Homework 2 Questions

Instructions

- 4 questions.
- Write code where appropriate.
- Feel free to include images or equations.
- Please make this document anonymous.
- **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: Your answer here.

input : an imege and small matrix called the kernel or convolution matrix.

transformation : $g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l)$ (f is an image, h is kernel) , This equation also written $g = f * h$

output : new image g

We can use convolution method to detect important features in an image. In addition, other images can be predicted or generated through the relationship between the two images using convolution method. It is also a necessary concept in the CNN lyaer model. Therefore convlution is useful for computer vision because it is used in many places in computer vision.

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

Please use imfilter to experiment! Look at the 'options' parameter in MATLAB Help to learn how to switch the underlying operation from correlation to convolution.

A2: Your answer here.

Convolution is commutative ($a * b = b * a$) and associative ($a * (b * c) = (a * b) * c$), but Correlation is not commutative and associative. Both operations are identical when the filter is symmetric, but different when the filter is non-symmetric.

```
test_image = im2single(imread('../data/einstein.bmp'));
filter = [-1 0 1; -2 0 2; -1 0 1];

image = imfilter(test_image, filter, 'corr');
figure(1); imshow(image);

image = imfilter(test_image, filter, 'conv');
figure(2); imshow(image);

filter = rot90(filter, 2);
image = imfilter(test_image, filter, 'corr');
figure(3); imshow(image);
```

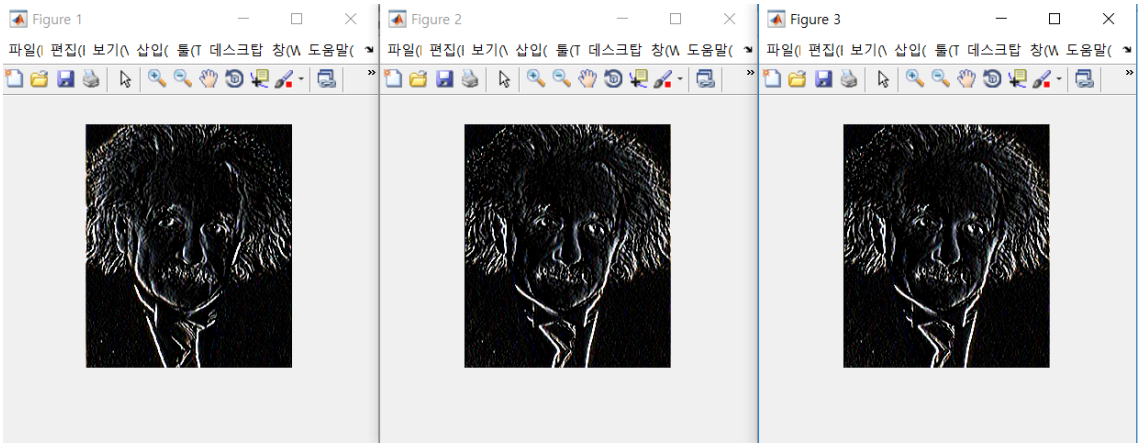

Figure 1: Result of avobe code

However, if one filter is rotated 180 degrees, the two operations become identical. It can be seen that figure 1 differs from the others, and figure 2 and figure 3 show the same results.

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: Your answer here.

The low pass filter is constructed of a filter that blur images such as Gaussian blur filters. So it allows only the low frequency portion of the original image to remain. However, the high pass filter allows only the high frequency portion of the original image to remain. Examples of high pass filter are subtract the low frequency content from the original image, discrete Laplacian filter etc.

```
test_image = im2single(imread('../data/submarine.bmp'));
figure(1); imshow(test_image);

blur_filter = fspecial('Gaussian', 31, 7); %Example of
    low pass filter
blur_image = imfilter(test_image, blur_filter, 'conv');
figure(2); imshow(blur_image);

laplacian_filter = [0 1 0; 1 -4 1; 0 1 0]; %Example of
    high pass filter
laplacian_image = imfilter(test_image, laplacian_filter,
    'conv');
figure(3); imshow(laplacian_image);
```

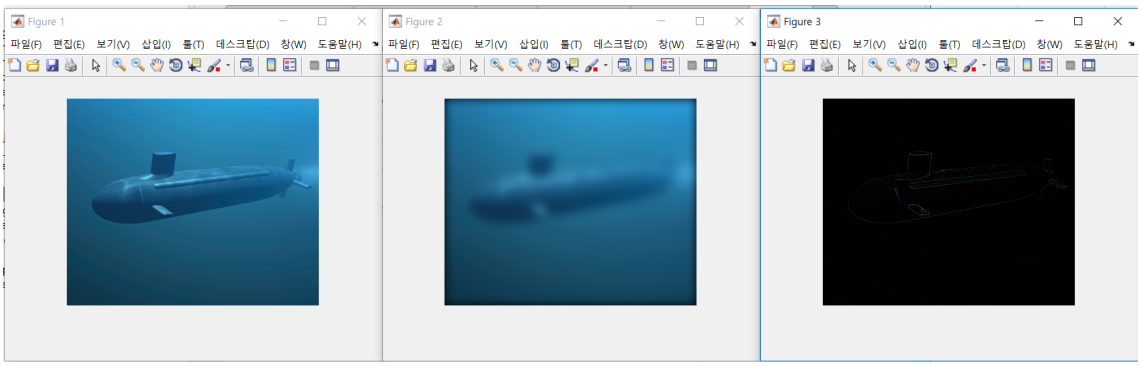

Figure 2: Result of avobe code, left : original, middle : low pass filter, right : high pass filter

In low pass filter case, because the high frequency part disappears, we can see that the boundary is blurred. In contrast, in high pass filter case, because only the high frequency part remains, we can see that the boundaries have become clear.

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 *imfilter* to produce a matrix of values. Use the *imresize* function to vary the size of an image. Use an appropriate charting function to plot your matrix of results, such as *scatter3* or *surf*.

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

See RISDance.jpg in the attached file.

A4: Your answer here.

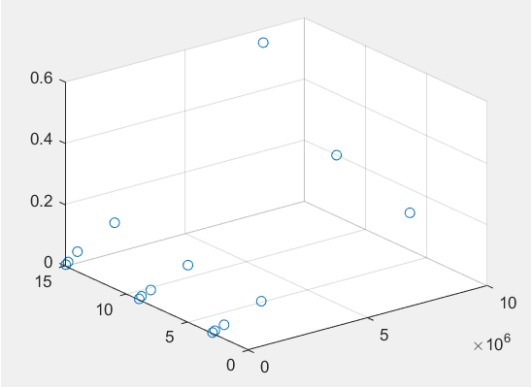

Figure 3: axis that consist x10^6 term is pixel number of image, another axis of ground is size of one side of filter, z-axis is time

Considering the convolution formula when the size of the image is n * m and the size of the filter is k * l, we can see that the time complexity is proportional to n * m * k * l. So, we can assume n*m is pixel number in x-axis in above picture, k = l, k or l is y-axis in above picture, and z-axis is n*m*k*l instead time.

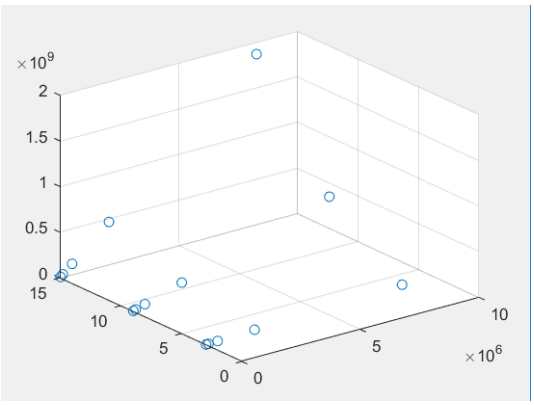

Figure 4: graph of above thought

We can see that the two graphs are very similar.

```
img = im2single(imread('../questions/RISDance.jpg'));
disp(size(img));
filter1 = fspecial('Gaussian', 3, 5); filter2 = fspecial(
   'Gaussian', 9, 6); filter3 = fspecial('Gaussian', 15,
   6);
x = []; y = []; z = [];
for i=1:5
    [n,m,c] = size(img);
    p = n*m;
    x = [x p]; y = [y 3];
    tic
    imfilter(img, filter1, 'conv');
    toc
    z = [z toc];
    x = [x p]; y = [y 9];
    tic
    imfilter(img, filter2, 'conv');
    toc
    z = [z toc];
    x = [x p]; y = [y 15];
    tic
    imfilter(img, filter3, 'conv');
    toc
    z = [z toc];  img = imresize(img, 0.5, 'bilinear');
end
scatter3(x,y,z)
```

code that make figure 3

```
img = im2single(imread('../questions/RISDance.jpg'));
x = []; y = []; z = [];
for i=1:5
    [n,m,c] = size(img);
    p = n*m;
    x = [x p]; y = [y 3];
    z = [z p*3*3];
    x = [x p]; y = [y 9];
    z = [z p*9*9];
    x = [x p]; y = [y 15];
    z = [z p*15*15];  img = imresize(img, 0.5, 'bilinear'
       );
end
scatter3(x,y,z)
```

code that make figure 4