HW3

Method

1. Preprocessing

   I chose an image of IU, but the resolution is quite high, so I applied a box filter three times to blur the edges.

```python
def box_filter(image, filter_size=5):
    '''
    apply box filter to blur the image
    '''
    new_img = np.zeros(image.shape)
    pad_size = filter_size // 2
    padded_img = np.pad(image, ((pad_size, pad_size), (pad_size, pad_size)), 'constant')
    for i in range(pad_size, image.shape[0] - pad_size):
        for j in range(pad_size, image.shape[1] - pad_size):
            new_img[i, j] = np.mean(padded_img[i - pad_size:i + pad_size + 1, j - pad_size:j + pad_size + 1], axis=(0, 1))
    return new_img.astype(np.uint8)
```

2. Laplacian filter – spatial

   I tried two Laplacian filters of size 3*3.

```python
laplacian_filter1 = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]]).astype(np.float64)
laplacian_filter2 = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]]).astype(np.float64)
```

   I convert it to np.float64 and clip the value between 0 and 255 to avoid overflow.

```python
def Laplacian_filter_saptial(image, filter):
    image = image.astype(np.float64)
    pad_size = 1
    new_img = np.zeros(image.shape)
    # same padding
    padded_img = np.pad(image, ((pad_size, pad_size), (pad_size, pad_size)), 'constant')

    for i in range(pad_size, image.shape[0] - pad_size):
        for j in range(pad_size, image.shape[1] - pad_size):
            new_img[i, j] = np.clip(np.sum(padded_img[i - pad_size:i + pad_size + 1, j - pad_size:j + pad_size + 1] * filter, axis=(0, 1)), 0, 255)
    return new_img.astype(np.uint8)
```

3. Laplacian filter – frequency

   Convert filter and image to frequency domain and multiply them.

```python
def Laplacian_filter_frequency(image, filter):
    image = image.astype(np.float64)

    # pad to avoid weird edge values
    padded_image = np.pad(image, [(1, 1), (1, 1)], mode='constant')

    image_fft = fft2(padded_image)
    filter_fft = fft2(filter, s=padded_image.shape)
    new_img_fft = image_fft * filter_fft
    new_img = ifft2(new_img_fft)

    # deal with complex number
    new_img = np.abs(new_img)
    new_img = np.clip(new_img, 0, 255)
    return new_img.astype(np.uint8)
```

Result

Original

Spatial domain - filter 1/2 (The difference in hair is obvious)

Frequency domain – filter 1/2

Feedback

In this lab, I implemented Laplacian sharpening in both spatial and frequency domain. There are a few details during the implementation (padding / clipping), and you might get some imperfect result if you miss one of them. I searched on the internet and few of them will use fft_shift to transform the frequency 0 to the middle of the image after trasforming to frequecy domain. I did some experiment on this function, and it turned out that it has little effect on my result.