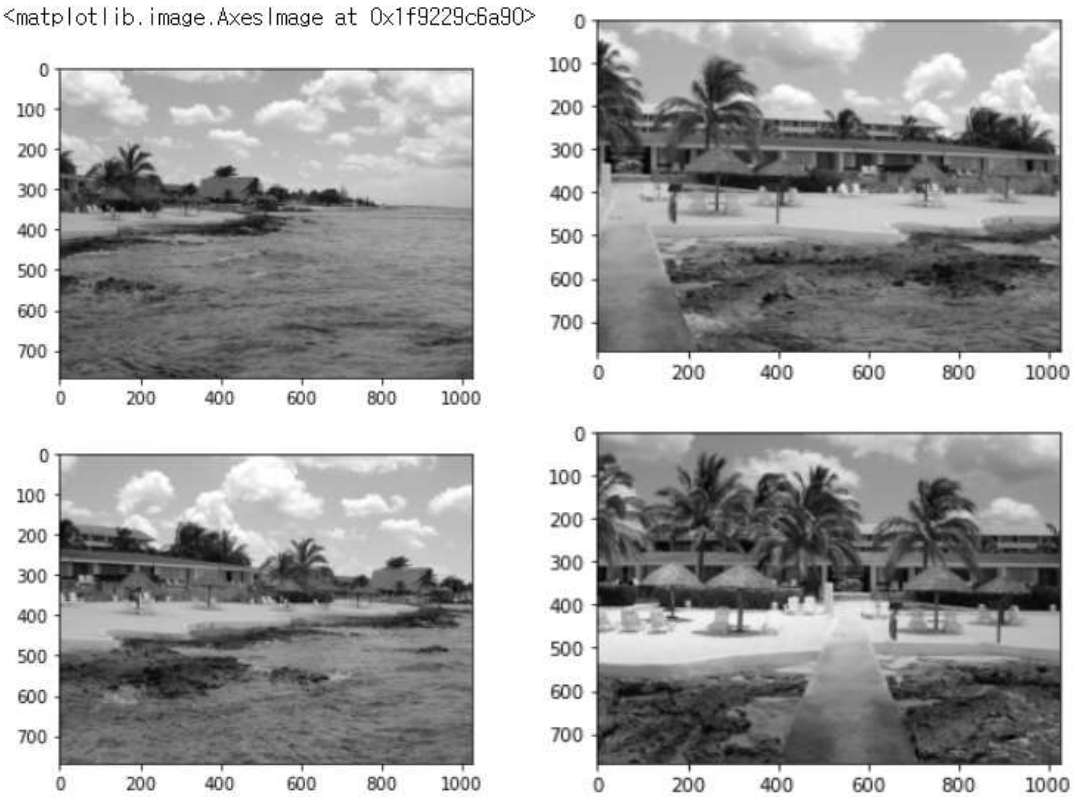
멀티미디어 정보처리 중간고사 대체 과제

2017012106 손민우

1. CV2.Stitcher_create() 함수를 사용한 파노라마 이미지 생성

```
img_names = ['hotel-00.png', 'hotel-01.png', 'hotel-02.png', 'hotel-03.png']

imgs = []
for i, name in enumerate(img_names):
    img = cv2.imread(name)
    plt.figure(figsize=(15,15))
    plt.subplot(len(img_names)//3+1,3,i+1)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    imgs.append(img)

stitcher = cv2.Stitcher_create()

status, dst = stitcher.stitch(imgs)

plt.figure(figsize=(20,20))
plt.imshow(cv2.cvtColor(dst, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1f9229c6a90>
```



2. CV2.Stitcher_create() 함수를 사용하지 않고 파노라마 이미지 생성

```
def KeypointsMatching(keyPoints_L, keyPoints_R, descriptors_L, descriptors_R):

    bf = cv2.BFMatcher()
    raw_matches = bf.knnMatch(descriptors_L, descriptors_R, k=2)

    matches = []
    for m in raw_matches:
        if len(m) == 2 and m[0].distance < m[1].distance * 0.79:
            matches.append((m[0].trainIdx, m[0].queryIdx))

    if len(matches) >= 4:

        keyPoints_L = np.float32([keyPoints_L[i] for (_, i) in matches])
        keyPoints_R = np.float32([keyPoints_R[i] for (i, _) in matches])

        H, status = cv2.findHomography(keyPoints_L, keyPoints_R, cv2.RANSAC, 4.0)


    return matches, H, status
```

```
def Draw_with_Matching(image_L, image_R, keyPoints_L, keyPoints_R, matches, status):
    h1, w1 = image_L.shape[:2]
    h2, w2 = image_R.shape[:2]

    img_matching_result = np.zeros((max(h1, h2), w1 + w2, 3), dtype="uint8")

    img_matching_result[0:h2, 0:w2] = image_R
    img_matching_result[0:h1, w2:] = image_L

    for ((trainIdx, queryIdx), s) in zip(matches, status):

        if s == 1:
            keyPoint2 = (int(keyPoints_R[trainIdx][0]), int(keyPoints_R[trainIdx][1]))
            keyPoint1 = (int(keyPoints_L[queryIdx][0]) + w2, int(keyPoints_L[queryIdx][1]))
            cv2.line(img_matching_result, keyPoint1, keyPoint2, (0, 255, 0), 1)

    return img_matching_result
```

```
def Panorama_nonstit(img1, img2):
    if str(type(img1)) == "<class 'str'>":
        image_L = cv2.imread(img1)
        gray_L = cv2.cvtColor(image_L, cv2.COLOR_BGR2GRAY)
    else:
        image_L = img1
        gray_L = img1

    if str(type(img2)) == "<class 'str'>":
        image_R = cv2.imread(img2)
        gray_R = cv2.cvtColor(image_R, cv2.COLOR_BGR2GRAY)
    else:
        image_R = img2
        gray_R = img2


    detector = cv2.xfeatures2d.SIFT_create()
    keyPoints_L, descriptors_L = detector.detectAndCompute(gray_L, None)
    keyPoints_R, descriptors_R = detector.detectAndCompute(gray_R, None)
    print('img1 - %d features, img2 - %d features' % (len(keyPoints_L), len(keyPoints_R)))

    keyPoints_L = np.float32([keypoint.pt for keypoint in keyPoints_L])
    keyPoints_R = np.float32([keypoint.pt for keypoint in keyPoints_R])

    matches, H, status = KeypointsMatching(keyPoints_L, keyPoints_R, descriptors_L, descriptors_R)

    img_matching_result = Draw_with_Matching(image_L, image_R, keyPoints_L, keyPoints_R, matches, status)


    result = cv2.warpPerspective(image_L, H,
                                 (image_L.shape[1] + image_R.shape[1], image_L.shape[0]))
    result[0:image_R.shape[0], 0:image_R.shape[1]] = image_R

    return result
#plt.figure(figsize=(20,20))
#plt.imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
```

```
pam1 = Panorama_nonstit('hotel-00.png', 'hotel-01.png')
pam2 = Panorama_nonstit(pam1, 'hotel-02.png')
pam3 = Panorama_nonstit(pam2, 'hotel-03.png')
```

```
img1 - 4494 features, img2 - 5467 features
img1 - 7192 features, img2 - 5415 features
img1 - 10003 features, img2 - 5959 features
```

```
plt.figure(figsize=(20,20))
plt.imshow(cv2.cvtColor(pam3, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1f92309ecf8>
```

3. 두 파노라마 이미지 비교 및 분석

우선 CV2.Stitcher_create()를 사용했을 때는 사용하지 않았을 때보다 더 자연스럽게 파노라마가 만들어진 것을 확인할 수 있었다. 파노라마 영상 크기부터 CV2.Stitcher_create()를 사용했을 때가 사용하지 않았을 때보다 절반 정도라는 것을 알 수 있다. 또한 이미지 간 경계가 CV2.Stitcher_create()를 사용했을 때보다 사용하지 않았을 때 더 뚜렷하게 보이는 것을 확인할 수 있다.

그 이유를 분석했을 때, 첫 번째 요인은 CV2.Stitcher_create()에서는 영상끼리 이어지는 부분에서 경계를 블렌딩 처리해서 보다 자연스러운 파노라마를 내도록 한 것이다. CV2.Stitcher_create()를 사용하지 않았을 때는 블렌딩 처리 부분이 없었기 때문에 영상끼리 이어지는 부분에서 경계가 뚜렷한데 비해 CV2.Stitcher_create()는 자연스럽기 때문에 영상 접합 시 블렌딩 처리를 했다는 것을 알 수 있었다.

두 번째 요인은 CV2.Stitcher_create()은 KeypointsMatching의 findHomography다. 이 부분에서 내가 작성한 코드와 다소 다른 부분이 있을 것이라 생각했다. RANSAC 반복 횟수(maxIters)가 더 높거나 RANSAC 재투영 에러 혀용치(ransacReprojThreshold) 혹은 confidence 수치가 좀 더 최적화 되어 있을 가능성이 높다고 생각한다. 이로 인해 이미지 특징점 매칭 정도가 CV2.Stitcher_create()를 사용하지 않았을 때 보다 최적화되어 파노라마의 크기가 더 작아질 수 있었다고 생각한다.

세 번째 요인은 이미지 명암 보정 기능이다. CV2.Stitcher_create()는 파노라마 영상 내에서 명암 정도가 자연스럽다. 특히 hotel-03.png 부분에서 명암이 많이 달리지는데 이것을 자체적으로 전체 명암값을 보정하는 부분이 CV2.Stitcher_create() 내에 포함되었다고 생각한다. CV2.Stitcher_create()를 사용하지 않았을 때는 hotel-03 부분만 명확하게 명암 정도가 다르기 때문이다.

위 세가지 요인 정도가 CV2.Stitcher_create()의 보다 더 자연스러운 파노라마 영상 생성에 도움이 되었을 것이라 생각한다. 문제점에 대한 해결 방안으로는 영상 접합 부분에서 블렌딩 처리, findHomography 함수의 parameter 최적화, 파노라마 명암 보정이 있을 것이다. 특히 두 번째 요인이 많은 영향을 끼쳤을 것으로 예상하며, 이 수치를 변경해가며 최적을 결과를 찾아보며 코드를 발전시켜야겠다는 생각을 하게 되었다.