



영상처리

13 주차 보고서

컴퓨터융합학부

202102699 정민경



1. 과제 설명

함수이름	get_roi_coordinates()
코드설명	<p>인자로 넘겨받은 image 의 각 꼭짓점좌표를 구하는 함수이다.</p> <p>이때 image 는 아직 변환이 일어나지 않은 즉, 변환을 시키려는 image 이고, 꼭짓점은 이미지 틀의 꼭짓점 좌표가 아닌 실제 Lena 이미지가 있는 사진에서의 꼭짓점좌표이다.</p> <p>인자로 받은 이미지의 각각의 모서리를 탐색한 후 모서리와 만나는 점이 꼭짓점좌표가 된다.</p> <p>이미지는 pixel 이기 때문에 우리가 생각했을 때는 한점이 만난다 생각할 수 있지만, 실제로 디버깅을 해보면 1~4개정도 만나는 것을 볼 수 있다. 이러한 부분에서는 그 중 1개의 점을 아무거나 사용해도 값이 비슷하기 때문에 아무값이나 하나의 좌표만을 남겨놓았다.</p> <p>내가 구현한 코드를 설명하면 나는 아래 코드와 같이 위쪽, 오른쪽, 아래쪽, 왼쪽 모서리를 탐색하면서 모서리와 만나는 점의 좌표를 모두 찾아 temp 라는 임시배열에 저장해주었다. 모서리와 만난다는 조건은 그 좌표의 값이 0이 아닐때 우리가 보고있는 레나이미지의 좌표임을 확인할 수 있다. 만약 가장 바깥쪽 모서리를 돌았을 때 만나는 점이 하나도 없었다면 그 모서리의 바로 안쪽 모서리를 돌면서 계속 반복해주었다. 만약 이렇게 돌다가 temp배열의 size 가 0이 아니면, 즉 temp 배열에 하나라도 들어갔다면 반복문을 종료해주고, 우리가 return 할 coordinates_list 에 저장해주었다. 그리고 이미지의 틀과 이미지가 같을 때는 모서리의 모든 좌표가 임시배열에 저장되게 된다. 이러한 점을 커버하고자 임시배열의 길이와 모서리의 길이가 동일하다면, 즉 모서리의 모든 좌표가 임시배열에 저장되어있다면 가장 마지막 원소도 최종배열에 저장하였다.</p> <p>위와 같은 방법을 4가지 방향에 적용시켜 하나의 coordinates_list 를 반환해줄도록 하였다.</p>
이미지	<pre> h, w = src.shape coordinates_list = [] # 4개의 좌표값을 도출해야 함. # 위쪽 모서리 탐색 temp = [] for i in range(h): for j in range(w): if (src[i][j] != 0): temp.append((i, j)) if (len(temp) != 0): break coordinates_list.append(temp[0]) if ((len(temp) - 1) == (w - 1)): coordinates_list.append(temp[len(temp)-1]) # 오른쪽 모서리 탐색 temp = [] for i in range(w-1, -1, -1): for j in range(h): if (src[j][i] != 0): temp.append((j, i)) if (len(temp) != 0): break coordinates_list.append(temp[0]) if ((len(temp) - 1) == (h - 1)): coordinates_list.append(temp[len(temp) - 1]) </pre>

```
# 아래쪽 모서리 탐색
temp = []
for i in range(h-1, -1, -1):
    for j in range(w):
        if (src[i][j] != 0):
            temp.append((i, j))
    if (len(temp) != 0):
        break

coordinates_list.append(temp[0])
if ((len(temp) - 1) == (w - 1)):
    coordinates_list.append(temp[len(temp) - 1])

# 왼쪽 모서리 탐색
temp = []
for i in range(w):
    for j in range(h):
        if (src[j][i] != 0):
            temp.append((j, i))
    if (len(temp) != 0):
        break

coordinates_list.append(temp[0])
if ((len(temp) - 1) == (h - 1)):
    coordinates_list.append(temp[len(temp) - 1])

return coordinates_list
```

함수이름	get_max_min_coordinates()
코드설명	<p>get_roi_coordinates 함수에서 변환되기 전의 이미지 꼭짓점 좌표를 구했다면 이 get_max_min_coordinate 함수에서는 변환된 이미지의 크기를 구하기 위한 row 와 col 각각의 max, min 값을 반환하는 함수이다.</p> <p>변환된 이미지의 크기를 구하는 방법은 $row = row_max - row_min$ 이고, $col = col_max - col_min$ 이다. 그렇기 때문에 이 함수에서 row, col 각각의 max, min 값을 구하는 것이다.</p> <p>이전 get_roi_coordinates 함수에서 구한 좌표에서 이 함수의 인자로 넘어올 때 중복을 제거해 넘어오게 된다. 그렇기 때문에 get_roi_coordinates 함수에서 중복 이 생겼더라도 get_max_min_coordinates 함수에서는 중복을 고려하지 않아도 괜찮은 것이다.</p> <p>그래서 get_roi_coordinates 에서 구한 roi_coordinate 좌표를 처음부터 끝까지 돌면서 역행렬과 행렬곱을 수행한다. 그렇게 되면 M에 의해 변환된 좌표값을 알 수 있다.</p> <p>그 후 np.min과 np.max 함수를 사용해서 각각의 max, min 값을 찾아준 뒤 row_max, row_min, col_max, col_min 값을 구해주었다.</p> <p>이 때 max 값은 np.ceil 함수를 통해 올림을 해주었고, min 값은 np.floor 함수를 통해 내림을 해주었다. 이렇게 나눠 수행한 이유는 언제나 최댓값을 가지기 위해 이러한 연산을 수행해주었다. Max값이 1.3이라면 2로 올려주어야 모든 값을 커버할 수 있고, min 값이 4.3이라면 4로 내림을 해주어야 모든 값을 커버할 수 있다.</p> <p>이렇게 구한 row, col 각각의 max, min 값을 return 해주었다.</p>
이미지	<pre> # dst shape 구하기 cor_transform = [] for i in range(len(roi_coordinates)): row, col = roi_coordinates[i] P = np.array([[col], [row], [1]]) P_dst = np.dot(M, P) cor_transform.append((P_dst[0][0], P_dst[1][0])) # Original에서 M에 의해 변환된 좌표의 최대 최소 범위 파악 cor_transform = np.array(cor_transform) # 추출한 좌표들을 기반으로 행의 최소, 최대값 열의 최소, 최댓값을 추출 # row_max = ??? # row_min = ??? # col_max = ??? # col_min = ??? min_value = np.min(cor_transform, axis=0) max_value = np.max(cor_transform, axis=0) row_max = int(np.ceil(max_value[0])) row_min = int(np.floor(min_value[0])) col_max = int(np.ceil(max_value[1])) col_min = int(np.floor(min_value[1])) return row_max, row_min, col_max, col_min </pre>

함수이름	backward()
코드설명	<p>지금까지 위에서 구현한 함수 두개를 사용해서 변환된 이미지에서의 최대 최소값을 구해준다. 그 후 이미지의 size 를 다시 정해주는데 row(=h_) = row_max - row_min 으로, col(=w_) = col_max - col_min 으로 계산해주었다.</p> <p>그 후 h_ * w_ 만큼을 돌면서, 즉 우리가 변환시키는 이미지의 프레임의 모든 픽셀을 돌면서 backward warping 으로 각각의 pixel 값들을 구해주었다. 이 때 P의 좌표는 프레임의 각각의 index 에다가 col_min 또는 row_min 을 더해 이미지를 우리가 만들어준 프레임의 중앙으로 움직여주는 효과를 나타내주었다.</p> <p>이렇게 변환된 값은 정수가 아닐 수 있으므로 bilinear interpolation 을 사용해 값을 채워주었다.</p> <p>나머지 backward warping 은 실습 때 사용했던 backward warping 를 참고해 구현하였다.</p>
이미지	<pre> print('< backward >') print('M') print(M) h, w = src.shape # M 역행렬 구하기 M_inv = np.linalg.inv(M) print('M_inv') print(M_inv) # 중복 제거 roi_coordinates = list(set(get_roi_coordinates(src))) (col_max, col_min_, row_max, row_min) = get_max_min_coordinates(roi_coordinates, M) ##### # TODO # TODO 2.2 결과 이미지의 크기 구하기 ##### # h_ = ??? # w_ = ??? h_ = row_max - row_min w_ = col_max - col_min dst = np.zeros((h_, w_)) for row in range(h_): for col in range(w_): </pre>

```

P_dst = np.array([
    [col + col_min],
    [row + row_min],
    [1]
])

# original 좌표로 매핑
P = np.dot(M_inv, P_dst)
# src_col = ???
# src_row = ???
src_col = P[0, 0]
src_row = P[1, 0]
# bilinear interpolation

src_col_right = int(np.ceil(src_col))
src_col_left = int(src_col)

src_row_bottom = int(np.ceil(src_row))
src_row_top = int(src_row)

# index를 초과하는 부분에 대해서는 값을 채우지 않음
if src_col_right >= w or src_row_bottom >= h or src_col_left < 0 or src_row_top < 0:
    continue

# intensity = ???
s = src_col - src_col_left
t = src_row - src_row_top

intensity = (1 - s) * (1 - t) * src[src_row_top, src_col_left] \
    + s * (1 - t) * src[src_row_top, src_col_right] \
    + (1 - s) * t * src[src_row_bottom, src_col_left] \
    + s * t * src[src_row_bottom, src_col_right]

dst[row, col] = intensity

dst = dst.astype(np.uint8)

print('dst shape : {}'.format(dst.shape))
print('dst min : {} dst max : {}'.format(np.min(dst), np.max(dst)))

return dst




```

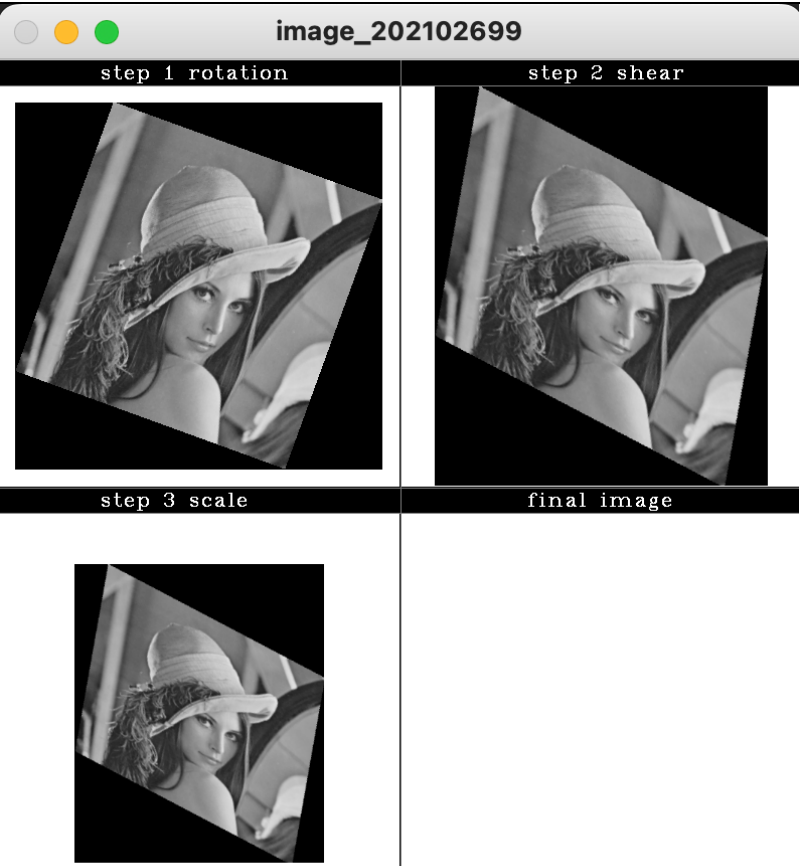
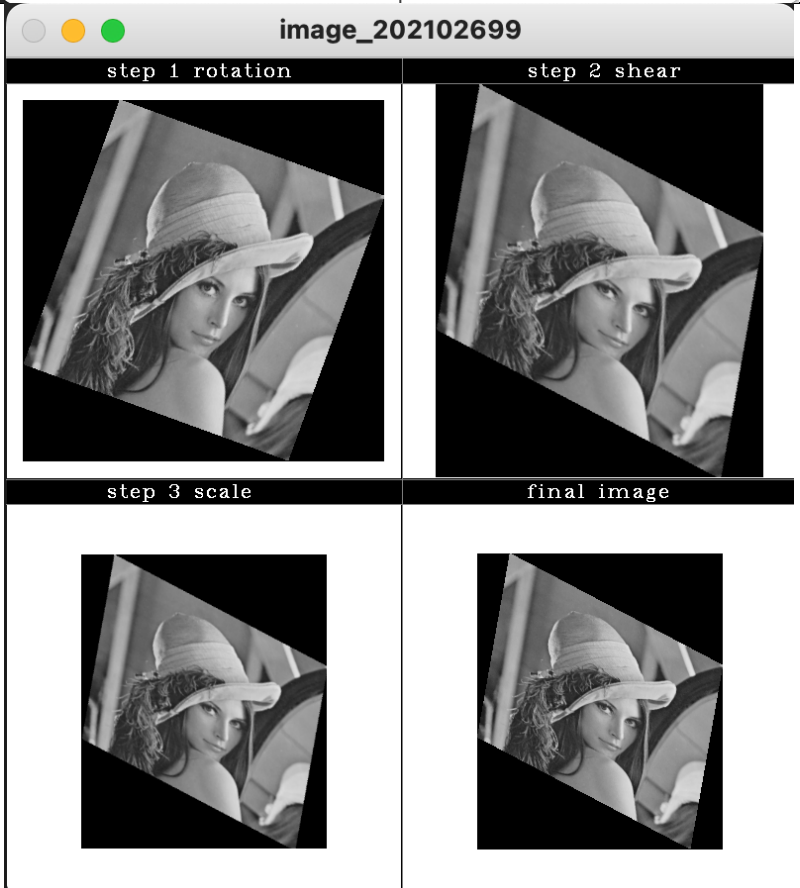
함수이름	generate_rotation()
코드설명	이미지를 회전시켜주는 행렬을 return 해주는 함수이다. 이론시간 및 실습시간에서 배운 수식을 np.array 를 사용해서 구현해주었다.
이미지	<pre> def generate_rotation(degree): # M_ro = ??? M_ro = np.array([[np.cos(np.deg2rad(degree)), -np.sin(np.deg2rad(degree)), 0], [np.sin(np.deg2rad(degree)), np.cos(np.deg2rad(degree)), 0], [0, 0, 1]]) return M_ro </pre>

함수이름	generate_scaling()
코드설명	Scale 을 변환해주는 행렬을 return 해주는 함수이다. 이론시간 및 실습시간에서 배운 수식을 np.array 를 사용해서 구현해주었다.
이미지	<pre> def generate_scaling(x_scaling, y_scaling): # M_sc = ??? M_sc = np.array([[x_scaling, 0, 0], [0, y_scaling, 0], [0, 0, 1]]) return M_sc </pre>

함수이름	generate_shearing()
코드설명	shearing 할 행렬을 return 해주는 함수이다. 이론시간 및 실습시간에서 배운 수식을 np.array 를 사용해서 구현해주었다.
이미지	<pre> def generate_shearing(x, y): # M_sh = ??? M_sh = np.array([[1, x, 0], [y, 1, 0], [0, 0, 1]]) return M_sh </pre>

2. 결과 이미지

Step 1	<div data-bbox="526 291 1300 779"><div data-bbox="526 291 1300 347">image_202102699</div><div data-bbox="526 347 911 376">step 1 rotation</div><div data-bbox="541 383 896 734"></div><div data-bbox="526 748 911 779">step 3 scale</div><div data-bbox="911 347 1300 376">step 2 shear</div><div data-bbox="911 748 1300 779">final image</div></div>
Step 2	<div data-bbox="526 1158 1300 1646"><div data-bbox="526 1158 1300 1214">image_202102699</div><div data-bbox="526 1214 911 1243">step 1 rotation</div><div data-bbox="541 1249 896 1601"></div><div data-bbox="526 1615 911 1646">step 3 scale</div><div data-bbox="911 1214 1300 1243">step 2 shear</div><div data-bbox="946 1249 1268 1615"></div><div data-bbox="911 1615 1300 1646">final image</div></div>

<div data-bbox="260 629 359 667" data-label="Text"><p>Step 3</p></div>	<div data-bbox="512 192 1316 1055" data-label="Image">A sequence of four grayscale images of a woman wearing a wide-brimmed hat, arranged in a 2x2 grid. The top row shows 'step 1 rotation' and 'step 2 shear'. The bottom row shows 'step 3 scale' and 'final image'. The images are presented within a window titled 'image_202102699' with standard macOS window controls (red, yellow, green buttons).</div>
<div data-bbox="260 1532 359 1570" data-label="Text"><p>Result</p></div>	<div data-bbox="512 1099 1316 1984" data-label="Image">A sequence of four grayscale images of a woman wearing a wide-brimmed hat, arranged in a 2x2 grid. The top row shows 'step 1 rotation' and 'step 2 shear'. The bottom row shows 'step 3 scale' and 'final image'. The images are presented within a window titled 'image_202102699' with standard macOS window controls (red, yellow, green buttons).</div>