REPORT

Assignment3

강의: 비주얼 컴퓨팅

김동준 교수님

2018204042 배홍섭

목 차

1.원본 이미지

set_tree1.jpg

set_tree2.jpg

set_tree3.jpg

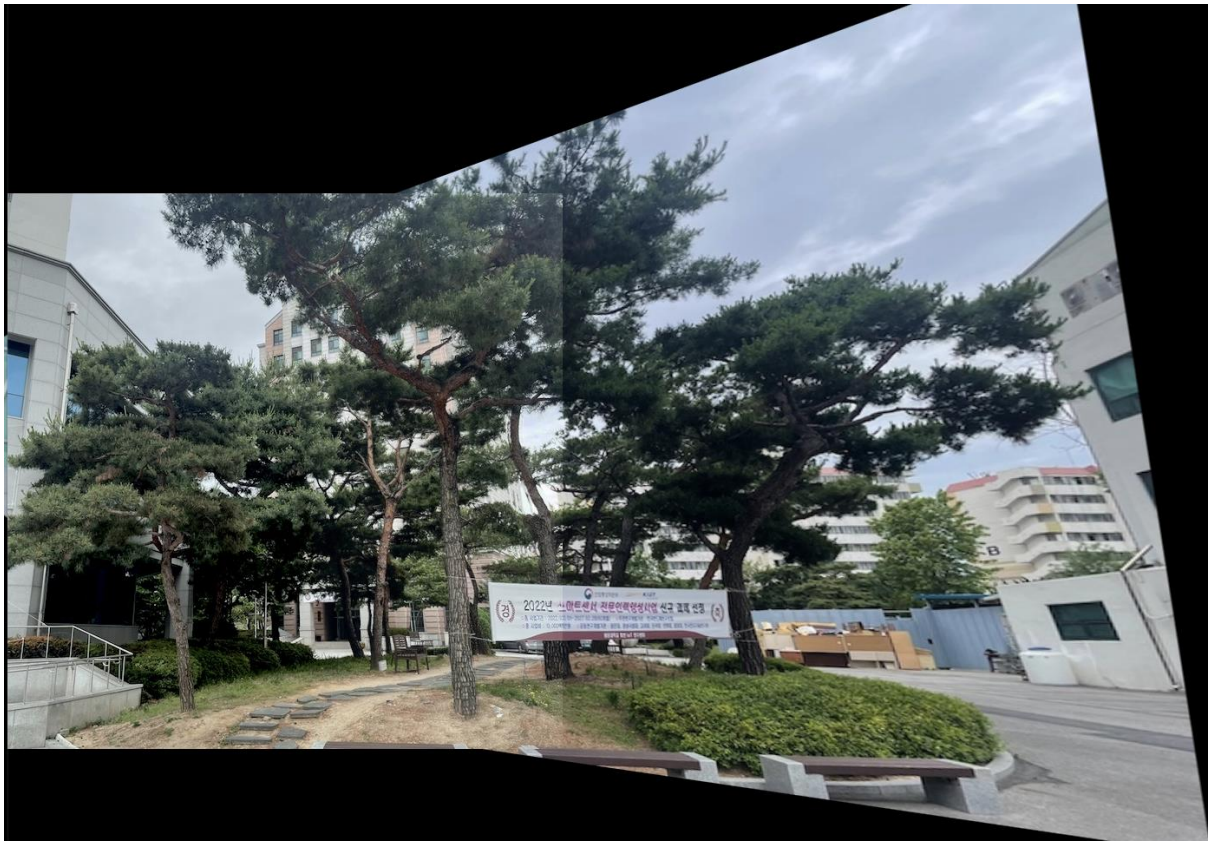
1) 좌우 회전 포함

   1 번 사진(가장 좌측) 회전 확인가능

2) 밝기 대비 차이 확인

   1 번 사진(가장 좌측)은 해를 등지고 찍었고, 3 번 사진(가장 우측)은 해가 떠있는 방향으로 찍었으므로 밝기가 다르다. 이는 두개만 Stitch 를 진행한 아래 이미지로 확인할 수 있다.

3) 영상에서 겹치는 부분 확인



ㄴ 2) 밝기 차이가 있음을 확인함과 동시에 3)영상에서 겹치는 부분도 있음을 확인할 수 있다.

2.두번째 영상을 중심으로 Stitching

```
img1 = cv.imread('set_tree1.jpg')
img2 = cv.imread('set_tree2.jpg')
img3 = cv.imread('set_tree3.jpg')
```

ㄴ Stitching 이전 2 번 영상을 기준으로 왼쪽에 붙을 영상을 img1, 오른쪽에 붙을 영상을 img3 으로 변수 설정

- 우선 2 번 영상을 중심으로 1 번 영상을 Stitching 진행

(1 번 영상은 회전이 심하게 이루어져 있어 2-3 번 Stitching 을 먼저 진행했다.)

```
else:
    result = cv.warpPerspective(img3, H,(math.ceil(max(conerX))  ,  math.ceil(max(conerY))))

mtrx = np.float32([[1, 0, 0],
                   [0, 1, 0]])
img2 = cv.warpAffine(img2, mtrx, (result.shape[1], result.shape[0]) )

resultN = cv.addWeighted(result, 0.5, img2, 0.5,0)

for i in range((resultN.shape[0] * resultN.shape[1])):
    x = (i) % (resultN.shape[1]) # 지금 진행하는 픽셀의 행 번호
    y = (i) // resultN.shape[1] # 지금 진행한는 픽셀의 열 번호
    if(sum(resultN[y][x]) < sum(img2[y][x])):
        resultN[y][x] = img2[y][x]

for i in range((resultN.shape[0] * resultN.shape[1])):
    x = (i) % (resultN.shape[1]) # 지금 진행하는 픽셀의 행 번호
    y = (i) // resultN.shape[1] # 지금 진행한는 픽셀의 열 번호
    if(sum(resultN[y][x]) < sum(result[y][x])):
        resultN[y][x] = result[y][x]


beforeWidth = max(conerX)

cv.imshow('result1', resultN)
# cv.imshow('matching result', img_matching_result)
```

ㄴ Img3 영상을 cv.warpPerspective 함수를 활용하여 원근을 조정하여 Stitch 를 위한 전처리를 수행하고 (result 에 저장) 이후 cv.warpAffine 진행하였다.

이때 Stitching 결과가 영상이 잘리는 현상이 발생했다.

따라서 겹치는 영역의 네 모서리 좌표를 추출하여 영상이 잘리지 않도록 크기를 조정해줄 필요가 있다.

- img3 의 각 모서리 좌표 * H(Homography matrix) = 겹치는 영역의 각 모서리 좌표

```
point = [0,0,1]
A = np.array(point).transpose()
point = [img3.shape[1],0,1]
B = np.array(point).transpose()
point = [0,img3.shape[0],1]
C = np.array(point).transpose()
point = [img3.shape[1],img3.shape[0],1]
D = np.array(point).transpose()

matA = H.copy()
matB = H.copy()
matC = H.copy()
matD = H.copy()

AA =np.matmul(matA,A)
BB =np.matmul(matB,B)
CC =np.matmul(matC,C)
DD =np.matmul(matD,D)
```

이때 작업하는 환경의 2 차원으로 보기 위해서는 x,y 좌표로 다루어야하는데 현재 AA 는 3 차원을 가지므로 z 축의 값으로 나눠주어야 다룰 수 있는 x 좌표가 된다.

```
conerX = []
conerY = []

conerX.append(AA[0]/AA[2])
conerX.append(BB[0]/BB[2])
conerX.append(CC[0]/CC[2])
conerX.append(DD[0]/DD[2])

conerY.append(AA[1]/AA[2])
conerY.append(BB[1]/BB[2])
conerY.append(CC[1]/CC[2])
conerY.append(DD[1]/DD[2])
conerY.append(img3.shape[0])

middleBoundary = [[0,0], [img3.shape[1],0],[0,img3.shape[0]],[img3.shape[1],img3.shape[0]]]
```

ㄴ 겹치는 영역 middleBoundary 변수 생성 완료.

3.모든 영상이 잘림 없도록 영상 크기 설정

```
if min(conerY) < 0:
    dx = 0
    dy = math.ceil( min(conerY))
    mtrx = np.float32([[1, 0, dx],
                       [0, 1, -dy]])

    img3 = cv.warpAffine(img3, mtrx, (img3.shape[1]+dx, img3.shape[0]-dy) )
    img2 = cv.warpAffine(img2, mtrx, (img2.shape[1]+dx, img2.shape[0]-dy) )
    middleBoundary[0][1] = middleBoundary[0][1] - dy
    middleBoundary[1][1] = middleBoundary[1][1] - dy
    middleBoundary[2][1] = middleBoundary[2][1] - dy
    middleBoundary[3][1] = middleBoundary[3][1] - dy
```

ㄴ conerY, 즉 겹치는 영역의 Y 값을 기준으로 middleBoundary 의 좌표를 수정해준다.

따라서 겹치는 영역의 범위가 수정되어, 겹치는 부분이 2 번영상을 중심으로 사진이 잘리지 않고 모든 영상이 표현된다.

4. 겹치는 부분에서 영상의 자연스러운 블렌딩 처리

```
intMiddleBoundary = np.array(middleBoundary, dtype=int)
saveImage = result2[ intMiddleBoundary[0][1] +5:intMiddleBoundary[2][1] - 5,
                     intMiddleBoundary[0][0] + 5: intMiddleBoundary[1][0] - 5].copy()

gauImg = cv.GaussianBlur( result2[intMiddleBoundary[0][1] - 5:intMiddleBoundary[2][1]+ 5,
                                  intMiddleBoundary[0][0] - 5 : intMiddleBoundary[1][0] + 5] , (5,5),0)

result2[ intMiddleBoundary[0][1] - 5:intMiddleBoundary[2][1]+ 5,
         intMiddleBoundary[0][0] - 5 : intMiddleBoundary[1][0] + 5] =  gauImg
result2[ intMiddleBoundary[0][1] +5:intMiddleBoundary[2][1] - 5,
         intMiddleBoundary[0][0] + 5: intMiddleBoundary[1][0] - 5 ] = saveImage
```

겹치는 영역의 테두리에 일정범위만 gauusianblur 처리를 진행

5.최종결과물

ㄴ 최종 결과물 생성