이름 : 강시온
학번 : 201802045
학과 : 컴퓨터공학과
구현 코드

```python
def my_padding(src, pad_shape, pad_type='zero'):
    (h, w, c) = src.shape
    (p_h, p_w) = pad_shape
    pad_img = np.zeros((h+2*p_h, w+2*p_w, c), dtype=np.float32)
    pad_img[p_h:p_h+h, p_w:p_w+w] = src

    if pad_type == 'repetition':
        print('repetition padding')
        #up
        for i in range(0, p_h):
            pad_img[i, p_w:p_w + w] = pad_img[p_h, p_w:p_w + w]
        #down
        for i in range(p_h+h, 2*p_h+h):
            pad_img[i, p_w:p_w + w] = pad_img[p_h+h-1, p_w:p_w + w]
        #left
        for i in range(0, p_w):
            pad_img[0:2*p_h+h, i] = pad_img[0:2*p_h+h, p_w]
        #right
        for i in range(p_w+w, 2*p_w+w):
            pad_img[0:2*p_h+h, i] = pad_img[0:2*p_h+h, p_w+w-1]

    else:
        print('zero padding')

    return pad_img
```

```python
def my_get_Gaussian2D_mask(msize, sigma=1):
    y, x = np.mgrid[-(msize//2):msize//2+1, -(msize//2):msize//2+1]

    # 2차 gaussian mask 생성
    gaus2D = (np.exp(-((x*x)+(y*y))/(2*sigma*sigma))/(2*np.pi*sigma*sigma))
    # mask의 총 합 = 1
    gaus2D /= np.sum(gaus2D)

    return gaus2D
```

```python
def backward_filterling(img1, img2, M):
    '''
    :param img1: 변환시킬 이미지
    :param img2: 변환 목표 이미지
    :param M: 변환 matrix
    :return: 변환된 이미지
    '''
    h, w, c = img2.shape
    result = np.zeros((h, w, c))

    mask = my_get_Gaussian2D_mask(5, 0.5)
    m_h, m_w = mask.shape
    pad_img = my_padding(img1, (m_h // 2, m_w // 2))

    for row in range(h):
        for col in range(w):
            xy_prime = np.array([[col, row, 1]]).T
            xy = (np.linalg.inv(M)).dot(xy_prime)

            x_ = xy[0, 0]
            y_ = xy[1, 0]

            if x_ < 0 or y_ < 0 or (x_ + 1) >= w or (y_ + 1) >= h:
                continue

            for f_row in range(m_h):
                for f_col in range(m_w):
                    result[row, col, :] += my_bilinear(pad_img, x_ + f_row, y_ + f_col)*mask[f_row][f_col]

    return result
```

코드 설명

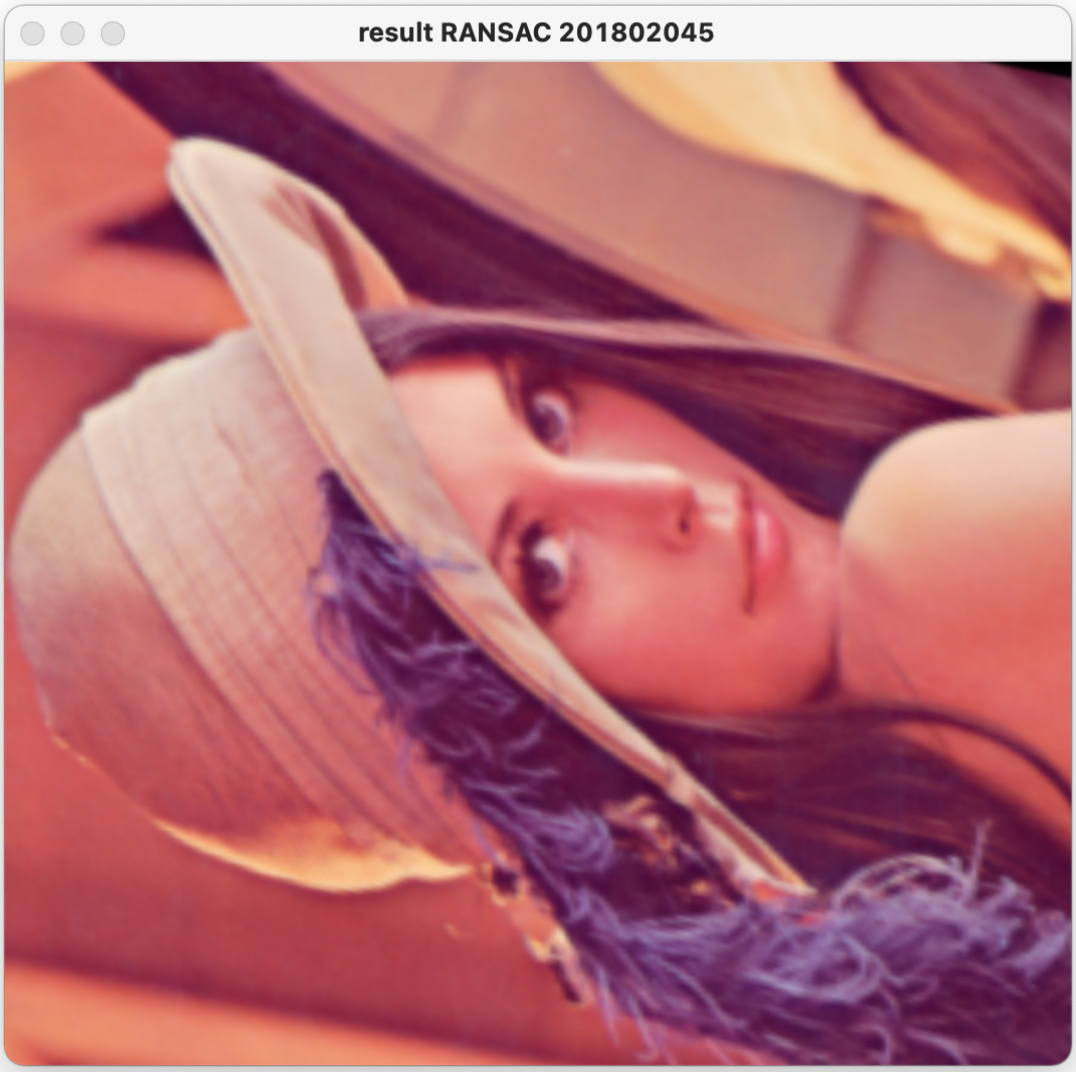
ransac을 구하는 코드들은 저번 과제에서 사용한 코드를 그대로 사용하여 제외하고 이번과제를 위해 새로 구현한 코드들을 위주로 설명.

My_padding()
Gaussian 필터링을 적용하기 위한 padding작업을 해주는 함수, 실제로 구현에서는 zero 패딩을 사용하였다.

My_get_gaussian2D_mask()
Gaussian 필터를 만들어서 반환해주는 함수, 필터의 크기와 sigma값을 인자로 받아 만들어줄 수 있다.

Backward_filtering()
이번과제의 핵심 코드인 backward_filtering이다.
먼저 위에서 설명한 함수를 이용하여 gaussian 필터를 만들어서 mask에 저장해둔다.
Mask의 크기만큼 반복하고 padding을 해야하므로 mask의 shape도 저장해둔다.
그리고 저번과제와 다르게 이번 과제는 변환된 이미지의 크기가 달라지므로 전체적으로 표현 할 수 있도록 원본이미지가 아닌 목표 이미지의 크기로 result를 만들어주었다.
먼저 저번시간에 구현한 backward코드처럼 ransac을 이용해서 구한 M의 역함수를 이용해 목표 이미지와 원본이미지의 매칭되는 점을 찾고 마지막 이중포문을 통해 mask의 사이즈만큼 주변 점들을 방문하면서 각각 bilinear 해준 값들을 mask를 곱해줘 필터링을 적용해준다.

결과 이미지



난이도
그동안 배운것들을 종합적으로 활용해야 풀 수 있는 문제여서 그동안 했던 것들을 다시한번 찾아보고 복습 할 수 있는 기회가 된 것 같습니다.