Project 1 Report and Questions

Report

The main purpose of the project is to make hybrid image, an image with tow interpretation for humans' visual perception. This project process is mainly implementing convolution from scratch to convolute any filter kernel with the image and then applying the algorithm for making hybrid images. To make a hybrid image, one of the images it convoluted with a low pass filter to extract its low frequencies content. The second image is convoluted with a high pass filter to extract its high frequencies content. The limit specifying the barrier between high frequencies and low frequencies is know as cut-off frequency. The cut-off frequency is not a standard limit for images, it is a tunable parameter for each two images to form their hybrid image.

To apply the previously stated algorithm, this python-based computer vision project begins by implementing convolution function, Gaussian filter formation function, zero padding function, reflective padding function and hybrid images formation function. The project is mainly separated into 2 related parts. In part 1, an image is read using openCV library and this convoluted with multiple filters. The filters used are box filter, identity filter, high pass filter by subtracting the low pass filtered image from the original one, Sobel filter and Laplacian filter. In part 2, once two aligned images are read, the first image is applied to a low pass filter (Gaussian) and the other image is applied to a high pass filter by subtracting the low pass filtered image from the original one. At the end, the visualization of the hybrid image is created by plotting the hybrid image in different distances by down-sampling the original hybrid image by different amounts to give the sense of far images which would eliminates the high frequency contents for humans' naked eyes.

0.1 Results

0.1.1 Part 1

In part 1, the image of "mona lisa" is used to test the convolution function with multiple filters. Figure (2) shows the results of identity filter and low pass filter (blurring filter) kernels applied on the image using the manually made convolution function. On the other hand, figure (3) shows the results of high pass filter kernels applied on the image which are a Soble filter and a Laplacaing filter in addition to a high pass filtered image created by subtracting the low pass filtered image from the original one.

0.1.2 Part 2

In part 2, the two images of "Einstein" and "Marilyn Monroe", shown in figure 4, are used to test the hybrid image creation process due to the possible alignment of their faces in these images.

Figure 1: The original image

Figure 2: Results from Identity filter and Blurring filter respectively

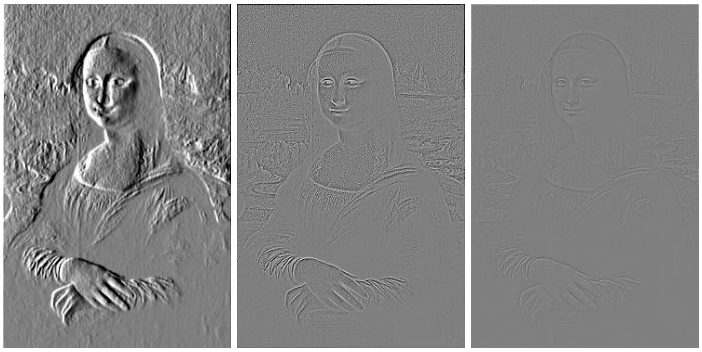

Figure 3: Results from Sobel filter, Laplacian filter, subtraction the low pass content respectively

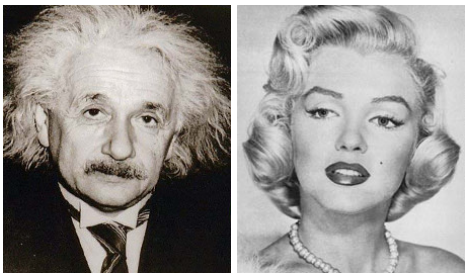

Figure 4: The original images used for hybrid image

A Gaussian filter is applied on both images, the image of ”Einstein” is used as the low frequencies content image though. The resultant low frequencies image of ”Marilyn Monroe” is subtracted from its original image to get the high frequencies image. Then, the hybrid image is created by merging the two filtered images together as shown in figure (5).

Figure 5: Low frequencies image, High frequencies image, Hybrid image

To sense the illusion of the hybrid image, different scales (down-sampled) versions of the hybrid image is created to mimic the effect of different distances and hence different perception of the hybrid image is shown in figure (6).

Figure 6: Differently scaled hybrid image

0.2 Reflective Padding

In the convolution process, different types for padding are used according to the implementation. The most common more padding is the zero padding which is used in our "myimfilter" in this python project as well as reflective padding. Reflective padding is mainly mirroring the pixels lines on the edges in reversed order at the newly created pixels lines (horizontally and vertically) instead of just inserting zeros in them. Figure (7) shows the results of our both zero padding and reflective padding using the "mona lisa" image shown in figure (1). The size of padding is chosen relatively high (25) to be visible.

Figure 7: Zero padding VS Reflective padding

Questions

Q1: Explicitly describe image convolution: the input, the transformation, and the output. Why is it useful for computer vision?

A1: Convolution is an extremely important mathematical operation in signal processing and image processing. Convolution represents applying a certain system effect/response on an input, the system can be a filter or any function to be applied where the input can be an image or a signal. In image processing, the filter is multiplied by some pixels and the resultant value is saved in a corresponding pixel then the filter is shifted towards any direction to repeat the multiplication with the new corresponding pixels and save the new value of multiplication and summation. At the end of the operation, the output (in case of image processing) has the same dimensions of the input image but having the certain effect of the run filter.

Convolution is important in image processing because it is a corner stone in image processing to apply a filter that may blur the signal, remove the noise or extract a certain frequency contents out of the image. There is always some sort of relation between the pixel and the surrounding ones is needed to do so, hence the convolution is used as this property exists in convolution by definition.

Mathematically, convolution is performed by flipping the kernel 180 degree, multiplying the kernel elements by the corresponding image pixels, summation of the multiplication matrix values and then sliding the kernel and repeating as provided by the mathematical equation (1).

$$XY(x,y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} X(i,j)Y(x-i,y-j) \tag{1}$$

Q2: What is the difference between convolution and correlation? Construct a scenario which produces a different output between both operations.

Please use scipy.ndimage.convolve and scipy.ndimage.correlate to experiment!

A2: However being very close in the mathematical equations, convolution and correlation are different in sense of their meaning. Convolution is mainly a mathematical operation results in applying the effect of the filter/kernel on the input image where correlation is a mathematical operation that results in an output represents the similarity between two images. The correlation between to images X,Y can be implemented using equation (2).

$$XY(x,y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} X(i,j)Y(x+i,y+j) \tag{2}$$

For convolution, the same equation can be used just by replacing the positive sign with a negative sign as equation (1).

Therefore, the correlation is identical to convolution in case of symmetric kernel. The following examples (Figures 8,9) show the output of convolution and correlation operations in case of using symmetrical and non-symmetrical kernels.

```
from scipy.ndimage import correlate, convolve
img = cv2.imread('mona_lisa.jpg',cv2.IMREAD_GRAYSCALE)
kernel = [[0.1, 1, -0.1],
          [1, 1, 1],
          [-0.1, 1, 0.1]]
correlated = correlate(img, kernel)
correlated
```

```
array([[100, 120, 163, ..., 165, 168, 166],
       [138, 155, 188, ..., 177, 177, 174],
       [151, 163, 189, ..., 204, 206, 206],
       ...,
       [ 59,  59,  59, ...,  19,  20,  20],
       [ 60,  60,  60, ...,  17,  19,  20],
       [ 60,  60,  60, ...,  16,  19,  20]], dtype=uint8)
```

```
convolved = convolve(img, kernel)
convolved
```

```
array([[100, 120, 163, ..., 165, 168, 166],
       [138, 155, 188, ..., 177, 177, 174],
       [151, 163, 189, ..., 204, 206, 206],
       ...,
       [ 59,  59,  59, ...,  19,  20,  20],
       [ 60,  60,  60, ...,  17,  19,  20],
       [ 60,  60,  60, ...,  16,  19,  20]], dtype=uint8)
```

Figure 8: Convolution and Correlation using symmetric kernel

For better visualization of the difference between correlation and convolution, a proposed test is to extract a small block of an image (non-symmetric one) and then applying both convolution and correlation between the extracted block and the original image. In this test, high values (white areas) will appear extensively at the original place of the extracted

```
from scipy.ndimage import correlate, convolve
img = cv2.imread('mona_lisa.jpg',cv2.IMREAD_GRAYSCALE)
kernel = [[0.1, 1, -0.1],
          [1, 1, 2],
          [0.01, 1, 1]]
correlated = correlate(img, kernel)
correlated
```

```
array([[  0,  37,  98, ...,  81,  81,  79],
       [ 51,  80, 126, ..., 101,  98,  94],
       [ 60,  81, 117, ..., 139, 144, 144],
       ...,
       [ 83,  83,  83, ...,  27,  28,  28],
       [ 84,  84,  84, ...,  25,  27,  28],
       [ 84,  84,  84, ...,  23,  26,  28]], dtype=uint8)
```

```
convolved = convolve(img, kernel)
convolved
```

```
array([[237,   2,  54, ...,  75,  81,  81],
       [ 33,  50,  90, ...,  89,  93,  89],
       [ 59,  72, 103, ..., 124, 131, 130],
       ...,
       [ 82,  82,  82, ...,  27,  28,  28],
       [ 84,  84,  84, ...,  24,  26,  28],
       [ 84,  84,  84, ...,  22,  25,  28]], dtype=uint8)
```

Figure 9: Convolution and Correlation using non-symmetric kernel

part in the main image in case of correlation. However, no sensible image will result from the convolution operation.

Q3: What is the difference between a high pass filter and a low pass filter in how they are constructed, and what they do to the image? Please provide example kernels and output images.

A3: The construction of high pass filters depends mainly on inserting high values at each pixel at which high difference between its neighbours exists and low values at each pixel at which low difference between its neighbours exists. At a pixel with identical neighbours, the filter is designed to insert 0 at this pixel place. In contrast, the construction of low pass filters depends on relating each pixel with its neighbours such as getting their average, this averaging reduces the differences between the successive pixels and hence reduces the high value changes/variations which corresponds to reducing the high frequencies.

As their names indicates and as their construction criteria implies, the difference in high pass filters and low pass filters effect on images is that the low pass filters reduces (or eliminates) the high frequencies and the high pass filters eliminate the low frequencies. The effect of reducing high frequencies can be visualized in blurring and eliminating most of the edges as the edges are the straight sort of high frequency content in the image. For low frequencies elimination, the edges are the most unaffected part of the image where the smooth variations in the image are mostly reduced or eliminated.

To test low pass filters and high pass filters, the previously mentioned criteria is applied to propose a low pass filter and a high pass filter as shown in figure (10). Figure (11) shows the difference among the real image and its correspondence after using low pass filter and high pass filter.

The python script for this question can be found *here*.

```
blur_filter
```

```
array([[0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04]], dtype=float32)
```

```
high_filter
```

```
array([[ 0.,  1.,  0.],
       [ 1., -4.,  1.],
       [ 0.,  1.,  0.]], dtype=float32)
```

Figure 10: Proposed low pass and high pass filters

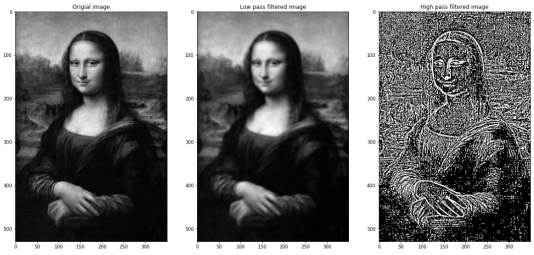

Figure 11: The effect of low pass and high pass filters on the image

Q4: How does computation time vary with filter sizes from 3×3 to 15×15 (for all odd and square sizes), and with image sizes from 0.25 MPix to 8 MPix (choose your own intervals)? Measure both using $scipy.ndimage.convolve$ or $scipy.ndimage.correlate$ to produce a matrix of values. Use the $skimage.transform$ module to vary the size of an image. Use an appropriate charting function to plot your matrix of results, such as $Axes3D.scatter$ or $Axes3D.plot\_surface$.

Do the results match your expectation given the number of multiply and add operations in convolution?

A4: Increasing filter size or image size results in more computations in convolution or correlation. For constant image size, the sort of computation increasing is the higher number of multiplications preformed by each filter shifting operation. For constant filter size, the sort of computation increasing is the higher number of summations and multiplications performed along the image itself. A quick python script was written and run to visualize the time elapsed, as an indication of the computation complexity, for convolution operation using different image sizes and kernel sizes and the results are shown in figure (12).

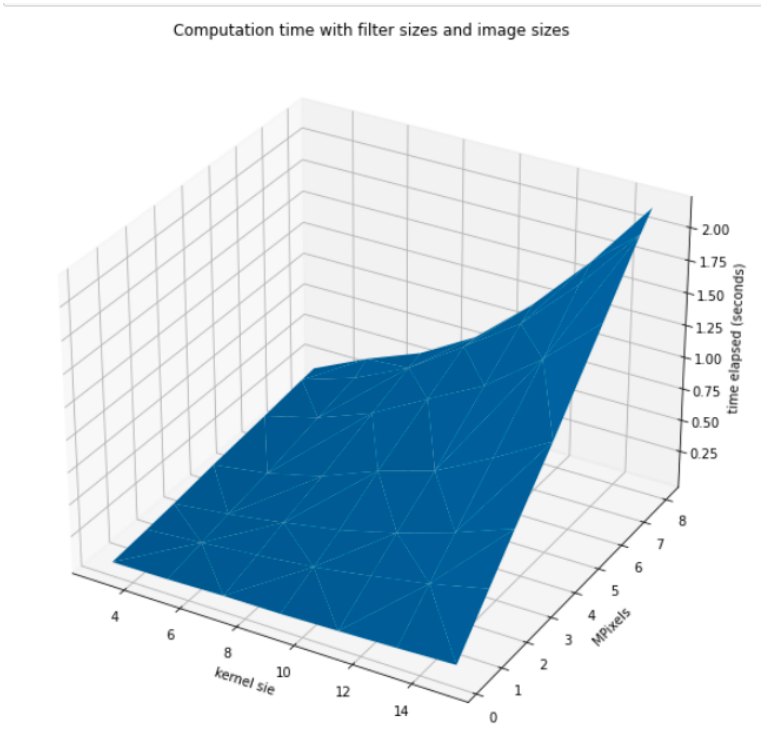

Figure 12: Computation time VS Filter size VS Image size

The python script for this question can be found *here*.