

컴퓨터 그래픽스 과제 #06

학과 : 컴퓨터공학과

학번 : 201802161

이름 : 조은빈

1. 구현 코드

* feature_matching

```
46     # no RANSAC
47     if not RANSAC:
48
49         A = []
50         B = []
51         for idx, point in enumerate(points):
52             '''
53             #ToDo
54             #A, B 완성
55             # A.append(???) 이런식으로 할 수 있음
56             # 결과만 잘 나오면 다른방법으로 해도 상관없음
57             '''
58             A.append([point[0][0], point[0][1], 1, 0, 0, 0])
59             A.append([0, 0, 0, point[0][0], point[0][1], 1])
60             B.append([point[1][0]])
61             B.append([point[1][1]])
62
63         A = np.array(A)
64         B = np.array(B)
65
66         '''
67         #ToDo
68         #X 완성
69         #np.linalg.inv(V) : V의 역행렬 구하는것
70         #np.dot(V1, V2) : V1과 V2의 행렬곱
71         # V1.T : V1의 transpose
72         '''
73         X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), B)
74
75
76         '''
77         # ToDo
78         # 위에서 구한 X를 이용하여 M 완성
79         '''
80         M = [[X[0][0], X[1][0], X[2][0]],
81              [X[3][0], X[4][0], X[5][0]],
82              [0, 0, 1]]
83
84         M = np.array(M)
85         M_inv = np.linalg.inv(M)
```

```

88     """
89     # ToDo
90     # backward 방식으로 dst완성
91     """
92     #Backward 방식
93     h, w = img1.shape[:2]
94     dst = np.zeros((int(np.round(h)), int(np.round(w)), 3), dtype=np.uint8)
95     h_, w_ = dst.shape[:2]
96     temp = []
97     for row in range(h_):
98         for col in range(w_):
99             vec = np.dot(M_, np.array([[col, row, 1]]).T)
100             temp.append(vec)
101             c = vec[0, 0]
102             r = vec[1, 0]
103             c_left = int(c)
104             c_right = min(int(c+1), w-1)
105             r_top = int(r)
106             r_bottom = min(int(r+1), h-1)
107             s = c - c_left
108             t = r - r_top
109             if r_bottom >= h-1 or c_right >= w-1 or c_left < 0 or r_top < 0:
110                 dst[row, col] = 0
111             else:
112                 intensity = (1-s) * (1-t)*img1[r_top, c_left]\
113                     +s*(1-t)*img1[r_top, c_right]\
114                     +(1-s)*t*img1[r_bottom, c_left]\
115                     +s*t*img1[r_bottom, c_right]
116
117                 dst[row, col] = intensity
118     temp = np.array(temp)

```

```

121 #use RANSAAC
122 else:
123     points_shuffle = points.copy()
124
125     inliers = []
126     M_list = []
127     for i in range(iter_num):
128         random.shuffle(points_shuffle)
129         three_points = points_shuffle[:3]
130
131         A = []
132         B = []
133         #3개의 point만 가지고 M 구하기
134         for idx, point in enumerate(three_points):
135             ...
136
137             #ToDo
138             #A, B 완성
139             # A.append(???) 이런식으로 할 수 있음
140             # 결과만 잘 나오면 다른방법으로 해도 상관없음
141             ...
142
143             A.append([point[0][0], point[0][1], 1, 0, 0, 0])
144             A.append([0, 0, 0, point[0][0], point[0][1], 1])
145             B.append([point[1][0]])
146             B.append([point[1][1]])
147
148         A = np.array(A)
149         B = np.array(B)
150         try:
151             ...
152
153             #ToDo
154             #X 완성
155             #np.linalg.inv(V) : V의 역행렬 구하는것
156             #np.dot(V1, V2) : V1과 V2의 행렬곱
157             # V1.T : V1의 transpose 단, type이 np.array일때만 가능. type이 list일때는 안됨
158             ...
159
160             X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), B)

```

```

159 except:
160     print('can\'t calculate np.linalg.inv((np.dot(A.T, A)) !!!!!)')
161     continue
162
163     ...
164     # ToDo
165     # 위에서 구한 X를 이용하여 M 완성
166     ...
167     M = [[X[0][0], X[1][0], X[2][0]],
168           [X[3][0], X[4][0], X[5][0]],
169           [0, 0, 1]]
170
171     M_list.append(M)
172
173     count_inliers = 0
174     for idx, point in enumerate(points):
175         ...
176         # ToDo
177         # 위에서 구한 M으로(3개의 point로 만든 M) 모든 point들에 대하여 예상 point 구하기
178         # 구해진 예상 point와 실제 point간의 L2 distance 를 구해서 threshold_distance보다 작은 값이 있는 경우 inlier로 판단
179         ...
180         M_point = np.dot(M, [[point[0][0], [point[0][1], [1]]])
181         real_point = [point[1][0], point[1][1]]
182         if L2_distance(np.array([M_point[0][0], M_point[1][0]]), np.array(real_point)) < threshold_distance:
183             count_inliers += 1
184
185     inliers.append(count_inliers)
186
187     inliers = np.array(inliers)
188     max_inliers_idx = np.argmax(inliers)
189
190     best_M = np.array(M_list[max_inliers_idx])
191
192     M = best_M
193     M_inv = np.linalg.inv(M)
194

```

```

196     ...
197     # ToDo
198     # backward 방식으로 dst완성
199     ...
200     #Backward 방식
201     h, w = img1.shape[:2]
202     dst = np.zeros((int(np.round(h*2.5)), int(np.round(w*1.5)), 3), dtype=np.uint8)
203     h_, w_ = dst.shape[:2]
204
205     for row in range(h_):
206         for col in range(w_):
207             vec = np.dot(M_inv, np.array([col, row, 1])).T
208             c = vec[0, 0]
209             r = vec[1, 0]
210             c_left = int(c)
211             c_right = min(int(c + 1), w - 1)
212             r_top = int(r)
213             r_bottom = min(int(r + 1), h - 1)
214             s = c - c_left
215             t = r - r_top
216             if r_bottom >= h - 1 or c_right >= w - 1 or c_left < 0 or r_top < 0:
217                 dst[row, col] = 0
218             else:
219                 intensity = (1 - s) * (1 - t) * img1[r_top, c_left] \
220                     + s * (1 - t) * img1[r_top, c_right] \
221                     + (1 - s) * t * img1[r_bottom, c_left] \
222                     + s * t * img1[r_bottom, c_right]
223
224                 dst[row, col] = intensity
225
226     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

$$\begin{aligned} x'_1 &= ax_1 + by_1 + c & x'_2 &= ax_2 + by_2 + c \\ y'_1 &= dx_1 + ey_1 + f & y'_2 &= dx_2 + ey_2 + f \end{aligned}$$

$$\begin{aligned} x'_3 &= ax_3 + by_3 + c & x'_4 &= ax_4 + by_4 + c \\ y'_3 &= dx_3 + ey_3 + f & y'_4 &= dx_4 + ey_4 + f \end{aligned}$$

$$\begin{matrix} \uparrow \\ 8 \\ \downarrow \end{matrix} \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ \vdots \end{bmatrix} \Rightarrow \mathbf{Ax} = \mathbf{b}$$

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

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. $x_1 = \text{point}[0][0]$, $y_1 = \text{point}[0][1]$, $x'_1 = \text{point}[1][0]$, $y'_1 = \text{point}[1][1]$ 이므로 이를 토대로 위와 같은 형태로 A와 B 배열에 추가해 준다.

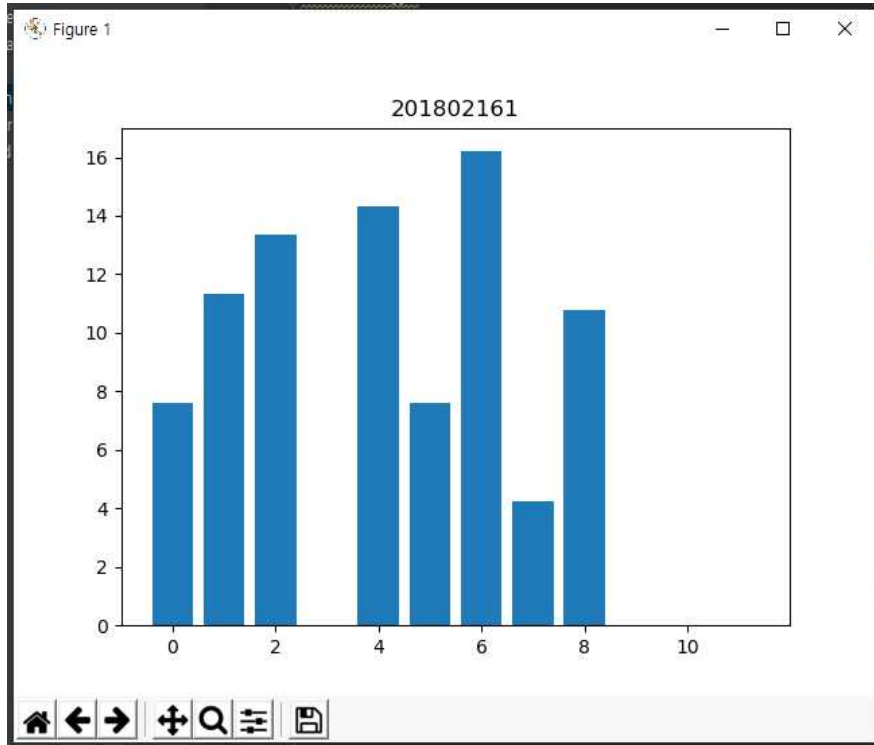
X는 아래 공식을 코드로 구현해 구한다.

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

X를 구하면 a 부터 f 까지 6개의 미지수를 알 수 있다. $X[0][0]$ 부터 $X[5][0]$ 이 차례대로 a 부터 f 까지의 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



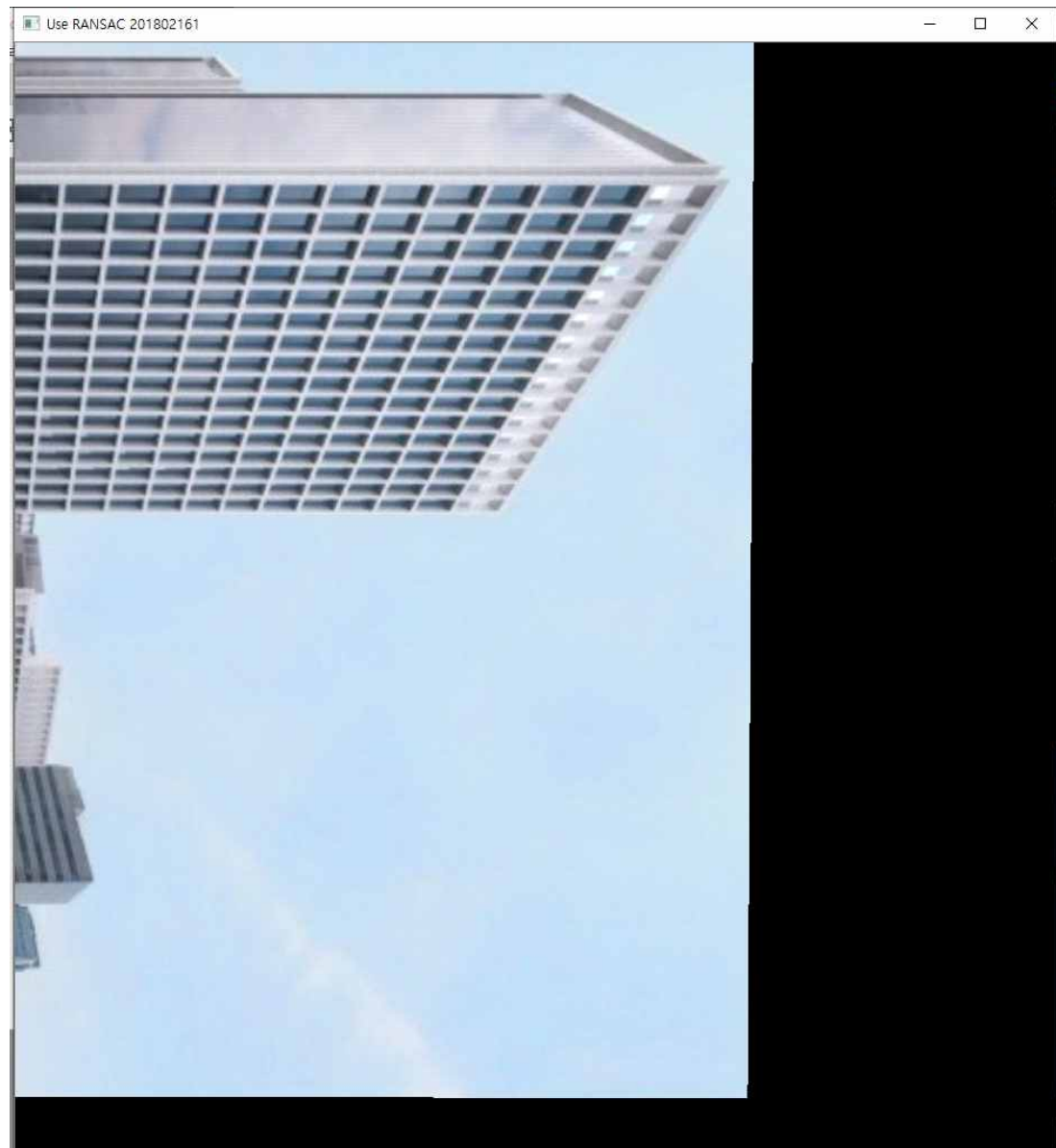
* reference



* 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.