Homework 2 Questions

Instructions

- 4 questions.
- Write code where appropriate.
- Feel free to include images or equations.
- **Please use only the space provided and keep the page breaks.** Please do not make new pages, nor remove pages. The document is a template to help grading.
- If you really need extra space, please use new pages at the end of the document and refer us to it in your answers.

Questions

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

A1: The input is the image we originally had, and the output is the image we want to create according to the filter. Image convolution uses the calculations shown in the following image. In this case, h is the output image, I is the input image, and f is the filter.

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

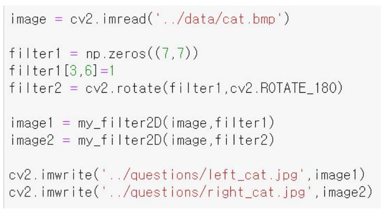
Figure 1: Convolution

We can enlarge, reduce, sharpen or blur the original image. Like the hybrid image in this homework, we can create various images depending on the purpose.

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

A2: Convolution is equivalent to correlation by rotating the filter 180 degrees. Therefore, if the filter is rotated 180 degrees, and the filter is the same as before, the image using convolution and the image using correlation will produce the same result. But if they are different, we will get different images.

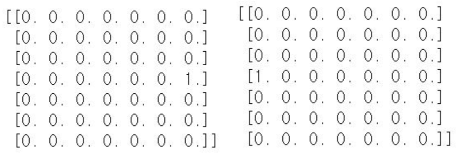
For example, after creating myfilter2D function in homework, I tried running the following code.



```
image = cv2.imread('../data/cat.bmp')

filter1 = np.zeros((7,7))
filter1[3,6]=1
filter2 = cv2.rotate(filter1,cv2.ROTATE_180)

image1 = my_filter2D(image,filter1)
image2 = my_filter2D(image,filter2)

cv2.imwrite('../questions/left_cat.jpg',image1)
cv2.imwrite('../questions/right_cat.jpg',image2)
```

Figure 2: Convolution vs. Correlation code

Filter 1 and filter 2 are as follows.



```
[[0. 0. 0. 0. 0. 0. 0.]   [[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]    [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]    [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1.]    [1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]    [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]    [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]   [0. 0. 0. 0. 0. 0. 0.]]
```

Figure 3: *Left:* Filter1 *Right:* Filter2

The following images were produced according to each filter.

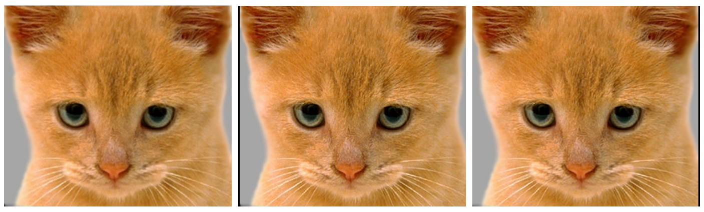


Figure 4: *Left:* Original image *Middle:* Filter1 image *Right:* Filter2 image

In the picture above, the picture on the middle moves to the right, and the picture on the right moves to the left. If the filter is rotated 180 degrees like this, the image may come out differently.

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: By performing Fourier transform of the image, the amplitude and edge according to frequency can be obtained. In this case, the frequency indicates how often the pixel value of the image changes.

The high pass filter is the filter that passes the part with the high frequency. Because the frequently changing part of the image passes, we can get the edge of the image through the high pass filter.

The typical high pass filter is Laplacian filter. Because it makes the edge of the picture sharper, the high frequency part passes through.

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

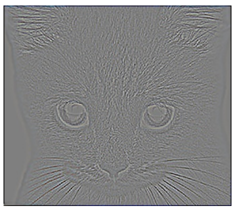


Figure 5: *Left:* Example of Laplacian filter *Right:* High pass filter image

The low pass filter, on the other hand, is the filter that passes the low frequency part. Because the sparse part of the image passes, we can see the overall appearance of the image through the low pass filter.

The typical low pass filter is Gaussian filter. It lowers the interval values beween pixels, so the low frequency part passes through.

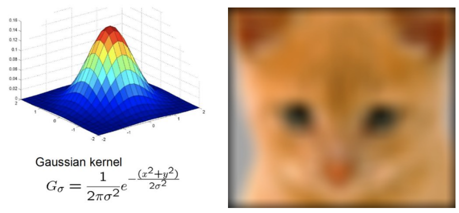


Figure 6: *Left:* Example of gaussian filter and equation *Right:* Low pass filter image

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 $cv2.filter2D()$ to produce a matrix of values. Use the $cv2.resize()$ function to vary the size of an image. Use an appropriate 3D charting function to plot your matrix of results, such as $plot\_surface()$ or $contour3D$.

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

See RISDance.jpg in the attached file.

A4: I wrote computation.py code

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
from mpl_toolkits import mplot3d

Filter = []
for i in range(3,17,2):
  Filter.append(np.random.randn(i, i))

Image = []
Image.append(cv2.imread('../questions/RISDance.jpg'))
for i in range(1,5):
  Image.append(cv2.resize(Image[0],dsize=(0, 0),
     fx=1-i*0.2, fy=1-i*0.2))

ImageSize = []
for i in range(5):
  ImageSize.append((Image[i].shape[0])*(Image[i].shape[1])/1000000)

Measure = [0,0,0,0,0,0,0]
for i in range(7):
  Measure[i] = [0,0,0,0,0]
for i in range(7):
  for j in range(5):
    start_time = time.time()
    output = cv2.filter2D(Image[j],-1,Filter[i])
    Measure[i][j] = time.time() - start_time


x = np.array(ImageSize)
y = np.linspace(3,15,7)
X,Y = np.meshgrid(x,y)
Z = np.array(Measure)

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_title('Computation Time')
ax.set_xlabel('Image Size (MPix)')
ax.set_ylabel('Filter Width (pixel)')
ax.set_zlabel('Computation Time')

plt.show()
```

And, the result is the figure below.

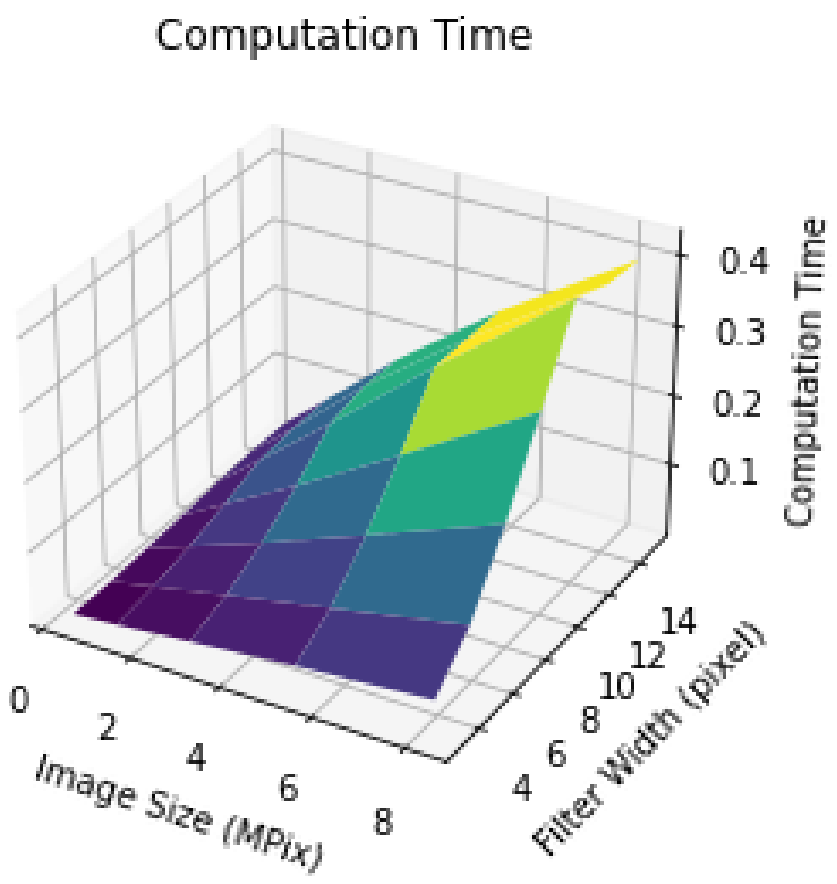


Figure 7: Computation time

If filter is not separable, the computation time is about MNPQ in M*N image, P*Q filter. The figure above is similar to the expectation. However, the increase rate is slightly lower when the filter width is very large.