INT3404E 20 - Image Processing: Homeworks 2

Hoang Duc Anh

21st April 2024

1 Goals

The goals of this homework are:

1.1 To achieve a comprehensive understanding of how basic image filters operate

1.2 To gain a solid understanding of the Fourier Transform (FT) algorithm

2 Details

2.1 Image filtering

Function implementation:

```
pad_size = filter_size // 2
    padded_img = [[img[min(max(i-pad_size, 0), img.shape[0]-1)][min(max(j-pad_size, 0), img.shape
        [1]-1)]
                   for j in range(img.shape[1] + 2*pad_size)]
                  for i in range(img.shape[0] + 2*pad_size)]
    return padded_img
```

```
def mean_filter(img, filter_size=3):
    padded_img = padding_img(img, filter_size)
    smoothed_img = np.zeros_like(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            neighborhood = [padded_img[i+k][j+l] for k in range(filter_size) for l in range(
                filter_size)]
            smoothed_img[i, j] = sum(neighborhood) / (filter_size * filter_size)
    return smoothed_img
```

```
def median_filter(img, filter_size=3):
    padded_img = padding_img(img, filter_size)
    smoothed_img = np.zeros_like(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            neighborhood = [padded_img[i+k][j+l] for k in range(filter_size) for l in range(
                filter_size)]
            neighborhood.sort()
            smoothed_img[i, j] = neighborhood[len(neighborhood) // 2]
    return smoothed_img
```

The output of these 2 filter are shown below:

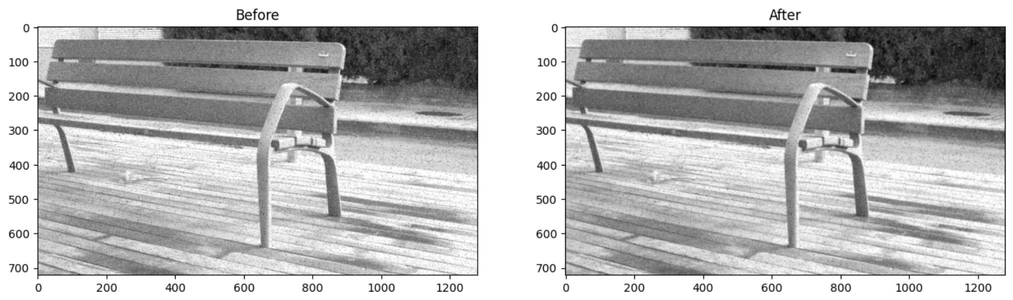


Figure 1: Mean filter result.

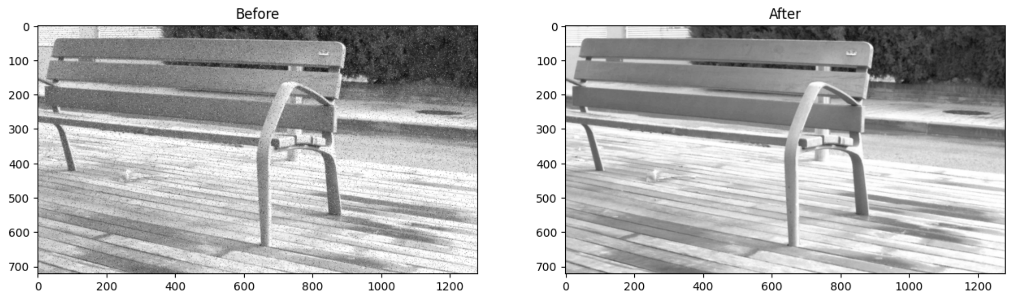


Figure 2: Median filter result.

The PSNR score for mean filter approximately 31.6 while the score for median filter is 37. Higher score stands for better image when applying the filter. Therefore, the median should be chosen

2.2 Fourier transform

2.2.1 1D Fourier transform

```
def DFT_slow(data):
  N = len(data)
  n = np.arange(N)
  k = n.reshape((N, 1))
  e = np.exp(-2j * np.pi * k * n / N)

  X = np.dot(e, data)

  return X
```

2.3 2D Fourier transform

```
def DFT_2D(gray_img):
    row_fft = np.array([DFT_slow(row) for row in gray_img])
    row_col_fft = np.array([DFT_slow(col) for col in row_fft.T]).T

    return row_fft, row_col_fft
```

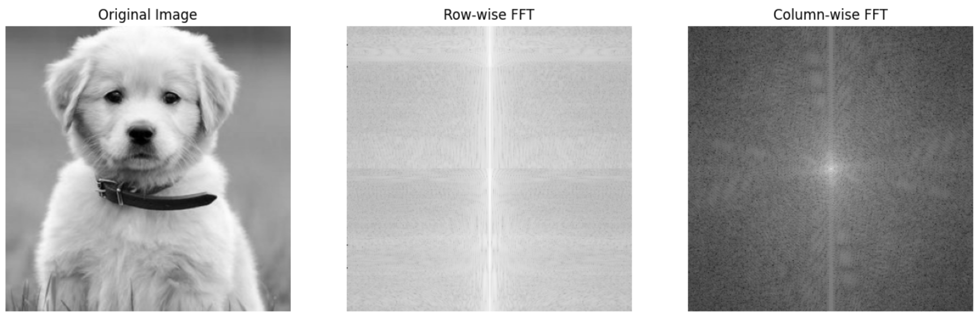
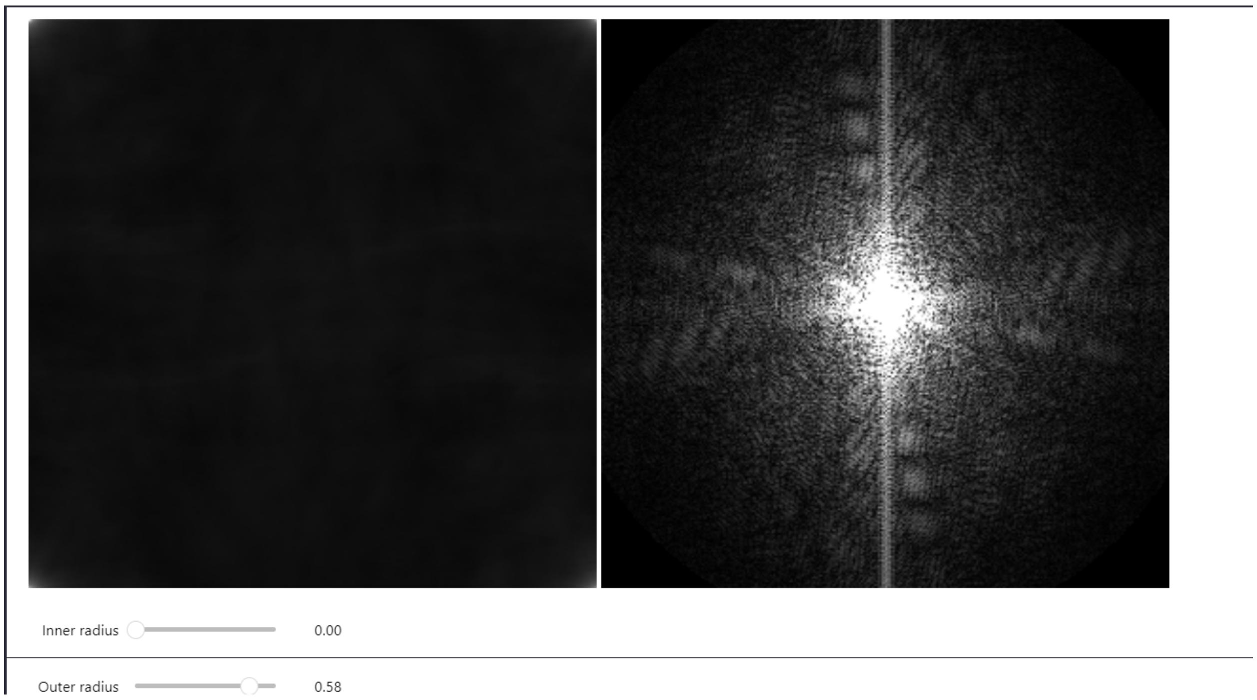


Figure 3: 2D FFT result.

2.4 Frequency Removal Procedure

```
def filter_frequency(orig_img, mask):
  f_org_image = np.fft.fft2(orig_img)
  fshift = np.fft.fftshift(f_org_image)
  fshift_masked = fshift * mask
  fshift_masked = np.abs(fshift_masked)
  f_ishift = np.fft.ifftshift(fshift_masked)
  img_back = np.fft.ifft2(f_ishift)
  img_back = np.abs(img_back)

  return fshift_masked, img_back
```



Inner radius 0.00

Outer radius 0.58

Figure 4: 2D Frequency Removal.

2.4.1 Creating a Hybird Image

```python
def create_hybrid_img(img1, img2, r):
  f_img1 = np.fft.fftshift(np.fft.fft2(img1))
  f_img2 = np.fft.fftshift(np.fft.fft2(img2))

  H, W = img1.shape
  y, x = np.ogrid[:H, :W]
  center = (H // 2, W // 2)
  mask = (x - center[1])**2 + (y - center[0])**2 <= r**2

  f_hybrid_img = f_img1 * mask + f_img2 * ~mask

  hybrid_img = np.abs(np.fft.ifft2(np.fft.ifftshift(f_hybrid_img)))

  return hybrid_img
```

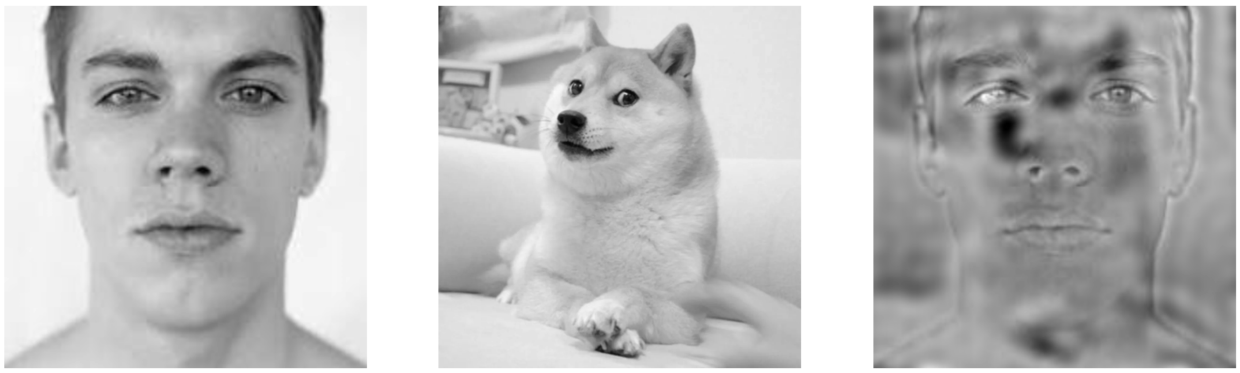


Figure 5: Hybrid Image.