Exercise 2 Image Filters

March 16, 2018

** Xiangyi Cheng (xxc283)**

1 Concept and Backgroud

Discrete cosine transform (DCT), as discussed in Exericse 1, transfers image data from spatial space to frequential space. Different views of image information are given by implementing the transform. Once having the frequency basis functions, filtering is applied to the image by modifying the coeffiency of each basis to achieve the processing goal, such as image compression.

2 Implement

To develop the low-pass and high-pass filter, what we need to do is simply zeroing different zones of the DCT coefficients.

First step is to add libraies which will be used inside the code.

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from scipy.fftpack import dct,idct
        import cv2
        import math
```

Since only 1-D DCT and inverse cosine transform (iDCT) is provided by Python, it would be better to come up with our own 2-D DCT and iDCT by self-defined functions.

```
In [3]: # define the function to do dct and idct in 2D space
        def dct2(image):
                return dct(dct(image.T,norm='ortho').T,norm='ortho')

        def idct2(dctmatrix):
                return idct(idct(dctmatrix.T,norm='ortho').T,norm='ortho')
```

OpenCV is used to load and write the image. Then the defined function is called to get the 2-D DCT of the image so that the image data in frequential space is obtained.

```
In [28]: #img=cv2.imread('sydney.jpg',cv2.IMREAD_GRAYSCALE)
         img=cv2.imread('sydney.jpg',0)
         gray_img=cv2.imwrite('gray_sydney.jpg',img)
```

```
        # do the dct to transfer the image from spatial space into frequency space
        dct_img=dct2(img)
        m,n=dct_img.shape
```

Generating the low-pass filter is achieved by removing the high frequency zone. In this track, simply zeroing the coeffient in the high frequency is totally enough. As talked in exercise 1, the frequency is increasing by comparing each row and each column. Low frequency relatively exists in the top left corner of the matrix while high frequency appears in the bottom right corner.

Therefore, zeroing the row 10 to the end and column 10 to end of the coefficent matrix is to apply the low pass filter.

```
In [29]: low_filter=dct_img
         low_filter[10:,10:]=0
```

Inversing the processed matrix back to x,y space is the next step. Then we can compare the image with low-pass filter with the original image to see the differences between them.

```
In [30]: low_idct=idct2(low_filter)
         diff_low=abs(low_idct-img)
         cv2.imwrite('low_filtered_sydney.jpg',low_idct)
         cv2.imwrite('diff_low.jpg',diff_low)

Out[30]: True
```

Follow the basis idea of applying low-pass filter to obtain the high-pass filter. This time the coeffient of the lower frequency zone should be zero. Therefore, zero the row 0 to 100 and column 0 to 100.

```
In [31]: high_filter=dct_img
         high_filter[0:100,0:100]=0
         high_idct=idct2(high_filter)
         diff_high=abs(high_idct-img)
         cv2.imwrite('high_filtered_sydney.jpg',high_idct)
         cv2.imwrite('diff_high.jpg',diff_high)

Out[31]: True
```

To illustrate the results and differences between images, the processed images and differences are plotted out.

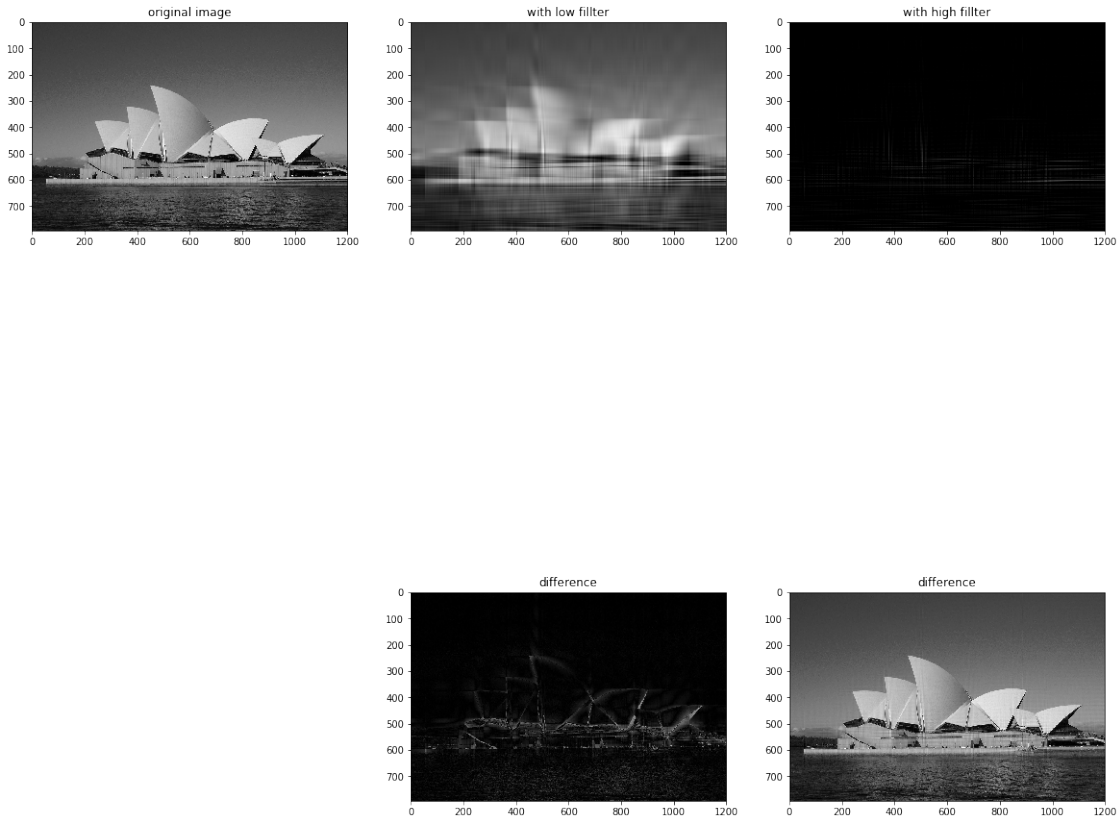
```
In [32]: # show the original image
         plt.figure(figsize=(20,20))
         original_img=cv2.imread('gray_sydney.jpg')
         original_img=cv2.cvtColor(original_img,cv2.COLOR_BGR2RGB)
         plt.subplot(2,3,1)
         plt.imshow(original_img)
         plt.title('original image')
```

```
# show the processed image by the low pass filter
low_filtered_sydney=cv2.imread('low_filtered_sydney.jpg')
low_filtered_sydney=cv2.cvtColor(low_filtered_sydney,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,2)
plt.imshow(low_filtered_sydney)
plt.title('with low fillter')

# show the difference between the orignal image and processed image applied low pass fi
diff_low=cv2.imread('diff_low.jpg')
diff_low=cv2.cvtColor(diff_low,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,5)
plt.imshow(diff_low)
plt.title('difference')


# show the processed image by the high pass filter
high_filtered_sydney=cv2.imread('high_filtered_sydney.jpg')
high_filtered_sydney=cv2.cvtColor(high_filtered_sydney,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,3)
plt.imshow(high_filtered_sydney)
plt.title('with high fillter')

# show the difference between the orignal image and processed image applied high pass f
diff_high=cv2.imread('diff_high.jpg')
diff_high=cv2.cvtColor(diff_high,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,6)
plt.imshow(diff_high)
plt.title('difference')
plt.show()
```



To have a clear sense that how the filters work, the size of zero zones are modified to 40 for the low-pass filter and 40 for the high-pass filter.

```
In [25]: # apply the low pass filter
         low_filter=dct_img
         low_filter[40:,40:]=0
         low_idct=idct2(low_filter)
         diff_low=abs(low_idct-img)
         cv2.imwrite('low_filtered_sydney.jpg',low_idct)
         cv2.imwrite('diff_low.jpg',diff_low)

         dct_img=dct2(img)
         m,n=dct_img.shape
         # apply the high pass filter
         high_filter=dct_img
         high_filter[0:40,0:40]=0
         high_idct=idct2(high_filter)
         diff_high=abs(high_idct-img)
         cv2.imwrite('high_filtered_sydney.jpg',high_idct)
         cv2.imwrite('diff_high.jpg',diff_high)

Out[25]: True
```

Plot the results and show the results.

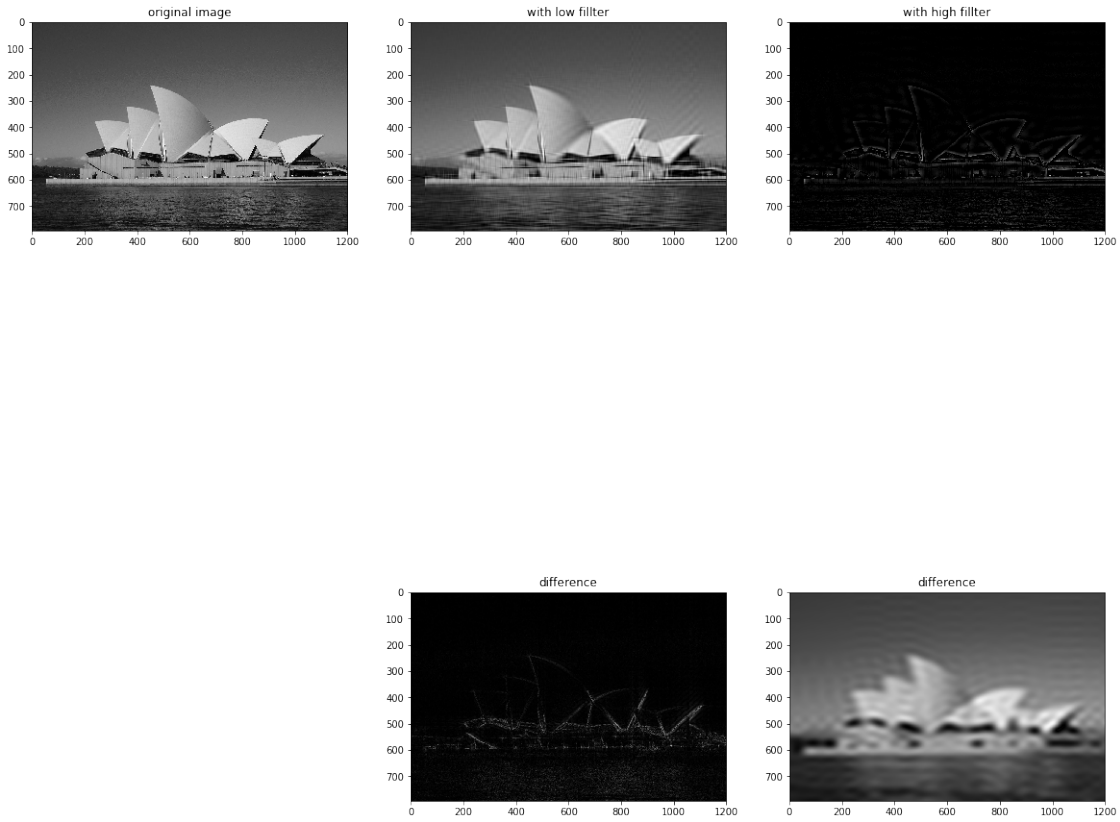
In [30]:
```
# show the original image
plt.figure(figsize=(20,20))
original_img=cv2.imread('gray_sydney.jpg')
original_img=cv2.cvtColor(original_img,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,1)
plt.imshow(original_img)
plt.title('original image')


# show the processed image by the low pass filter
low_filtered_sydney=cv2.imread('low_filtered_sydney.jpg')
low_filtered_sydney=cv2.cvtColor(low_filtered_sydney,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,2)
plt.imshow(low_filtered_sydney)
plt.title('with low fillter')

# show the difference between the orignal image and processed image applied low pass fi
diff_low=cv2.imread('diff_low.jpg')
dct_img=dct2(img)
m,n=dct_img.shape
diff_low=cv2.cvtColor(diff_low,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,5)
plt.imshow(diff_low)
plt.title('difference')


# show the processed image by the high pass filter
high_filtered_sydney=cv2.imread('high_filtered_sydney.jpg')
high_filtered_sydney=cv2.cvtColor(high_filtered_sydney,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,3)
plt.imshow(high_filtered_sydney)
plt.title('with high fillter')

# show the difference between the orignal image and processed image applied high pass f
diff_high=cv2.imread('diff_high.jpg')
diff_high=cv2.cvtColor(diff_high,cv2.COLOR_BGR2RGB)
plt.subplot(2,3,6)
plt.imshow(diff_high)
plt.title('difference')10
plt.show()
```



3 Extension

To get a band filter, only the information in the middle zone is kept.

```
In [39]: # apply the band pass filter
         band_filter=dct_img
         band_filter[0:20,0:20]=0
         band_filter[80:,80:]=0
         band_idct=idct2(band_filter)
         diff_band=abs(band_idct-img)
         cv2.imwrite('band_filtered_sydney.jpg',band_idct)
         cv2.imwrite('diff_band.jpg',diff_band)
```

```
Out[39]: True
```

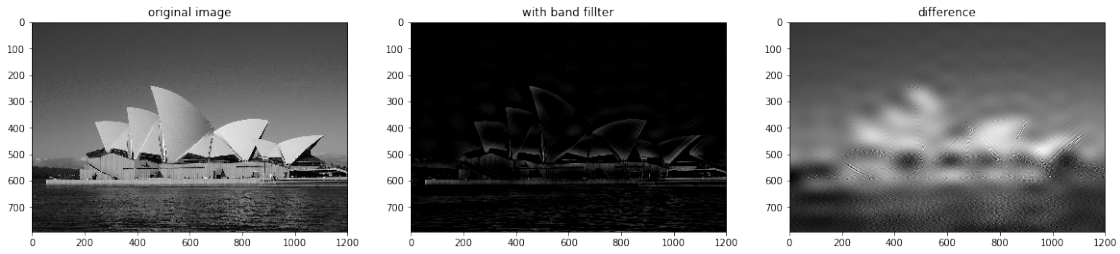
Show the result.

```
In [40]: # show the original image
         plt.figure(figsize=(20,20))
         original_img=cv2.imread('gray_sydney.jpg')
         original_img=cv2.cvtColor(original_img,cv2.COLOR_BGR2RGB)
```

```
plt.subplot(1,3,1)
plt.imshow(original_img)
plt.title('original image')


# show the processed image by the band pass filter
band_filtered_sydney=cv2.imread('band_filtered_sydney.jpg')
band_filtered_sydney=cv2.cvtColor(band_filtered_sydney,cv2.COLOR_BGR2RGB)
plt.subplot(1,3,2)
plt.imshow(band_filtered_sydney)
plt.title('with band fillter')

# show the difference between the orignal image and processed image applied band pass f
diff_band=cv2.imread('diff_band.jpg')
dct_img=dct2(img)
diff_band=cv2.cvtColor(diff_band,cv2.COLOR_BGR2RGB)
plt.subplot(1,3,3)
plt.imshow(diff_band)
plt.title('difference')
plt.show()
```



4 Conclusion and Analysis

The goal applying low-pass and high-pass filter to the images is achieved by doing 2-D DCT, zeroing certain zones of the matrix and then applying iDCT to the image data.

The results obviously express that the most of the information describing the images exist in low frequency. As shown in these two examples, the image processed by the low-pass filter whose dimensions are only 10x10 shows more blur than that applied the low-pass filter with 40x40 dimensions. Supprisingly, even only 10x10 dimensions are applied, the processed image still look not bad and can be identified easily compared to the one applied by the high-pass filter. And the high-pass filter can be used for edge detection since it keeps more information on the edges. Moreover, since human eyes have limitations on high frequency, some modifications such as removing the data from high frequency are not recognized by people. Therefore, image compression to reduce dimensions or decrease the storage space is practical and meaningful.