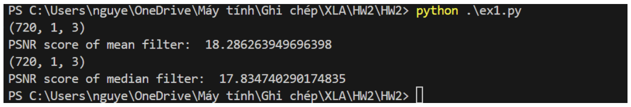
INT3404E 20 - Image Processing: Homeworks 2

Nguyen Khoa Dang

Repo : LINK

1 Image Filtering

Although mean filter get a better score, median filter image is better looking images we should choose it



```
PS C:\Users\nguye\OneDrive\Máy tính\Ghi chép\XLA\HW2\HW2> python .\ex1.py
(720, 1, 3)
PSNR score of mean filter:  18.286263949696398
(720, 1, 3)
PSNR score of median filter:  17.834740290174835
PS C:\Users\nguye\OneDrive\Máy tính\Ghi chép\XLA\HW2\HW2> 
```

Figure 1: The metrics

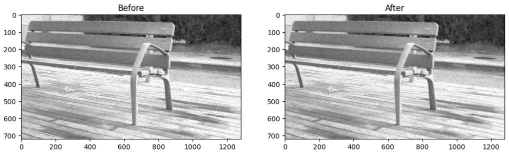


Figure 2: Mean Filter

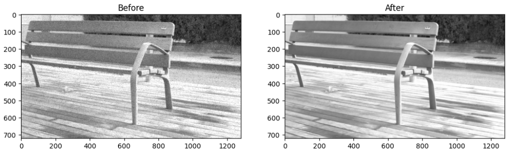


Figure 3: Median Filter

Please read the ex1.python file for detailed comment on the code

2 Fourier Transform

2.1 1D and 2D

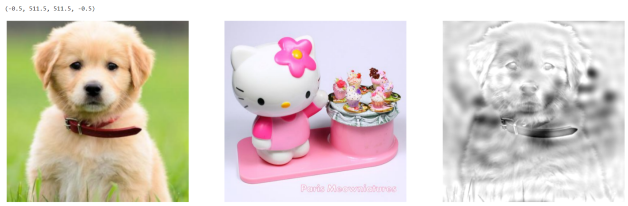
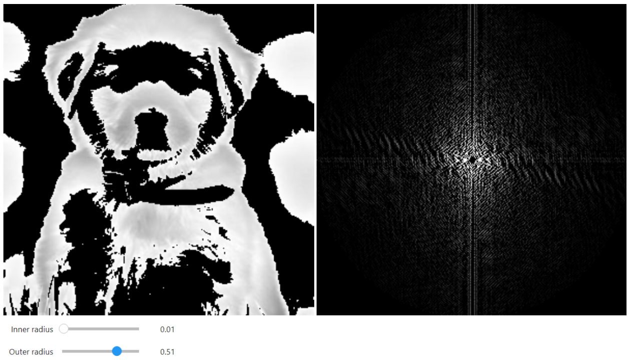


Figure 4: 2D Fourier Transform

```
def DFT_2D(gray_img):
    # Ensure image has dtype of float for calculations (assuming uint8)
    img_float = gray_img.astype(np.float32)

    # Perform row-wise FFT
    row_fft = np.fft.fft(img_float, axis=1)

    # Perform column-wise FFT on the row-wise transformed image
    col_fft = np.fft.fft(row_fft, axis=0)

    # Ensure complex dtype for output
    return col_fft.astype(np.complex_), row_fft.astype(np.complex_)

def DFT_slow(data):
  # You need to implement the DFT here
  # Ensure data has dtype of float for calculations (assuming float or int)
  data_float = data.astype(np.float32)

  # Perform DFT using NumPy's FFT function
  DFT = np.fft.fft(data_float)

  # Ensure complex dtype for output
  return DFT.astype(np.complex_)
```

2.2 Frequency Removal Procedure

```
    def filter_frequency(orig_img, mask):
    """
    You need to remove frequency based on the given mask.
    Params:
    orig_img: numpy image
    mask: same shape with orig_img indicating which frequency hold or remove
    Output:
    f_img: frequency image after applying mask
    img: image after applying mask
    """
    # You need to implement this function
    # Convert image to grayscale if it's RGB
    if len(orig_img.shape) == 3:
        orig_img = orig_img.mean(axis=2)

    # 1. Transform using fft2
    f = np.fft.fft2(orig_img)

    # 2. Shift frequency coefs to center using fftshift
    f_shifted = np.fft.fftshift(f)

    # 3. Apply mask to filter frequencies
    filtered_f = f_shifted * mask

    # 4. Shift frequency coefs back using ifftshift
    filtered_f_ishifted = np.fft.ifftshift(filtered_f)

    # 5. Invert transform using ifft2
    img = np.fft.ifft2(filtered_f_ishifted).real

    # Clip to avoid potential artifacts
    img = np.clip(img, 0, 255)

    return filtered_f.astype(np.float32), img.astype(np.uint8)
```

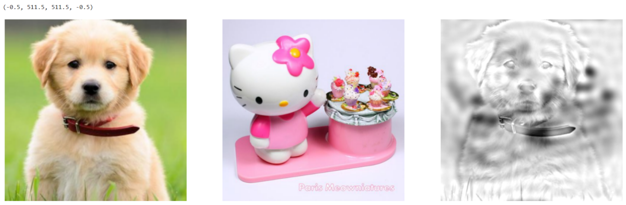


Inner radius 0.01

Outer radius 0.51

Figure 5: FRC

2.3 Creating a Hybrid Image



```
(-0.5, 511.5, 511.5, -0.5)
```

Figure 6: Enter Caption

```
def create_hybrid_img(img1, img2, r):
    """
    Create hydrid image
    Params:
    img1: numpy image 1
    img2: numpy image 2
    r: radius that defines the filled circle of frequency of image 1. Refer to the homework title to know more.
    """
    # You need to implement the function

    # Ensure images have the same dimensions
    if img1.shape != img2.shape:
        raise ValueError("Input images must have the same dimensions.")
```

```python
    # Convert images to grayscale if they are RGB
    if len(img1.shape) == 3:
        img1 = img1.mean(axis=2)
        img2 = img2.mean(axis=2)

    # 1. Transform using fft2
    f1 = np.fft.fft2(img1)
    f2 = np.fft.fft2(img2)

    # 2. Shift frequency coefs to center using fftshift
    f1_shifted = np.fft.fftshift(f1)
    f2_shifted = np.fft.fftshift(f2)

    # 3. Create a mask based on the given radius (r) parameter
    rows, cols = img1.shape
    center_x = int(cols / 2)
    center_y = int(rows / 2)
    mask = np.zeros((rows, cols))
    y, x = np.ogrid[:rows, :cols]
    mask = np.sqrt((x - center_x)**2 + (y - center_y)**2) <= r

    # 4. Combine frequency of 2 images using the mask
    hybrid_f = f1_shifted * mask + f2_shifted * (1 - mask)

    # 5. Shift frequency coefs back using ifftshift
    hybrid_f_ishifted = np.fft.ifftshift(hybrid_f)

    # 6. Invert transform using ifft2
    hybrid_img = np.fft.ifft2(hybrid_f_ishifted).real

    # Clip to avoid potential artifacts
    hybrid_img = np.clip(hybrid_img, 0, 255)

    return hybrid_img.astype(np.uint8)
```