영상정보처리 11주차 과제 템플리트

이름: 이현정
학번: 32203660

구글 드라이브 마우팅 및 작업 경로로 이동

- 다음 쉘에 필요한 작업을 하시오.

```
from google.colab import drive
drive.mount('/gdrive')

%cd /gdrive/MyDrive/ImageProcClass/Notebook-Week11

# matplot color display
def show_with_matplotlib_jh(img, title):
    if img is None:
      print("show_with_matplotlib: Could not read the image.")
      return

    if img.shape[2] != 3:
      print()
      print("show_with_matplotlib: given image does not contains 3 channels")
      return

    # Convert BGR image to RGB:
    img_RGB = img[:, :, ::-1]

    # Show the image using matplotlib:
    plt.imshow(img_RGB)
    plt.title(title)
    plt.show()

# matplot grayscale display
def show_with_matplotlib_gray_jh(img, title):
   if img is None:
      print("show_with_matplotlib_gray: Could not read the image.")
      return

   if img.ndim > 2:
      print()
      print("show_with_matplotlib: given image has more than 2 dim")
      return

   plt.imshow(img, cmap="gray")
   plt.title(title)
   plt.show()

def show_with_matplotlib_M04(color_img, title, pos, axis_show):
```

```
    # Convert BGR image to RGB
    img_RGB = color_img[:, :, ::-1]

    ax = plt.subplot(3, 6, pos)
    plt.imshow(img_RGB)
    plt.title(title)

    if not axis_show:
      plt.axis('off')

    plt.show()
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive",
/gdrive/MyDrive/ImageProcClass/Notebook-Week11
```

```
image_path_airplane = '../Dongkeun-OpenCV-ImgData/airplane_bw.png'
image_path_horse = '../Dongkeun-OpenCV-ImgData/horse_bw.png'
```

다음 두 개의 이미지에 대해 스켈레톤을 구하는 프로세스를 작성하고, 결과를 가시화하시오.

입력 이미지 - 이미지 폴더에 없는 경우, 첨부된 이미지를 다운받아 폴더에 넣고 실행하기

- airplane_bw.png
- horse_bw.png

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

def skeleton(path): # 면적 없이 형태만 볼때
  src = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
  ret, A = cv2.threshold(src, 128, 255, cv2.THRESH_BINARY)
  skel_dst = np.zeros(src.shape, np.uint8) # 검정 이미지
  B= cv2.getStructuringElement(shape=cv2.MORPH_CROSS, ksize=(3,3))

  done = True
  while done:
      erode = cv2.erode(A, B) # 작은 영역을 줄임
      opening = cv2.morphologyEx(erode, cv2.MORPH_OPEN, B) # 열림 연산
      tmp = cv2.subtract(erode, opening) # 줄어든 영역
      skel_dst = cv2.bitwise_or(skel_dst, tmp) # 계속 물체의 흰색 안쪽으로 이동되는 양을 누적
      A = erode.copy()
      done = cv2.countNonZero(A) != 0 # 0이 아닌 요소의 개수 반환

  plt.imshow(src, cmap='gray')
  plt.show()
  plt.imshow(skel_dst, cmap='gray')
  plt.show()

print('airplane')
skeleton(image_path_airplane)
```

```
print('skeleton')
skeleton(image_path_horse)
```

문제 2

"2021-1 ImgProc JB-CH07-JHU2104-V1.pdf" 에서 저자 구현 코드와 opencv 함수를 이용하는 방법 둘 다 이용해서 예시를 보여주고 있습니다. 저자 구현 코드와 opencv 를 이용한 방법의 결과를 디스플레이하고, 두 결과를 픽셀 단위로 비교하여 몇 개의 픽셀이 다른 지 계산하고, 픽셀이 다른 경우, 다른 부분만을 영상을 만들어 디스플레이 하시오.

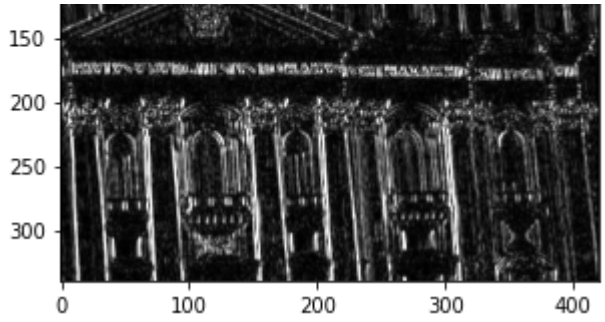
- 필요한 이미지는 '../Dongkeun-OpenCV-ImgData' 에 복사하여 넣어서 수행

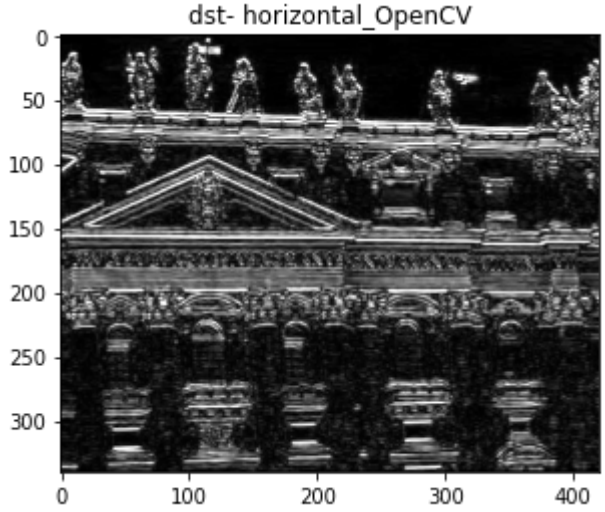
1. 예제 7.2.5 (소벨 엣지 검출)
2. 예제 7.2.6 (라플라시안 엣지 검출)
3. 예제 7.2.8 (캐니 엣지 검출)

1. 예제 7.2.5 (소벨 엣지 검출)

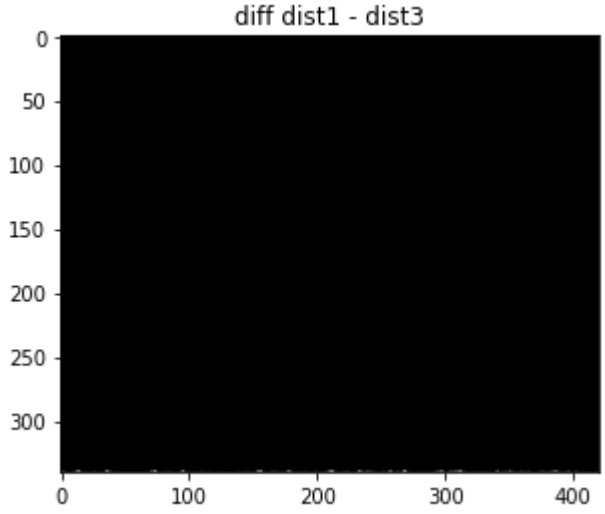

```
# 저자 구현
import numpy as np, cv2

# 회선 수행 함수 - 행렬 처리 방식(속도 면에서 유리)
def filter(image, mask):
    rows, cols = image.shape[:2]
    dst = np.zeros((rows, cols), np.float32)                # 회선 결과 저장 행렬
    xcenter, ycenter = mask.shape[1]//2, mask.shape[0]//2  # 마스크 중심 좌표

    for i in range(ycenter, rows - ycenter):                 # 입력 행렬 반복 순회
        for j in range(xcenter, cols - xcenter):
            y1, y2 = i - ycenter, i + ycenter + 1             # 관심영역 높이 범위
            x1, x2 = j - xcenter, j + xcenter + 1             # 관심영역 너비 범위
            roi = image[y1:y2, x1:x2].astype("float32")       # 관심영역 형변환

            tmp = cv2.multiply(roi, mask)                     # 회선 적용 - OpenCV 곱셈
            dst[i, j] = cv2.sumElems(tmp)[0]                  # 출력화소 저장
    return dst

def differential(image, data1, data2):
    mask1 = np.array(data1, np.float32).reshape(3, 3)
    mask2 = np.array(data2, np.float32).reshape(3, 3)

    dst1 = filter(image, mask1)                    # 사용자 정의 회선 함수
    dst2 = filter(image, mask2)
    dst = cv2.magnitude(dst1, dst2)                # 회선 결과 두 행렬의 크기 계산

    dst = cv2.convertScaleAbs(dst)                     # 윈도우 표시 위해 OpenCV 함수로 형변환 및
    dst1 = cv2.convertScaleAbs(dst1)
    dst2 = cv2.convertScaleAbs(dst2)
```

```
    return dst, dst1, dst2

image = cv2.imread("../Dongkeun-OpenCV-ImgData/edge.jpg", cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception("영상파일 읽기 오류")

data1 = [-1, 0, 1, # 수직 마스크
         -2, 0, 2,
         -1, 0, 1]
data2 = [-1,-2,-1, # 수평 마스크
          0, 0, 0,
          1, 2, 1]
dst, dst1, dst2 = differential(image, data1, data2)     # 두 방향 회선 및 크기(에지 강도) 계산


# opencv
dst3 = cv2.Sobel(np.float32(image), cv2.CV_32F, 1, 0, 3)  # x방향 미분 - 수직 마스크
dst4 = cv2.Sobel(np.float32(image), cv2.CV_32F, 0, 1, 3)  # y방향 미분 - 수평 마스크
dst3 = cv2.convertScaleAbs(dst3) # 절댓값 및 uint8 형변환
dst4 = cv2.convertScaleAbs(dst4)



# 비교
show_with_matplotlib_gray_jh(dst, "dst- sobel edge")
show_with_matplotlib_gray_jh(dst1, "dst- vertical_mas")
show_with_matplotlib_gray_jh(dst2, "dst- horizontal_mask")
show_with_matplotlib_gray_jh(dst3, "dst- vertical_OpenCV")
show_with_matplotlib_gray_jh(dst4, "dst- horizontal_OpenCV")

result = 0
rx, ry = image.shape[:2]
for x in range(rx):
  for y in range(ry):
    if dst1[x,y] != dst3[x,y]:
      result += 1
print("dst1 - dst3 총",rx*ry, "개 중 다른 픽셀의 개수 :", result)
if (result != 0) :
  diff = cv2.bitwise_xor(dst1, dst3)
  show_with_matplotlib_gray_jh(diff, "diff dist1 - dist3")


result = 0
rx, ry = image.shape[:2]
for x in range(rx):
  for y in range(ry):
    if dst2[x,y] != dst4[x,y]:
      result += 1
print("dst2 - dst4 총",rx*ry, "개 중 다른 픽셀의 개수 :", result)
if (result != 0) :
  diff = cv2.bitwise_xor(dst2, dst4)
  show_with_matplotlib_gray_jh(diff, "diff dist1 - dist3")
```

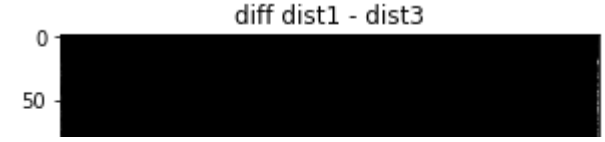




dst1 - dst3 총 142800 개 중 다른 픽셀의 개수 : 582



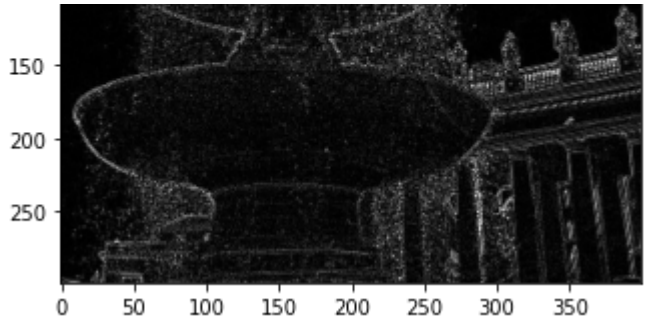
dst2 - dst4 총 142800 개 중 다른 픽셀의 개수 : 653

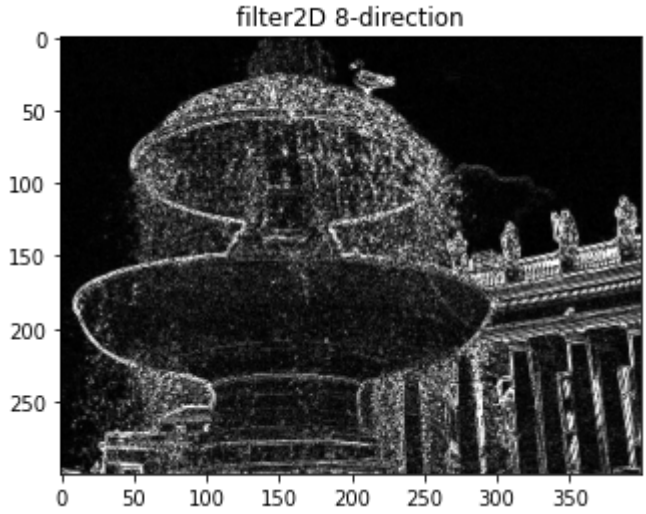


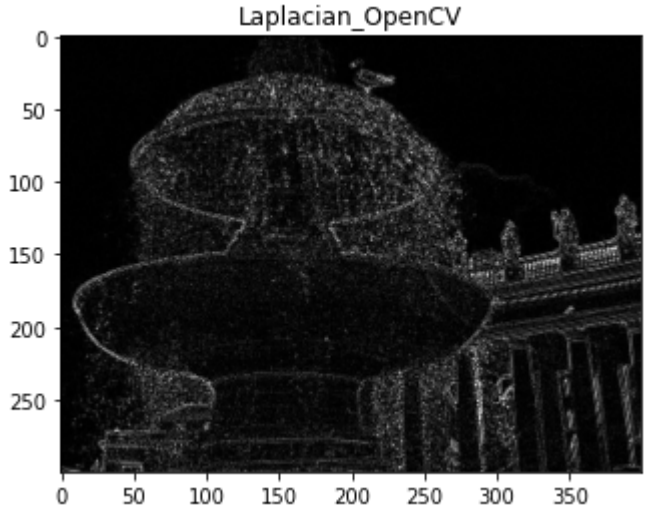
2. 예제 7.2.6 (라플라시안 엣지 검출)

```
# 저자 구현
import numpy as np, cv2

image = cv2.imread("../Dongkeun-OpenCV-ImgData/laplacian.jpg", cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception("영상파일 읽기 오류")

data1 = [   [0,     1,      0], # 4 방향 마스크
            [1,     -4,     1],
```

```
            [0,     1,      0]]
data2 = [   [-1,    -1,     -1], # 8 방향 마스크
            [-1,    8,      -1],
            [-1,    -1,     -1]]
mask4 = np.array(data1, np.int16) # 음수가 있으므로 자료형이 int8인 행렬 선언
mask8 = np.array(data2, np.int16)

# OpenCV 함수 cv2.filter2D() 통한 라플라시안 수행
dst1 = cv2.filter2D(image, cv2.CV_16S, mask4)
dst2 = cv2.filter2D(image, cv2.CV_16S, mask8)



# opencv
dst3 = cv2.Laplacian(image, cv2.CV_16S, 1) # OpenCV를 이용한 라플라시안 수행 함수



# 비교
show_with_matplotlib_gray_jh(image, "image")
show_with_matplotlib_gray_jh(cv2.convertScaleAbs(dst1), "filter2D 4-direction")
show_with_matplotlib_gray_jh(cv2.convertScaleAbs(dst2), "filter2D 8-direction")
show_with_matplotlib_gray_jh(cv2.convertScaleAbs(dst3), "Laplacian_OpenCV")

result = 0
rx, ry = image.shape[:2]
for x in range(rx):
  for y in range(ry):
    if cv2.convertScaleAbs(dst1)[x,y] != cv2.convertScaleAbs(dst3)[x,y]:
      result += 1
print("filter2D 4-direction - Laplacian_OpenCV 총",rx*ry, "개 중 다른 픽셀의 개수 :", result)
if (result != 0) :
  diff = cv2.bitwise_xor(cv2.convertScaleAbs(dst1), cv2.convertScaleAbs(dst3))
  show_with_matplotlib_gray_jh(diff, "diff filter2D 4-direction - Laplacian_OpenCV")

result = 0
rx, ry = image.shape[:2]
for x in range(rx):
  for y in range(ry):
    if cv2.convertScaleAbs(dst2)[x,y] != cv2.convertScaleAbs(dst3)[x,y]:
      result += 1
print("filter2D 8-direction - Laplacian_OpenCV 총",rx*ry, "개 중 다른 픽셀의 개수 :", result)
if (result != 0) :
  diff = cv2.bitwise_xor(cv2.convertScaleAbs(dst2), cv2.convertScaleAbs(dst3))
  show_with_matplotlib_gray_jh(diff, "diff filter2D 8-direction - Laplacian_OpenCV")
```







filter2D 4-direction - Laplacian_OpenCV 총 120000 개 중 다른 픽셀의 가
filter2D 8-direction - Laplacian_OpenCV 총 120000 개 중 다른 픽셀의 가

3. 예제 7.2.8 (캐니 엣지 검출)

```
# 저자 구현
import numpy as np, cv2

def nonmax_suppression(sobel, direct):
    rows, cols = sobel.shape[:2]
    dst = np.zeros((rows, cols), np.float32)  # 행렬
    for i in range(1, rows - 1):
        for j in range(1, cols - 1): # 이웃 화소 가져오기
            values = sobel[i-1:i+2, j-1:j+2].flatten() # 3x3으로 9개 화소
            first = [3, 0, 1, 2] # 처음으로 이웃한 좌표 : 0 45 90 135도 순
            id = first[direct[i, j]] # 기울기 방향에 따라 2개 이웃 화소 선택
            v1, v2 = values[id], values[8-id]

            ## if 문으로 이웃 화소 가져오기
            # if direct[i, j] == 0: # 기울기 방향 0도
            # v1, v2 = sobel[i, j-1], sobel[i, j+1]
            # if direct[i, j] == 1: # 기울기 방향 45도
            # v1, v2 = sobel[i-1, j-1], sobel[i+1, j+1]
            # if direct[i, j] == 2: # 기울기 방향 90도
            # v1, v2 = sobel[i-1, j], sobel[i+1, j]
            # if direct[i, j] == 3 # 기울기 방향 135도
            # v1, v2 = sobel[i+1, j-1], sobel[i-1, j+1]

            dst[i, j] = sobel[i, j] if (v1 < sobel[i , j] > v2) else 0 # 이웃 화소 2개 보다 중심 화
    return dst

def trace(max_sobel, i, j, low): # 에지 추적 함수
    h, w = max_sobel.shape
    if (0 <= i < h and 0 <= j < w) == False: return # 추적 화소 범위 확인 : 범위 밖 i or j 종료
    if pos_ck[i, j] == 0 and max_sobel[i, j] > low: # 추적 조건 확인
        pos_ck[i, j] = 255 # 완료된 좌표
        canny[i, j] = 255 # 에지 지정

        trace(max_sobel, i - 1, j - 1, low) # 추적 함수 재귀 호출 - 8방향 추적
        trace(max_sobel, i    , j - 1, low)
        trace(max_sobel, i + 1, j - 1, low)
        trace(max_sobel, i - 1, j    , low)
        trace(max_sobel, i + 1, j    , low)
        trace(max_sobel, i - 1, j + 1, low)
        trace(max_sobel, i    , j + 1, low)
        trace(max_sobel, i + 1, j + 1, low)
```

```
def hysteresis_th(max_sobel, low, high): # 이력 임계값 수행 함수
    rows, cols = max_sobel.shape[:2]
    for i in range(1, rows - 1):  # 에지 영상 순회
        for j in range(1, cols - 1):
            if max_sobel[i, j] > high:  trace(max_sobel, i, j, low)  # high보다 큰 경우(높은 임계값

image = cv2.imread("../Dongkeun-OpenCV-ImgData/canny.jpg", cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception("영상 파일 읽기 오류")

pos_ck = np.zeros(image.shape[:2], np.uint8) # 추적 완료 점검
canny = np.zeros(image.shape[:2], np.uint8) # 케니 에지

# 사용자 정의 캐니 에지
gaus_img = cv2.GaussianBlur(image, (5, 5), 0.3) # 1. 블러링으로 노이즈 제거 -> 가우시안 블러링
Gx = cv2.Sobel(np.float32(gaus_img), cv2.CV_32F, 1, 0, 3)  # x방향 마스크
Gy = cv2.Sobel(np.float32(gaus_img), cv2.CV_32F, 0, 1, 3)  # y방향 마스크
sobel = np.fabs(Gx) + np.fabs(Gy)  # 두 행렬 절댓값 덧셈 / 2. 화소 기울기 강도와 방향(0,45,90,135)
# sobel = cv2.magnitude(Gx, Gy) # 두 행렬 벡터 크기

directs = cv2.phase(Gx, Gy) / (np.pi / 4)
directs = directs.astype(int) % 4
max_sobel = nonmax_suppression(sobel, directs)   # 3. 비최대치 억제
hysteresis_th(max_sobel, 100, 150) # 이력 임계값 -> 4. 이력 임계값으로 에지 결정
# canny에 엣지를 지정함



# opencv
canny2 = cv2.Canny(image, 100, 150) # OpenCV 캐니 에지



# 비교
show_with_matplotlib_gray_jh(image, "image")
show_with_matplotlib_gray_jh(canny,"canny")                 # 사용자 정의 캐니
show_with_matplotlib_gray_jh(canny2, "OpenCV_Canny")           # OpenCV 캐니 에지

result = 0
rx, ry = image.shape[:2]
for x in range(rx):
  for y in range(ry):
    if canny[x,y] != canny2[x,y]:
      result += 1
print("총",rx*ry, "개 중 다른 픽셀의 개수 :", result)
if (result != 0) :
  diff = cv2.bitwise_xor(canny, canny2)
  show_with_matplotlib_gray_jh(diff, "diff")
```