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:
Convolution is used in image processing to apply a filter to an image and produce a desired output image. The output is simply a matrix of values, or pixel values, that are linear combinations of input pixel values. The transformation matrix is used to form linear combinations of input pixel values.
This is useful for computer vision because it can be used in image processing - blurring, sharpening, edge detecting, and so on. It gives us ability to manipulate pixel information and thus the image itself.

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:
Convolution and correlation both are filtering techniques used to produce output pixels that depend on neighboring input pixels. Convolution, however, uses a filter that has been rotated by 180 degrees from the one used by correlation. Also, convolution is interested in obtaining an output image from an input image and transformation information, whereas correlation is interested in determining the similarities between input information and target information.

Code:

```
kernel = [8 1 6; 3 5 7; 4 9 2];
image = zeros(5);
image(:) = 1:25;

corr_output = imfilter(image, kernel);
conv_output = imfilter(image, kernel, 'conv');
conv_output
corr_output
```

Convolution output:

```
conv_output =

81   185   335   485   341
131   315   540   765   581
161   360   585   810   611
191   405   630   855   641
127   275   425   575   519
```

Correlation output:

```
corr_output =

79   205   355   505   419
139   315   540   765   589
169   360   585   810   619
199   405   630   855   649
153   295   445   595   361
```

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:
High pass filter is constructed by making sum of all values in the filter 0. In high pass filter, values can be either positive, negative, or 0. Most high pass filters are consisted of a single positive value at the center surrounded by negative values. High pass filter tends to retain high frequency information while reducing low frequency information, thus sharpening the image. It does so by increasing brightness of center pixel relative to neighboring pixels. It also amplifies noise.

Low pass filter is constructed by making sum of all values in the filter 1. In low pass filter, values are positive or 0. Most low pass filters are constructed to decrease the disparity between neighboring pixel values. Low pass filter tends to retain low frequency information while reducing high frequency information, thus blurring the image. It does so by smoothing the image, or making the neighboring pixels closer to each other. It also smoothes out noise.

Example high pass filter: $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Example low pass filter: $\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$

Code:

```
hpf = [-1 -1 -1; -1 8 -1; -1 -1 -1];
lpf = [1/9 1/9 1/9; 1/9 1/9 1/9; 1/9 1/9 1/9];

image = imread('RISDance.jpg');
HPF = imfilter(image, hpf, 'same');
LPF = imfilter(image, lpf, 'same');
imwrite(HPF, 'HPF.jpg');
imwrite(LPF, 'LPF.jpg');
```

Input image: RISDance.jpg
High pass filtered image: HPF.jpg
Low pass filtered image: LPF.jpg

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:
Code:

```
row = 1;
col = 1;
ret = double(zeros(7, 32));

for i1 = 3:2:15 %row
    col = 1;
    for i2 = 0.25:0.25:8.0 %col
        filter = zeros(i1, i1);
        ratio = (i2 * 1000000) / numel(image);
        image2 = imresize(image, ratio);
        tic;
        result = imfilter(image2, filter);
        elapsed = toc;
        ret(row, col) = elapsed;
        col = col + 1;
    end
    row = row + 1;
end
surf(ret)
```

Computation time chart: computation.png

My expectation was that it would take longer time to compute convolution if size of image increases or if size of filter increases. Generally, this expectation was correct. However, the rate at which computation time increases was much more faster when size of image increases than when size of filter increases.