnitude 값은 $Mag = \sqrt{Sobelx^2 + Sobely^2}$ 과 같습니다. angle값은 $ang = \arctan(\frac{Sobely}{Sobelx})$과 같습니다. 이 수식을 적용한 결과물은 다음과 같습니다.

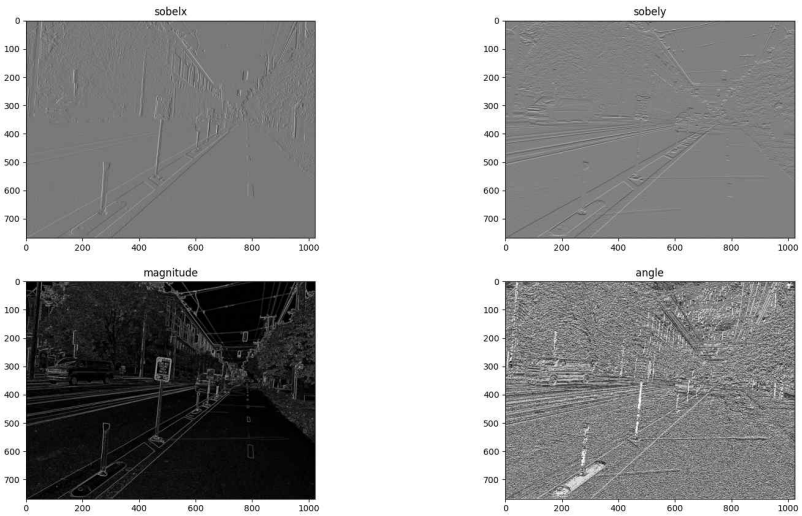


Magnitude와 angle은 Sobelx와 Sobely값을 조합하여 추출할 수 있는 결과값입니다. 위의 두 이미지 중에서 magnitude가 edge가 더 잘 검출되었으므로, 이 이미지를 활용할 예정이 있다면 magnitude를 사용하는 편이 좋습니다.

- Plot 생성 파트

```
plt.subplot(2, 2, 1)
plt.imshow(img_sobelx, cmap='gray')
plt.title('sobelx')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 2)
plt.imshow(img_sobely, cmap='gray')
plt.title('sobely')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 3)
plt.imshow(img_mag, cmap='gray')
plt.title('magnitude')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 4)
plt.imshow(img_ang, cmap='gray')
plt.title('angle')
plt.xticks([]), plt.yticks([])

plt.show()
```

이미지를 보기 편하게 Plot 형태로 출력하는 부분입니다. img_sobelx, im_sobely, magnitude, angle 네 이미지를 matplotlib.pyplot 라이브러리를 사용하여 출력합니다. 결과물은 다음과 같습니다.



3. 과제에서 배운 것 / 어려웠던 점

이번 과제에서는 이미지 처리 라이브러리 OpenCV를 사용하여 Sobel Edge Detector를 통해 edge를 검출하는 방법을 배웠습니다. 이미지에서 edge를 검출하는 원리가 미분 연산자라는 것과 이를 어떤 방식으로 조합하여 더 좋은 결과를 도출하는 방법 또한 알 수 있었습니다. 또한 앞으로 사용할 OpenCV 라이브러리에 조금이나마 익숙해지는 기회가 되었습니다.

이번 과제에서 어려웠던 점은 데이터 타입을 왜 변경해 주는지 이해할 수 없었던 점입니다. 원본 이미지에 필터를 적용할 때 원본과 같은 데이터 타입을 사용하면 출력 결과가 제대로 나오지 않거나 오류가 발생했습니다. 하지만 구글링을 통해 데이터 타입에 따른 데이터 손실이 발생한다는 것을 알게 되어 이미지를 float64형으로 고쳐서 사용할 수 있었습니다.