

컴퓨터 그래픽스 과제 #07

학과 : 컴퓨터공학과

학번 : 201802161

이름 : 조은빈

1. 구현 코드

* feature_matching

```
6 def feature_matching(img1, img2, RANSAC=False, threshold = 300, keypoint_num = None, iter_num = 500, threshold_distance = 10):
7     """
8     #ToDo
9     #바뀐것은 return dst -> return dst, M 이것밖에 없습니다. 6주차 과제 완성한 내용을 그대로 복사해주세요
10    """
11    # sift = cv2.xfeatures2d.SIFT_create(keypoint_num)
12    # sift = cv2.xfeatures2d.SIFT_create()
13    if keypoint_num == None:
14        sift = cv2.xfeatures2d.SIFT_create()
15    else:
16        sift = cv2.xfeatures2d.SIFT_create(keypoint_num)
17
18    kp1, des1 = sift.detectAndCompute(img1, None)
19    kp2, des2 = sift.detectAndCompute(img2, None)
20
21    distance = []
22    for idx_1, des_1 in enumerate(des1):
23        dist = []
24        for idx_2, des_2 in enumerate(des2):
25            dist.append(L2_distance(des_1, des_2))
26
27        distance.append(dist)
28
29    distance = np.array(distance)
30
31    min_dist_idx = np.argmin(distance, axis=1)
32    min_dist_value = np.min(distance, axis=1)
33
34    points = []
35    for idx, point in enumerate(kp1):
36        if min_dist_value[idx] >= threshold:
37            continue
38
39        x1, y1 = point.pt
40        x2, y2 = kp2[min_dist_idx[idx]].pt
41
42        x1 = int(np.round(x1))
43        y1 = int(np.round(y1))
```

```

45         x2 = int(np.round(x2))
46         y2 = int(np.round(y2))
47         points.append([(x1, y1), (x2, y2)])
48
49     # no RANSAC
50     if not RANSAC:
51
52         A = []
53         B = []
54         for idx, point in enumerate(points):
55             '''
56             #ToDo
57             #A, B 완성
58             # A.append(???) 이런식으로 할 수 있음
59             # 결과만 잘 나오면 다른방법으로 해도 상관없음
60             '''
61             A.append([point[0][0], point[0][1], 1, 0, 0, 0])
62             A.append([0, 0, 0, point[0][0], point[0][1], 1])
63             B.append([point[1][0]])
64             B.append([point[1][1]])
65
66         A = np.array(A)
67         B = np.array(B)
68
69         '''
70         #ToDo
71         #X 완성
72         #np.linalg.inv(V) : V의 역행렬 구하는것
73         #np.dot(V1, V2) : V1과 V2의 행렬곱
74         # V1.T : V1의 transpose
75         '''
76         X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), B)
77
78         '''
79         # ToDo
80         # 위에서 구한 X를 이용하여 M 완성
81         '''
82         M = [[X[0][0], X[1][0], X[2][0]],
83              [X[3][0], X[4][0], X[5][0]],
84              [0, 0, 1]]

```

```

86     M = np.array(M)
87     M_ = np.linalg.inv(M)
88
89     '''
90     # ToDo
91     # backward 방식으로 dst완성
92     '''
93     # Backward 방식
94     h, w = img1.shape[:2]
95     dst = np.zeros((int(np.round(h*2)), int(np.round(w*2)), 3), dtype=np.uint8)
96     h_, w_ = dst.shape[:2]
97
98     for row in range(h_):
99         for col in range(w_):
100             vec = np.dot(M_, np.array([[col, row, 1]]).T)
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

```

```

122     # use RANSAC
123     else:
124         points_shuffle = points.copy()
125
126         inliers = []
127         M_list = []
128         for i in range(iter_num):
129             random.shuffle(points_shuffle)
130             three_points = points_shuffle[:3]
131
132             A = []
133             B = []
134             # 3개의 point만 가지고 M 구하기
135             for idx, point in enumerate(three_points):
136                 """
137                 #ToDo
138                 #A, B 완성
139                 # A.append(???) 이런식으로 할 수 있음
140                 # 결과만 잘 나오면 다른방법으로 해도 상관없음
141                 """
142                 A.append([point[0][0], point[0][1], 1, 0, 0, 0])
143                 A.append([0, 0, 0, point[0][0], point[0][1], 1])
144                 B.append([point[1][0]])
145                 B.append([point[1][1]])
146
147             A = np.array(A)
148             B = np.array(B)
149             try:
150                 """
151                 #ToDo
152                 #X 완성
153                 #np.linalg.inv(V) : V의 역행렬 구하는것
154                 #np.dot(V1, V2) : V1과 V2의 행렬곱
155                 # V1.T : V1의 transpose 단, type이 np.array일때만 가능. type이 list일때는 안됨
156                 """
157                 X = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), B)
158
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 을 구해서 threshold_distance보다 작은 값이 있는 경우 inlier로 판단
179         ...
180         # ???(M으로 구한 point) # point1에 M을 곱해서 구한거 알하는듯
181         # ???(실제 point) # point안에 img1이랑 img2 각각의 좌표 있는거 그거 알하는듯
182         M_point = np.dot(M, [[point[0][0]], [point[0][1]], [1]])
183         real_point = [point[1][0], point[1][1]]
184         if L2_distance(np.array([M_point[0][0], M_point[1][0]]), np.array(real_point)) < threshold_distance:
185             count_inliers += 1
186         # if L2_distance(??? (M으로 구한 point), ??? (실제 point)) < threshold_distance:
187         #     count_inliers += 1
188
189     inliers.append(count_inliers)
190
191     inliers = np.array(inliers)
192     max_inliers_idx = np.argmax(inliers)
193
194     best_M = np.array(M_list[max_inliers_idx])
195
196     M = best_M
197     M_inv = np.linalg.inv(M)

```

```

199
200     # ToDo
201     # backward 방식으로 dst완성
202     '''
203     # Backward 방식
204     h, w = img1.shape[:2]
205     dst = np.zeros((int(np.round(h*2)), int(np.round(w * 2)), 3), dtype=np.uint8)
206     h_, w_ = dst.shape[:2]
207
208     for row in range(h_):
209         for col in range(w_):
210             vec = np.dot(M_, np.array([[col, row, 1]]).T)
211             c = vec[0, 0]
212             r = vec[1, 0]
213             c_left = int(c)
214             c_right = min(int(c + 1), w - 1)
215             r_top = int(r)
216             r_bottom = min(int(r + 1), h - 1)
217             s = c - c_left
218             t = r - r_top
219             if r_bottom >= h - 1 or c_right >= w - 1 or c_left < 0 or r_top < 0:
220                 dst[row, col] = 0
221             else:
222                 intensity = (1 - s) * (1 - t) * img1[r_top, c_left] \
223                     + s * (1 - t) * img1[r_top, c_right] \
224                     + (1 - s) * t * img1[r_bottom, c_left] \
225                     + s * t * img1[r_bottom, c_right]
226
227                 dst[row, col] = intensity
228
229     return dst, M

```

* L2_distance

```

231 def L2_distance(vector1, vector2):
232     '''
233     #ToDo
234     #6주차의 내용을 그대로 복붙해주세요
235     '''
236     distance = np.sqrt(np.sum(np.square(vector2 - vector1)))
237     return distance

```

* scaling_test

```

239 #실습 때 해터 코드입니다.
240 def scaling_test(src):
241     h, w = src.shape[:2]
242     rate = 2
243     dst_for = np.zeros((int(np.round(h*rate)), int(np.round(w*rate)), 3))
244     dst_back_bilinear = np.zeros((int(np.round(h*rate)), int(np.round(w*rate)), 3))
245     M = np.array([[rate, 0, 0],
246                  [0, rate, 0],
247                  [0, 0, 1]])
248
249     #FORWARD
250     h_, w_ = dst_for.shape[:2]
251     count = dst_for.copy()
252     for row in range(h):
253         for col in range(w):
254             """
255             #ToDo
256             #과제에서 사용하지 않지만 완성해주세요
257             #실습을 참고해서 완성해주세요
258             """
259             vec = np.dot(M, np.array([[col, row, 1]]).T)
260             x = vec[0,0]
261             y = vec[1,0]
262
263             x1 = int(np.floor(x))
264             x2 = int(np.ceil(x))
265             y1 = int(np.floor(y))
266             y2 = int(np.ceil(y))
267
268             points_list = [(y1, x1), (y1, x2), (y2, x1), (y2, x2)]
269             points = set(points_list)
270
271             for(row_, col_) in points:
272                 dst_for[min(row_, h_-1), min(col_, w_-1)] += src[row, col]
273                 count[min(row_, h_-1), min(col_, w_-1)] += 1
274
275     dst_for = (dst_for / count).astype(np.uint8)

```

```

277     #M 역행렬
278     M_ = np.linalg.inv(M)
279     print('M')
280     print(M)
281     print('M 역행렬')
282     print(M_)
283     h_, w_ = dst_back_bilinear.shape[:2]
284     #BACKWARD
285     for row_ in range(h_):
286         for col_ in range(w_):
287             '''
288             #ToDo
289             #bilinear
290             #실습을 참고해서 완성해주세요
291             '''
292             vec = np.dot(M_, np.array([[col_, row_, 1]]).T)
293             c = vec[0,0]
294             r = vec[1,0]
295             c_left = int(c)
296             c_right = min(int(c + 1), w - 1)
297             r_top = int(r)
298             r_bottom = min(int(r + 1), h - 1)
299             s = c - c_left
300             t = r - r_top
301             intensity = (1 - s) * (1 - t) * src[r_top, c_left] \
302                 + s * (1 - t) * src[r_top, c_right] \
303                 + (1 - s) * t * src[r_bottom, c_left] \
304                 + s * t * src[r_bottom, c_right]
305             dst_back_bilinear[row_, col_] = intensity
306
307     dst_back_bilinear = dst_back_bilinear.astype(np.uint8)
308
309     return dst_back_bilinear, M

```

2. 코드 설명

코드 설명은 6주차와 같다.

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

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

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

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

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. M 은 $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$ 이므로 $X[0][0]$ 부터 $X[5][0]$ 까지의 값으로 M 을 채운다.

M' 를 통해 반영된 빨간 점이 바뀐 이미지에서 어느 위치에 나타나는지 확인한다.

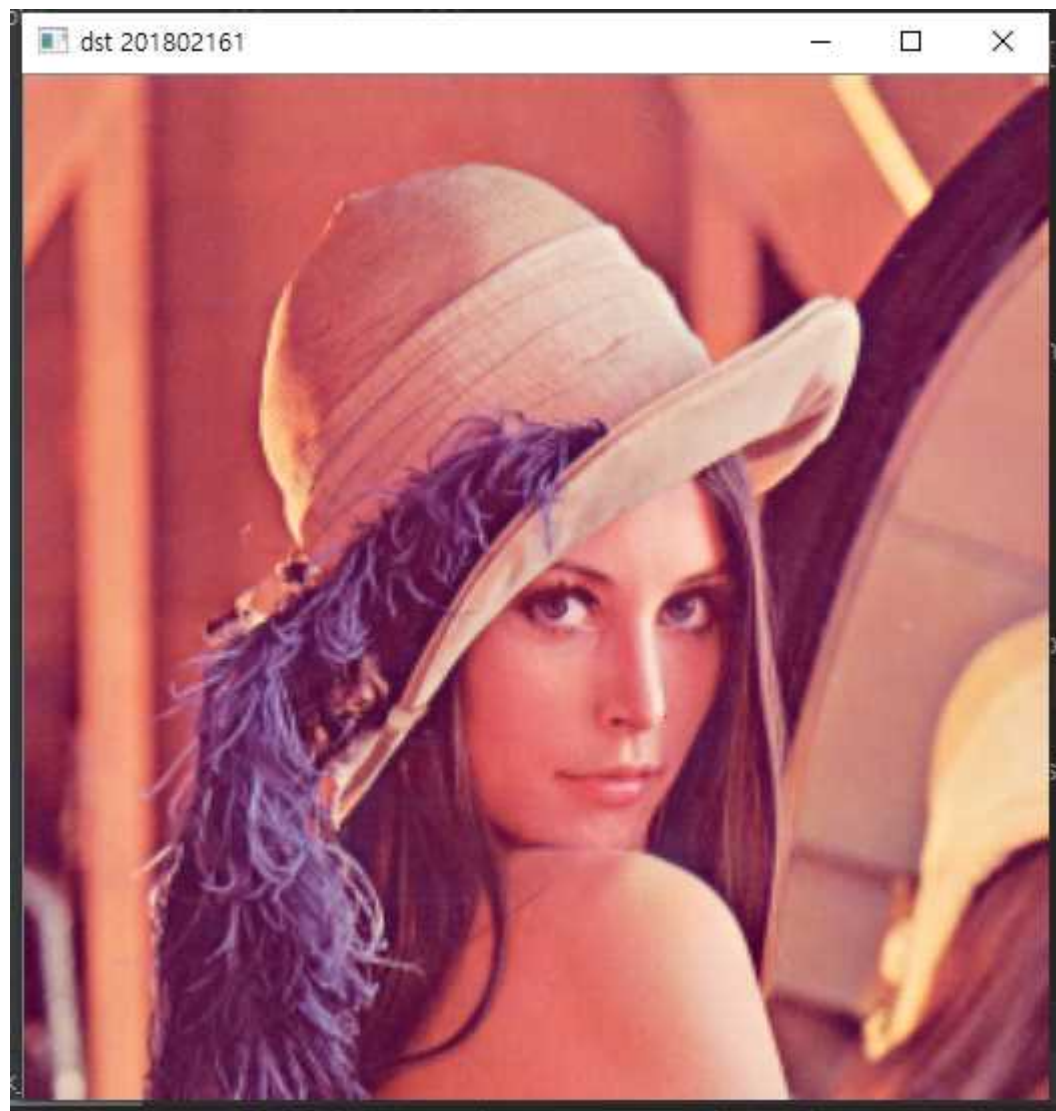
3. 이미지

* img_point

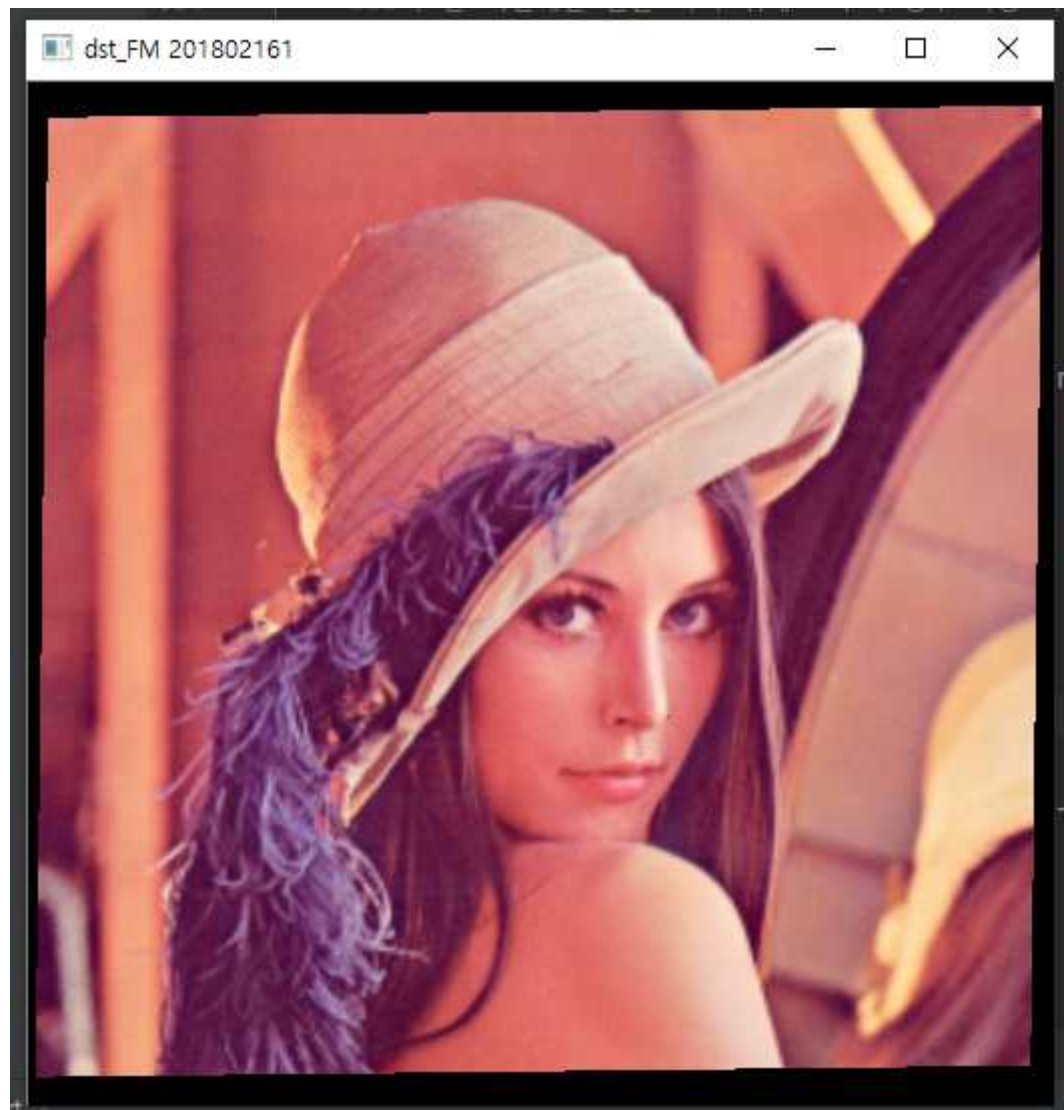
 img_point 201802161



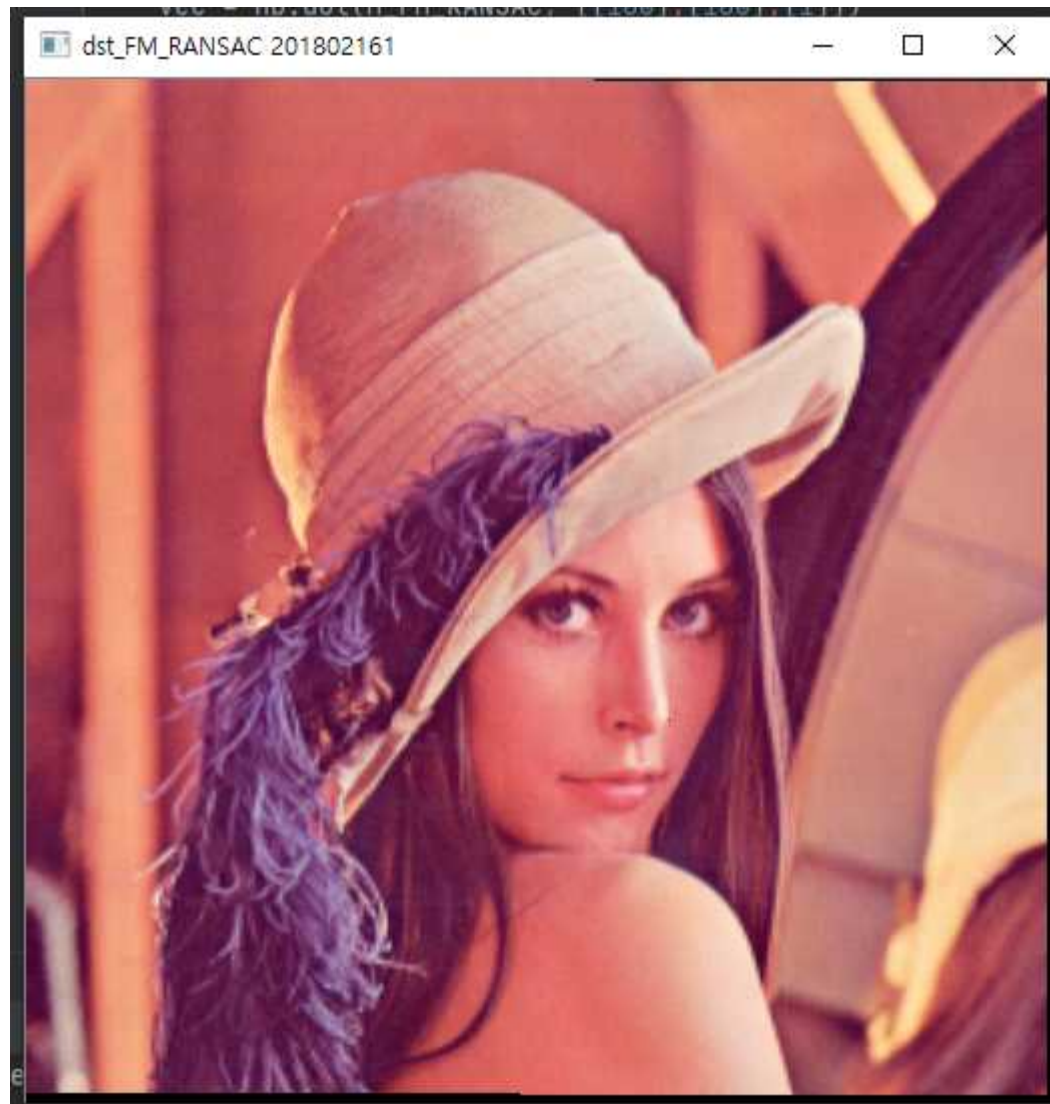
* dst



* dst_FM



* dst_FM_RNASAC



```
C:\Users\ChoEunBin\.conda\envs\homework\python.exe C:/Users/ChoEunBin/PycharmProjects/[C6]201802161_조은빈_7주차_과제/keypoint_check_report.py
M
[[2 0 0]
 [0 2 0]
 [0 0 1]]
M 역행렬
[[0.5 0. 0. ]
 [0. 0.5 0. ]
 [0. 0. 1. ]]
```

```
No RANSAC distance
point : 317 319
3.1622776601683795
Use RANSAC distance
point : 320 321
1.0
```

```
No RANSAC M
[[ 1.94708535 -0.02389899 11.32937122]
 [-0.02289127  1.88184856 19.09427529]
 [ 0.          0.          1.          ]]
RANSAC M
[[ 2.00000000e+00 -1.99840144e-15  1.00000000e+00]
 [ 5.57880056e-03  1.98864316e+00  1.20103606e+00]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

4. 느낀점

6주차 코드를 가져오면 되기 때문에 7주차 과제 자체에 대한 부담은 적었으나 6주차 코드를 완성하지 못하면 7주차도 같이 감점된다는 사실에 부담이 되었다.

5. 과제 난이도

보통이다.