Homework 2 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Image Filtering

I used Fast Fourier Transformation to implement image filtering.

First, I padded the image with zeros according to the size of filter. For top and bottom of the pixel matrix, ”floor(filter_size(1) / 2)” rows of 0 were added, and for right and left of the pixel matrix, ”floor(filter_size(2) / 2)” columns of 0 were added (filter_size = size(filter)). I also padded the filter using the same size and put the actual filter on the upper left corner.

Then, I ran a loop k many times, where dimension of the image is m $\times$ n $\times$ k. So, if the image is grayscale, the loop is run 1 time, and if the image is colored, the loop if run 3 times. Inside the loop, I found Fast Fourier Transformation of padded image and padded filter using fft2 function. I then found padded convolution by multiplying elements of padded image and padded filter. I then used ifft2 function to find image of that product and found appropriate portion to unpad the image.

Shown below is the code snippet:

```
filter_size = size(filter);
if rem(filter_size(1), 2) == 0 || rem(filter_size(2), 2)
   == 0
    error("Error: even-sized filter!");
end

filter_row = floor(filter_size(1)/2);
filter_col = floor(filter_size(2)/2);
image_size = size(image);

pimage = padarray(image, [filter_row, filter_col]);
pimage_size = size(pimage);
pfilter = zeros(pimage_size(1), pimage_size(2));
```

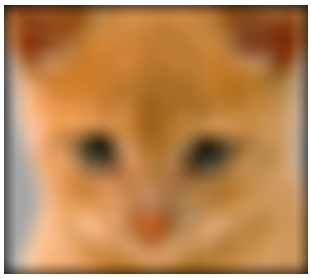
```
13 pfilter(1:filter_size(1), 1:filter_size(2)) = filter;
14 ret = zeros(image_size);
15
16 for k = 1:image_size(3)
17     f_image = fft2(pimage(:,:,k));
18     f_filter = fft2(pfilter);
19     padded_conv = ifft2(f_image .* f_filter);
20     ret(:,:,k) = padded_conv(2 * filter_row + 1 :
           pimage_size(1), 2 * filter_col + 1 : pimage_size
           (2));
21 end
22 output = ret;
```

Shown below are images that resulted from this code:

(a) Original picture: cat.jpg

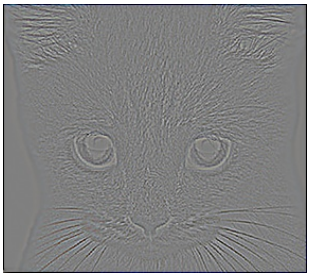
(b) Identity filter: identity_image.jpg

(c) Small blur with box filter: blur_image.jpg

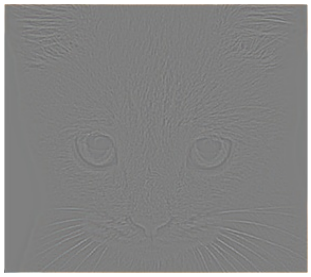


(d) Large blur: large_blur_image.jpg

(e) Oriented filter (Sobel Operator): sobel_image.jpg



(f) High pass filter (Discrete Laplacian): laplacian_image.jpg



(g) High pass ”filter” alternative: high_pass_image.jpg

Hybrid Image

For part 2, I implemented the function gen_hybrid_image(image1, image2, cutoff_frequency). This function first filters image1 through a low pass filter and image2 through a high pass filter. Then, it adds those two outputs up to create a new hybrid image.

First, I created a square n $\times$ n Gaussian filter, where n is 2 $\times$ cutoff_frequency + 1, and standard deviation is cutoff_frequency. Then, low_frequencies was calculated by filtering image1 through the filter, creating a blurred version of image1. Likewise, high_frequencies was calculated by subtracting blurred version of image2 - that was created by filtering image2 with filter - from image2 itself. Finally, I simply added those two matrices up to get hybrid_image.

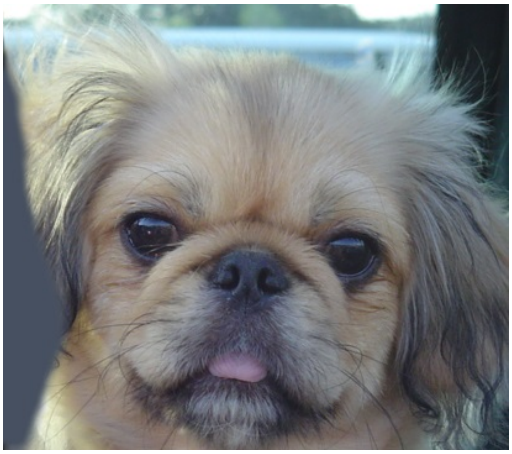
Shown below is the code snippet:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove high frequency to produce low frequencies only
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filter = fspecial('Gaussian', cutoff_frequency*2+1,
   cutoff_frequency);
high_pass_image = my_imfilter(image1, filter);
high_pass_image = my_imfilter(high_pass_image, filter');
low_frequencies = [high_pass_image];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove low frequency to produce high frequencies only
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
low_pass_image = my_imfilter(image2, filter);
low_pass_image = my_imfilter(low_pass_image, filter');
low_pass_image = image2 - low_pass_image;
high_frequencies = [low_pass_image];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Combine the high frequencies and low frequencies
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hybrid_image = [high_pass_image + low_pass_image];
```

Shown below are combinations of image1 = cat and image2 = dog:



(a) Image1:
dog.jpg

(b) Image1 low frequencies:
low_frequencies.jpg

(c) Image2:
cat.jpg

(d) Image2 high frequencies:
high_frequencies.jpg

(e) Two image combined:
hybrid_image.jpg

(f) Two image combined scaled:
hybrid_image_scales.jpg