Computer Vision보고서 9주차

임형찬, 202102559, 인공지능학과

[과제]

- 구현코드 및 코드설명

1. matches 리스트 초기화: 일치 정보를 저장할 빈 리스트 matches 초기화.
2. 첫 번째 이미지의 모든 특징점에 대해 반복, 각 특징점을 SecImage_des에서 하나씩 가져와서 sec_desc에 할당
3. 현재까지의 가장 좋은 일치 정보를 저장 in best_match, 해당 일치의 거리를 저장할 best_distance 변수를 초기화
4.cf). best_distance를 무한대(infinity)로 설정, 처음에는 어떤 거리보다도 크다고 가정하도록 한다.
5. 두 번째루프는 2번째 이미지의 모든 특징점에 대해 반복하며. 각 특징점을 BaseImage_des에서 하나씩 가져와 base_desc에 넣는다.
6. 현재 외부 및 내부 루프에서 선택한 두 개간의 L2 거리 계산.
7. 현재 계산한 거리 dist가 이전까지의 최적 거리 best_distance보다 작은 경우, 새로운 최적 일치가 발견되었음을 의미하므로 best_distance를 dist로 업데이트하고 best_match를 새로운 cv2.DMatch 객체로 업데이트한 후 이 일치 정보를 matches 리스트에 추가, cv2.DMatch 객체를 사용하여 어떤 특징점이 어떤 특징점과 어떤 거리로 일치했는지 저장한다.

간단하게 BruteForce방식의 matching Algorithm으로 구현을 진행.

```python
def My_matcher(SecImage_des, BaseImage_des):
    ################################################################################
    # TODO 두 이미지의 Keypoint Matching
    # 실습 PPT에 나와있는 조건1 ~ 조건3을 만족해야함
    # 나머지 사항은 자유롭게 구현
    # Matching 시 cv2.DMatch 클래스객체에 담아서 다음과 같은 형식으로 matches를 작성
    # 예시 matches.append([cv2.DMatch(_queryIdx=i, _trainIdx=j, _distance=dist)])
    # 위의 예시에서 cv2.DMatchD의 첫번째 인자 _queryIdx는 SecImage_des의 index
    # _trainIdx: BaseImage_des의 index
    # _distance: SecImage_des, BaseImage_des의 L2 distnace 값
    ################################################################################
    matches = []

    # Loop through descriptors and find matches
    for i, sec_desc in enumerate(SecImage_des):
        best_match = None
        best_distance = float('inf')

        for j, base_desc in enumerate(BaseImage_des):
            # Calculate L2 distance between descriptors
            dist = np.linalg.norm(sec_desc - base_desc)

            if dist < best_distance:
                best_distance = dist
                best_match = cv2.DMatch(_queryIdx=i, _trainIdx=j, _distance=dist)

        matches.append(best_match)

    return matches

def Keypoint_Extraction_and_Matching(SecImage, BaseImage):

    ################################################################################
    # TODO 두 이미지 간의 Keypoint 추출 및 Matching
    # 1. keypoint의 추출은 내장 모듈 사용(cv2.SIFT_create()) - 기본 코드 제공
    # 2. 두 이미지로부터 추출된 keypoint와 descriptor를 사용하여 Matching
    #    -> My_matcher 함수 작성
    ################################################################################

    # Using SIFT to find the keypoints and decriptors in the images
    Sift = cv2.SIFT_create()
    SecImage_kp, SecImage_des = Sift.detectAndCompute(cv2.cvtColor(SecImage, cv2.COLOR_BGR2GRAY), None)
    BaseImage_kp, BaseImage_des = Sift.detectAndCompute(cv2.cvtColor(BaseImage, cv2.COLOR_BGR2GRAY), None)
    InitialMatches = My_matcher(SecImage_des, BaseImage_des)

    # Applying ratio test and filtering out the good matches.
    GoodMatches = []
    for match in InitialMatches:
        if match.distance < 0.75 * n.distance:  # Unpack the n distance as well
            GoodMatches.append(match)

    GoodMatches = sorted(GoodMatches, key=lambda x: x.distance)
    return GoodMatches, BaseImage_kp, SecImage_kp
```

```python
def backward_warping(src, M, output_size, shift=[0, 0]):
    """
    인자 정보
    src: backward warping 시킬 이미지
    M: 변환 행렬
    output_size: backward warping 이미지의 높이, 너비
    shift: shift된 정도를 나타내는 값(y, x) 또는 (x, y)
    """

    ###########################################################################
    # TODO Backward warping 구현
    # 실습 PPT 참고
    # return 값
    # 1. backwarping된 결과 이미지
    # 2. backwarping된 결과 이미지의 실제 값이 존재하는 영역에 대한 mask
    #    -> 해당 mask는 높이 x 너비 x 1 구조이고 값은 0과 1로 구성된 binary
    ###########################################################################

    h, w, _ = src.shape

    # M 역행렬 구하기
    M_inv = np.linalg.inv(M)

    out_height, out_width = output_size

    dst = np.zeros((out_height, out_width, src.shape[2]), dtype=np.float32)

    mask = np.zeros((out_height, out_width, 1), dtype=np.float32)

    for y in range(out_height):
        for x in range(out_width):
            p = np.dot(M_inv, np.array([x - shift[0], y - shift[1], 1]).T)
            p /= p[2]
            src_x = int(p[0])
            src_y = int(p[1])

            if 0 <= src_x < w and 0 <= src_y < h:
                dst[y, x] = src[src_y, src_x]
                mask[y, x] = 1.0


    dst = dst.astype(np.uint8)
    return dst, mask
```

이전 과제와 거의 비슷하게 구현.

1.출력 이미지의 모든 픽셀을 반복하면서 (out_height 및 out_width를 기반으로), 각 픽셀의 좌표를 아핀 변환 매트릭스 (M_inv)를 사용하여 원본 이미지에서 가져온다.

2.원본 이미지에서 가져온 좌표가 원본 이미지의 범위 내에 있는지 확인하고, 그렇다면 해당 위치에서 값을 복사하여 출력 이미지를 생성.

3.또한, 유효한 데이터를 나타내는 마스크 (mask)를 생성하여 아핀 변환된 이미지의 유효한 영역을 표시.

구현오류 발생하는 것 같아서 미완입니다
죄송합니다...

```python
def GetStitchingFrameSize(AffineMatrix, Sec_ImageShape, Base_ImageShape):

    """
    인자 정보
    AffineMatrix: AffineMatrix
    Sec_Image: Sec_Image의 높이, 너비
    Base_ImageShape: Base_Image의 높이, 너비

    TODO GetStitchingFrameSize 구현
    # return 값 Stitching 이미지의 크기 Height, Width(높이, 너비) 및 Shift 정도(y ,x)
    # Stitching된 이미지의 크기를 계산
    # Secondary 이미지의 4개의 꼭짓점을 Affine Matrix를 사용하여 Base Image의 좌표계로로 변환(Forward)
    # 변환된 4개의 꼭짓점좌표는 음수 또는 BaseImage의 좌표값의 최댓값을 넘어설수있음
    # 만약 SecondaryImage의 변환된 y좌표의 최솟값이 음수면 음수값의 절댓값만큼 확대
    # 만약 SecondaryImage의 변환된 x좌표의 최댓값이 양수이면서 BaseImage의 최댓값보다크면 x좌표의 최댓값으로 확장
    # 만약 SecondaryImage의 변환된 x좌표의 최솟값이 음수이면 해당음수값의 절댓값만큼 확장
    # 만약 Secondary Image의 변환된 y좌표의 최댓값이 Base Image의 최댓값을 넘어가면 최댓값으로 확장
    # 얼마나 Shift 되었는지는 앞서 구한 X , Y의 최솟값으로 판단. 최솟값이 0이면 변함이 없고 음수면 절댓값 만큼 Shift 되어야함
"""

    sec_corners = np.array([[0, 0, 1],
                            [0, Sec_ImageShape[1], 1],
                            [Sec_ImageShape[0], 0, 1],
                            [Sec_ImageShape[0], Sec_ImageShape[1], 1]])

    transformed_corners = np.dot(AffineMatrix, sec_corners.T).T

    # Calculate the shift required (y, x)
    y_shift = min(0, np.min(transformed_corners[:, 0]))
    x_shift = min(0, np.min(transformed_corners[:, 1]))

    # Calculate the new dimensions for the stitched image
    New_Height = max(Base_ImageShape[0], Sec_ImageShape[0] - abs(y_shift))
    New_Width = max(Base_ImageShape[1], Sec_ImageShape[1] - abs(x_shift))

    Shift_cor = [-y_shift, -x_shift]
    return [New_Height, New_Width], Shift_cor
```