텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.  
위의 수업 자료에 있는 코드를 바탕으로

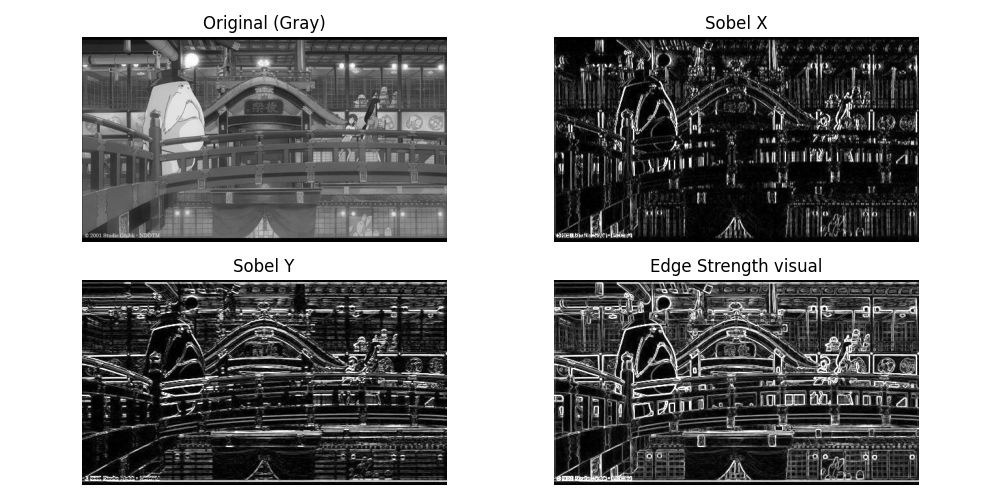
소벨 에지 검출 및 결과 시각화.py를 만들었다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

cv.Sobel()을 사용하여 X축(cv.CV\_64F, 1, 0)과 Y축(cv.CV\_64F, 0, 1)으로 초기화하는 것만 주의하면 연산자 적용까지는 같고.

cv.magnitude(grad\_x, grad\_y)를 통해 에지 강도를 계산하여 이를 출력하는 점이 차이점.

시각화도 plt를 통해서 진행했다.

2.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

수업 자료에 나온 코드를 바탕으로

2.캐니 에지 및 허프 변환을 이용한 직선 검출.py를 작성했다.  
텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

Canny의 경우 수업 중에 나오는 함수이며 파라미터만 바꿔주면 구현이 완료된다.

cv.HoughLinesP()의 경우를 잘 봐야하는데   
일단 함수 실행 결과는 직선의 시작 및 끝점 2차원 배열 리스트를 반환해준다.

cv.HoughLinesP(image, rho, theta, threshold, minLineLength=0, maxLineGap=0)

함수의 형태이며 각 파라미터는 아래의 의미를 가진다.

image 에지 이미지

rho 거리 해상도 (픽셀 단위). 예: 1이면 1픽셀 간격으로 r값 측정

theta 각도 해상도 (라디안 단위). 보통 np.pi / 180 → 1도 단위

threshold 한 직선으로 인정되기 위한 최소 누적값 (투표 수). 커질수록 엄격해짐

minLineLength 검출할 직선의 최소 길이. 이보다 짧으면 무시됨

maxLineGap 선분이 이 정도 거리까지 끊겨 있어도 한 직선으로 간주  
  
이후 cv.HoughLinesP()로 출력된 함수를 cv.line 함수 cv.line(img, pt1, pt2, color, 이하 파라미터 생략)으로 그려주고 plt로 시각화를 완료했다.

3.

배우지 않은 함수에 대해 코드를 작성하므로 함수들의 파라미터 및 반환값을 잘 알아야 한다.  
3.GrabCut을 이용한 대화식 영역 분할 및 객체 추출.py

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

img = cv.imread('tower.jpg')

if img is None:

print("이미지를 불러올 수 없습니다.")

exit()

mask = np.zeros(img.shape[:2], np.uint8)

bgdModel = np.zeros((1, 65), np.float64)

fgdModel = np.zeros((1, 65), np.float64)

x = int(input("사각형의 x 좌표를 입력하세요: "))

y = int(input("사각형의 y 좌표를 입력하세요: "))

width = int(input("사각형의 width를 입력하세요: "))

height = int(input("사각형의 height를 입력하세요: "))

rect = (x, y, width, height)

cv.grabCut(img, mask, rect, bgdModel, fgdModel, 5, cv.GC\_INIT\_WITH\_RECT)

mask2 = np.where((mask == cv.GC\_BGD) | (mask == cv.GC\_PR\_BGD), 0, 1).astype('uint8')

result = img \* mask2[:, :, np.newaxis]

img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)

mask\_rgb = mask2 \* 255

result\_rgb = cv.cvtColor(result, cv.COLOR\_BGR2RGB)

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)

plt.title("Original Image")

plt.imshow(img\_rgb)

plt.axis('off')

plt.subplot(1, 3, 2)

plt.title("Mask Image")

plt.imshow(mask\_rgb, cmap='gray')

plt.axis('off')

plt.subplot(1, 3, 3)

plt.title("Background Removed")

plt.imshow(result\_rgb)

plt.axis('off')

plt.tight\_layout()

plt.show()

본문에서 사용자 입력을 받는 방법에 대해서 정의하지 않았으므로 터미널을 통해서 입력을 받았다.

cv.grabCut()함수는 다음과 같이 생겼으며 이미지의 mask를 업데이트 하는 함수라 반환은 하지 않는다.  
cv.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount, mode)

각 파라미터는 다음 주의 사항을 가진다.

Img: 컬러로 사용한다.

Mask : 이미지의 픽셀에 정보를 이 배열에 부여하는 방식이므로 mask는 이미지와 같은 크기로 초기화 해준다.

Rect 초기 전경이 포함된 사각형으로 이 사각형 안은 전경으로 추정하고, 바깥은 배경으로 간주. 사용하려면 mode를 cv.GC\_INIT\_WITH\_RECT로 설정해야 한다.

bgdModel / fgdModel

np.zeros((1, 65), np.float64)로 제시한대로 초기화 하면 된다.

iterCount

하이퍼 파라미터로 5회로 설정했다.  
  
이후 mask는 cv.GC\_BGD, cv.GC\_FGD, cv.GC\_PR\_BGD, cv.GC\_PR\_FGD

값 중 하나를 가지게 되는데 각각 전경 확정, 배경 확정, 전경 추정, 배경 추정으로 두번째와 네번째 값을 0 아닌 값을 1로 설정하고

result = img \* mask2[:, :, np.newaxis]

차원을 맞춰주기 위해서 np.newaxis 함수를 사용하여 차원을 맞춰준다음 곱하면 된다.

이후 plt를 통해 시각화 한다.