컴퓨터 그래픽스 과제 #06

학과 : 컴퓨터공학과
학번 : 201802161
이름 : 조은빈

1. 구현 코드

* feature_matching

```
    # no RANSAC
    if not RANSAC:

        A = []
        B = []
        for idx, point in enumerate(points):
            '''
            #ToDo
            #A, B 완성
            # A.append(???) 이런식으로 할 수 있음
            # 결과만 잘 나오면 다른방법으로 해도 상관없음
            '''
            A.append([point[0][0], point[0][1], 1, 0, 0, 0])
            A.append([0, 0, 0, point[0][0], point[0][1], 1])
            B.append([point[1][0]])
            B.append([point[1][1]])

        A = np.array(A)
        B = np.array(B)

        '''
        #ToDo
        #X 완성
        #np.linalg.inv(V) : V의 역행렬 구하는것
        #np.dot(V1, V2) : V1과 V2의 행렬곱
        # V1.T : V1의 transpose
        '''
        X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T),B)


        '''
        # ToDo
        # 위에서 구한 X를 이용하여 M 완성
        '''
        M = [[X[0][0], X[1][0], X[2][0]],
             [X[3][0], X[4][0], X[5][0]],
             [0, 0, 1]]

        M = np.array(M)
        M_ = np.linalg.inv(M)
```

```
    '''
    # ToDo
    # backward 방식으로 dst완성
    '''
    #Backward 방식
    h, w = img1.shape[:2]
    dst = np.zeros((int(np.round(h)), int(np.round(w)), 3), dtype=np.uint8)
    h_, w_ = dst.shape[:2]
    temp = []
    for row in range(h_):
        for col in range(w_):
            vec = np.dot(M_, np.array([[col, row, 1]]).T)
            temp.append(vec)
            c = vec[0, 0]
            r = vec[1, 0]
            c_left = int(c)
            c_right = min(int(c+1), w-1)
            r_top = int(r)
            r_bottom = min(int(r+1), h-1)
            s = c-c_left
            t = r - r_top
            if r_bottom >= h-1 or c_right >= w-1 or c_left < 0 or r_top < 0:
                dst[row, col] = 0
            else:
                intensity = (1-s) * (1-t)*img1[r_top, c_left]\
                        +s*(1-t)*img1[r_top, c_right]\
                        +(1-s)*t*img1[r_bottom, c_left]\
                        +s*t*img1[r_bottom, c_right]

                dst[row, col] = intensity
    temp = np.array(temp)
```

```
    #use RANSAAC
    else:
        points_shuffle = points.copy()

        inliers = []
        M_list = []
        for i in range(iter_num):
            random.shuffle(points_shuffle)
            three_points = points_shuffle[:3]

            A = []
            B = []
            #3개의 point만 가지고 M 구하기
            for idx, point in enumerate(three_points):
                '''
                #ToDo
                #A, B 완성
                # A.append(???) 이런식으로 할 수 있음
                # 결과만 잘 나오면 다른방법으로 해도 상관없음
                '''
                A.append([point[0][0], point[0][1], 1, 0, 0, 0])
                A.append([0, 0, 0, point[0][0], point[0][1], 1])
                B.append([point[1][0]])
                B.append([point[1][1]])


            A = np.array(A)
            B = np.array(B)
            try:
                '''
                #ToDo
                #X 완성
                #np.linalg.inv(V) : V의 역행렬 구하는것
                #np.dot(V1, V2) : V1과 V2의 행렬곱
                # V1.T : V1의 transpose 단, type의 np.array일때만 가능. type의 list일때는 안됨
                '''
                X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T),B)
```

```
            except:
                print('can\'t calculate np.linalg.inv((np.dot(A.T, A)) !!!!!')
                continue

            '''
            # ToDo
            # 위에서 구한 X를 이용하여 M 완성
            '''
            M = [[X[0][0], X[1][0], X[2][0]],
                 [X[3][0], X[4][0], X[5][0]],
                 [0, 0, 1]]

            M_list.append(M)

            count_inliers = 0
            for idx, point in enumerate(points):
                '''
                # ToDo
                # 위에서 구한 M으로(3개의 point로 만든 M) 모든 point들에 대하여 예상 point 구하기
                # 구해진 예상 point와 실제 point간의 L2 distance 를 구해서 threshold_distance보다 작은 값이 있는 경우 inlier로 판단
                '''
                M_point = np.dot(M,[[point[0][0]], [point[0][1]], [1]])
                real_point = [point[1][0], point[1][1]]
                if L2_distance(np.array([M_point[0][0], M_point[1][0]]), np.array(real_point)) < threshold_distance:
                    count_inliers += 1

            inliers.append(count_inliers)

        inliers = np.array(inliers)
        max_inliers_idx = np.argmax(inliers)

        best_M = np.array(M_list[max_inliers_idx])

        M = best_M
        M_ = np.linalg.inv(M)
```

```
        '''
        # ToDo
        # backward 방식으로 dst완성
        '''
        #Backward 방식
        h, w = img1.shape[:2]
        dst = np.zeros((int(np.round(h*2.5)), int(np.round(w*1.5)), 3), dtype=np.uint8)
        h_, w_ = dst.shape[:2]

        for row in range(h_):
            for col in range(w_):
                vec = np.dot(M_, np.array([[col, row, 1]]).T)
                c = vec[0, 0]
                r = vec[1, 0]
                c_left = int(c)
                c_right = min(int(c + 1), w - 1)
                r_top = int(r)
                r_bottom = min(int(r + 1), h - 1)
                s = c - c_left
                t = r - r_top
                if r_bottom >= h - 1 or c_right >= w - 1 or c_left < 0 or r_top < 0:
                    dst[row, col] = 0
                else:
                    intensity = (1 - s) * (1 - t) * img1[r_top, c_left] \
                                + s * (1 - t) * img1[r_top, c_right] \
                                + (1 - s) * t * img1[r_bottom, c_left] \
                                + s * t * img1[r_bottom, c_right]

                    dst[row, col] = intensity

    return dst
```

* L2_distance

```
229     def L2_distance(vector1, vector2):
230         '''
231         #vector1과 vector2의 거리 구하기 (L2 distance)
232         #distance 는 스칼라
233         #np.sqrt(), np.sum() 를 잘 활용하여 구하기
234         #L2 distance를 구하는 내장함수로 거리를 구한 경우 감점
235         '''
236         distance = np.sqrt(np.sum(np.square(vector2-vector1)))
237         return distance
```

2. 코드 설명

$$x_1' = ax_1 + by_1 + c \qquad x_2' = ax_2 + by_2 + c$$
$$y_1' = dx_1 + ey_1 + f \qquad y_2' = dx_2 + ey_2 + f$$

$$x_3' = ax_3 + by_3 + c \qquad x_4' = ax_4 + by_4 + c$$
$$y_3' = dx_3 + ey_3 + f \qquad y_4' = dx_4 + ey_4 + f$$

$$\underset{8}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ & & & \vdots & & \end{bmatrix}} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ \vdots \end{bmatrix} \Rightarrow \mathbf{Ax = b}$$

$\mathbf{A}$ $\mathbf{x}$ $\mathbf{b}$

point는 [(img1 x 좌표, img1 y 좌표), (img2 x 좌표, img2 y 좌표)]이다. 위 공식과 같은 표기로 나타내면 [(x_1, y_1), (x_1', y_1')]이다. x_1 = point[0][0], y_1 = point[0][1], x_1' = point[1][0], y_1' = point[1][1]이므로 이를 토대로 위와 같은 형태로 A와 B 배열에 추가해 준다.

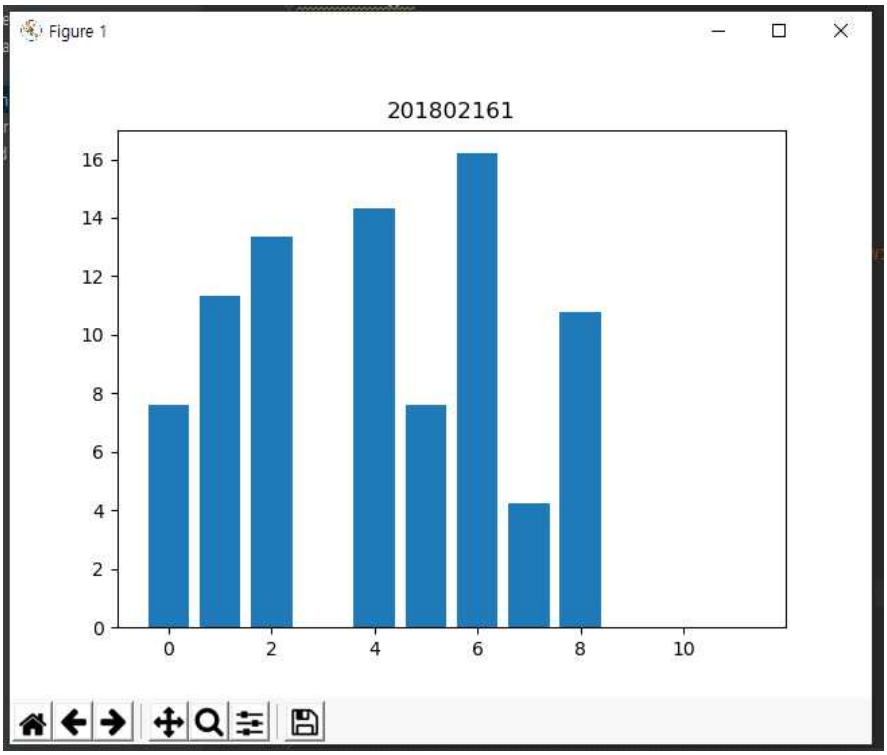
X는 아래 공식을 코드로 구현해 구한다.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

X를 구하면 a부터 f까지 6개의 미지수를 알 수 있다. X[0][0]부터 X[5][0]이 차례대로 a부터 f까지의 미지수이다. M은 $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$ 이므로 X[0][0]부터 X[5][0]까지의 값으로 M을 채운다.

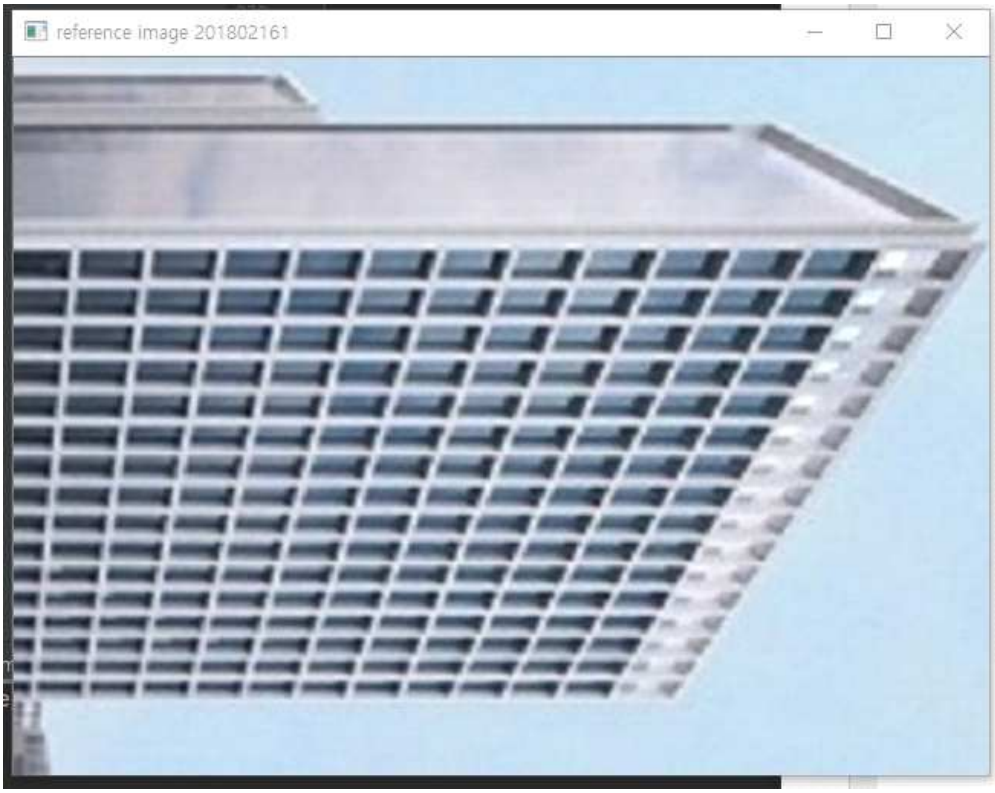
3. 이미지

* 실습



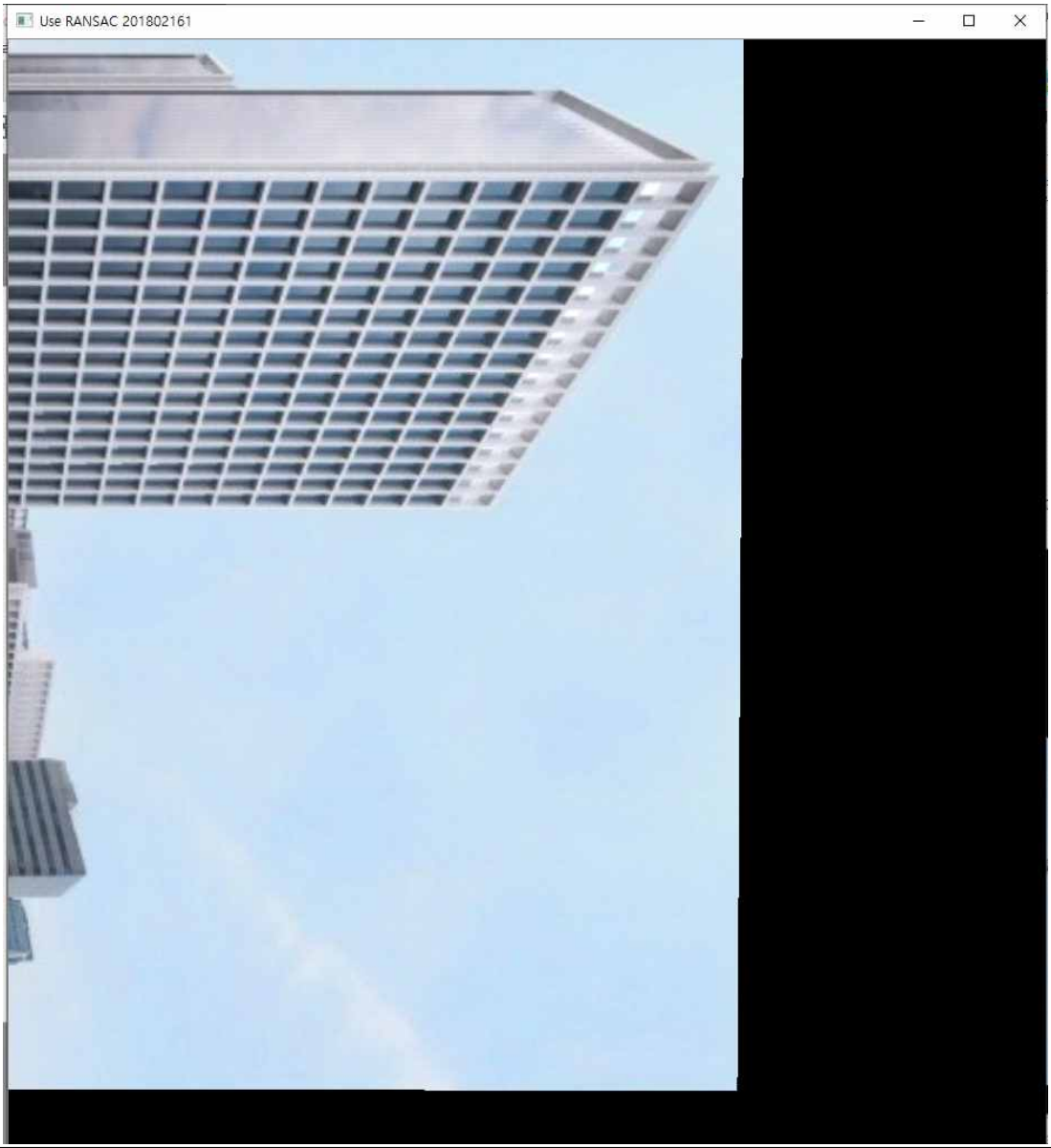
* original

* reference



* No RANSAC

* Use RANSAC



4. 느낀점

이론을 들을 땐 많이 어렵다고 생각되지 않았는데 코드로 구현할 땐 잘 풀리지 않아서 힘들었다. 잘못 구현해도 어디서 잘못된 건지 알기 어려워서 더 구현하기 힘든 것 같다.

5. 과제 난이도

지금까지 나온 과제들 중 가장 어려웠다.