컴퓨터그래픽스 8주차 과제 보고서
201300995 이상건 물리학과

구현 내용
수업시간에 배운 Affine 변환을 예제로 주어진 동영상의 스크린샷을 통해 직접 구현해 보는 것이다. 영상에서 특정 물체를 정면에서 찍은 사진과 옆면에서 찍은 사진을 가지고 이런식으로 변환하라는 목표를 주기 위해 affine 행렬을 만든다. 그래서 정면에서 찍은 물체의 사진을 옆면에서 찍은 것처럼 변환시켜준다.

이유(구현 방법)
일단 실습시간에 알려준 대로 my_LS, my_forward, my_backward부터 작성하기로 했다.
우선 my_LS를 보자. 이건 물체를 정면에서 찍은 사진과 옆면에서 찍은 사진 두 개를 가지고 구 사진간의 특징점을 구해 affine 행렬을 구하는 것이다.

```python
#Least square 방식으로 행렬 M의 요소인 [a, b, c, d, e, f].T 를 구한다.
def my_LS(matches, kp1, kp2):
    '''
    :param matches: keypoint matching 정보
    :param kp1: keypoint 정보.
    :param kp2: keypoint 정보2.
    :return: X : 위의 정보를 바탕으로 Least square 방식으로 구해진 Affine 변환 matrix의 요소 [a, b, c, d, e, f].T
    '''

    length = len(matches)

    '''행렬 A,B를 만들고 이를 이용해 X를 구해주세요.'''

    A = np.zeros((length*2, 6))
    B = np.zeros((length*2, 1))
    X = np.zeros((length*2, 1))
    for i in range(0, length):
        A[(2 * i), 0:3] = [kp1[matches[i].queryIdx].pt[0], kp1[matches[i].queryIdx].pt[1], 1]
        A[(2 * i) + 1, 3:6] = [kp1[matches[i].queryIdx].pt[0], kp1[matches[i].queryIdx].pt[1], 1]
        B[(2 * i)] = kp2[matches[i].trainIdx].pt[0]
        B[(2 * i) + 1] = kp2[matches[i].trainIdx].pt[1]

    X = np.matmul(np.linalg.pinv(A), B)

    return X
```

이미 get_matchpoint를 통해 interest point를 통해 각 포인트 간의 위치에 맞는 index 값이 이미 match 클래스 안에 들어있다. 이를 통해 각각의 매칭된 interest point를 가지고 matrix를 구하면 되는 것이다.
행렬 A와 행렬 B를 이론시간에 배웠던 그대로의 형태로 만든다. 점 x,y 와 x`,y`는 matches를 이용해서 구한다. 그리고 np.linalg.pinv 함수를 통해 least square 방식으로 affine matrix를 구하게 된다.

이제 my_forward를 보자.

```python
#forward warping을 수행한다.
# Mx = x'
def my_forward(img, X):
    '''
    :param img: warping 대상 image
    :param X: 변환행렬값 6개가 저장된 행렬 (6, 1)
    :return: forward warping이 된 image
    '''

    # 겹치는 좌표에 대한 처리는 값을 모두 더해서 평균냄.

    y, x, c = img.shape
    result = np.zeros((y,x,c))
    count = np.zeros((y,x,c))

    '''X를 이용해 Affine matrix M을 만드세요.'''
    M = np.zeros((3,3))
    M[0] = X.T[0,0:3]
    M[1] = X.T[0,3:6]
    M[2,2] = 1

    for i in range(y):
        for j in range(x):
            '''
            M과 좌표 (x,y)를 이용해 새로운 좌표인 newX, newY를 구하세요.
            '''
            oldPosition = np.array([j,i,1])
            newPosition = np.matmul(M,oldPosition).astype(int)
            newX = newPosition[0]
            newY = newPosition[1]
            if newY < y and newX < x and newY > 0 and newX > 0:
                result[newY, newX, :] += img[i,j,:] #값을 해당 좌표에 합산
                count[newY, newX, :] += 1           #몇 개의 값이 한 좌표에 겹쳤는지 count

    result = np.divide(result, count)
    result[np.isnan(result)] = 0                    #divide 후 not a number 처리
    return result
```

우선 계산을 위해 앞에서 구했던 affine matrix를 이론에서 배웠던 행렬의 형식에 맞게 바꾼다. 그 뒤 변환시키려는 이미지의 x,y 좌표값을 넣어 앞에서 학습한 affine matrix로 위치를 변환시킨다.

다음은 my_backward를 보자

```python
#backward warping을 수행
#x = M.inv * x' 로 x를 구한 뒤, 해당 좌표에서의 값을 interpolation으로 구함.
def my_backward(img, X):
    '''
    :param img: warping 대상 image
    :param X: 변환행렬값 6개가 저장된 행렬 (6, 1)
    :return: backward warping이 된 image
    '''

    '''backward warping을 수행하는 코드를 작성하세요.'''

    y, x, c = img.shape
    result = np.zeros((y,x,c))
    count = np.zeros((y,x,c))

    M = np.zeros((3,3))
    M[0] = X.T[0,0:3]
    M[1] = X.T[0,3:6]
    M[2,2] = 1
    Mr = np.linalg.inv(M)
    for i in range(y):
        for j in range(x):

            desPosition = np.array([j,i,1])
            oriPosition = np.matmul(Mr,desPosition.T).astype(int)
            color = my_bilinear(img,oriPosition[0],oriPosition[1])
            # print(color)
            # print(j,i)
            result[i,j] = color

    print(result.shape)
    return result
```

앞에서 구한 forward랑 비슷하지만 여기선 변환될 이미지의 픽셀 위치를 기준으로 변환대상 이미지의 위치를 불러온다는 것이 다르다. 그래서 forward에서 생기는 구멍은 여기선 생기지 않는다. 소수점의 위치를 불러올 때는 my_bilinear를 이용하여 보강시킨다.

위 사진이 원본사진과 목표사진이고

위에가 forward 결과이고 밑에가 backward 결과이다. backward를 쓴 방법이 중간에 검은색 없이 잘 출력됨을 알 수 있다.

느낀 점
어렵다

과제 난이도

어려웠음