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 the process of adding the elements of an image to its neighbors by the weights defined in a small matrix called the kernel. The symbol used for convolution is the matrix multiplication operator ”*”, however, it is different from traditional matrix multiplication. The input of image convolution is the image itself and the kernel. The transformation is performed by the kernel by calculating the weighted sum of elements. The output is the transformed image whose elements are the sums calculated with the kernel. Image convolution is useful in computer vision because filters such as edge detection and blur filter can easily be applied to an image using convolution.

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 is the measure of effect of one signal to the other, while correlation is the measure of similarity between two signals. Below is the code (*compare_imfilter.m*) that I used to compare the images generated by different options of imfilter using a Sobel filter. Also, the resulting images are shown in Figure 1. The direction of the colored pixels are different so it seems as if the direction of lighting is different.

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

convolution = imfilter(test_image, filter, 'conv');
correlation = imfilter(test_image, filter, 'corr');

imshow(convolution);
figure();
imshow(correlation);
```

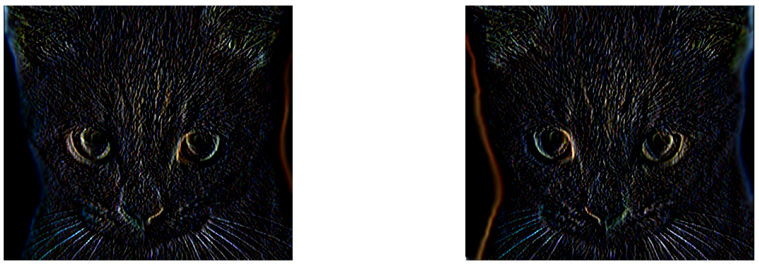

Figure 1: *Left:* Convolution option. *Right:* Correlation option.

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: A high pass filter retains the high frequency information while reducing the low frequency information, and vice versa for a low pass filter. A low pass filter blurs the image by reducing the disparity between pixel values. A high pass filter sharpens the image by increasing the brightness of the center pixel relative to neighboring ones. A low pass filter can by created by dividing a matrix of ones by its size. A high pass filter can be created from a matrix that contains negative values with a positive only at the center. Examples of low pass and high pass filters shown in the code below (*low_high_pass.m*).

```
low_pass = [1 1 1; 1 1 1; 1 1 1]/9;
high_pass = [1 1 1; 1 1 1; 1 1 1]/-9;
high_pass(2,2)=8/9;

test_image = im2single(imread('../data/cat.bmp'));
low = imfilter(test_image, low_pass, 'conv');
high = imfilter(test_image, high_pass, 'conv');

imshow(low);
figure();
imshow(high+0.5);
```

Figure 2: *Left:* Low pass filtered *Right:* High pass filtered.

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: The code used to measure computation times (*computation_time.m*) is shown below along with the result plot in Figure 3. The results match my expectations because the elapsed time is proportional to the image size and filter size. The number of operations is directly proportional to them.

```
% ... initializations ...
i_idx=1;
for i=filter_sizes
    j_idx=1;
    filter=ones(i)/i^2;
    for j=image_sizes
        scaled=imresize(image, j);
        tic;
        filtered=imfilter(scaled, filter, 'conv');
        t=toc;
        result(i_idx, j_idx) = t;
        j_idx=j_idx+1;
    end
    i_idx=i_idx+1;
end
surf(filter_sizes, image_sizes, result)
```

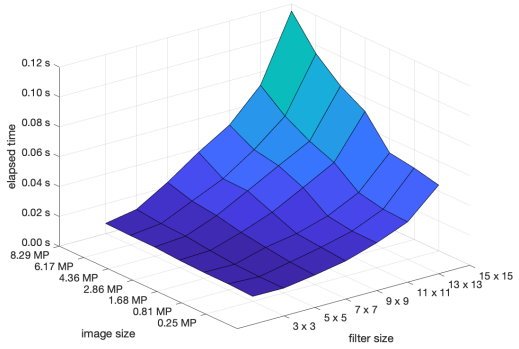

Figure 3: Computation time measurements.