Project 1 Writeup

Qin Yi
59207768

November 4, 2017

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.

FFT-based convolution

If we use Fast Fourier Transformation (FFT), we have to be careful with the way to pad the filter with zeros, which is to make the real centre of the filter matrix at the up-left corner of the padding matrix, and separately make the four parts at the corresponding corner.

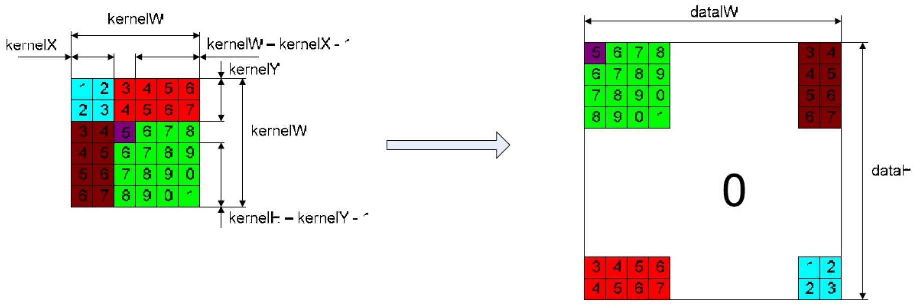

Figure 1. Expansion of the convolution kernel to the image size: cyclically shift the original convolution kernel, so that the central element of the kernel is at (0, 0) [1]

```
% Pad the input image with zeros.
padding = zeros(a,b);
mm = round(m/2);
nn = round(n/2);
padding(1:mm,1:nn) = filter(m-mm+1:m,n-nn+1:n);
padding(1:mm,b-(n-nn)+1:b) = filter(m-mm+1:m,1:n-nn);
padding(a-(m-mm)+1:a,1:nn) = filter(1:m-mm,n-nn+1:n);
padding(a-(m-mm)+1:a,b-(n-nn)+1:b) = filter(1:m-mm,1:n-nn
    );
```

The basic outline of FFT-based convolution is:

- Apply direct FFT to the image,
- Apply direct FFT to the zero-padded filter,
- Perform the point-wise multiplication of the two preceding results,
- Apply inverse FFT to the result of the multiplication.

```
% FFT-based convolution.
fft_image = fft2(image);
[a,b,c] = size(fft_image);
```

```
fft_filter = fft2(padding);
fft_output = fft_image .* fft_filter;
output = ifft2(fft_output);
```

We know the Fourier Transformation can simplify the convolution by:

$$f * g = F \cdot H,$$

so that for matrix, we have:

$$(f * g)[i,j] = \sum_{k,l} f[i-k, j-l]g[k,l],$$

$$(F \cdot H)[i,j] = F[i,j] \cdot H[i,j].$$

If the image is not too small, the direct convolution has much higher computation complexity than FFT-based convolution, even though there should be two Fourier Transformations and one Inverse Fourier Transformation.

Interesting Implementation Detail

When we do the filtering job, we have to notice that even shaped filters' output is undefined.

```
% Return an error message for even filters.
[m,n] = size(filter);
if mod(m*n,2)==0
    error('Error: even filters');
    return
end
```

Hybrid Image

It's just like to cut of each image into low-passed part and high-passed part, and then remerge one image's low-passed part with the other's high-passed part. The amount of blur that works best will vary with different image pairs, so we need to tune the $cutoff\_frequency$ (The standard deviation, in pixels, of the Gaussian blur that will remove high frequencies) in the real implementation part for each pair.

```
% Remove the high frequencies from image1 by blurring it.
low_frequencies = my_imfilter(image1, filter);
% Remove the low frequencies from image2.
blur_image2 = my_imfilter(image2, filter);
high_frequencies = image2 - blur_image2;
% Combine the high frequencies and low frequencies
hybrid_image = low_frequencies + high_frequencies;
```

Results and Parameters

My implementations, including which to high/low pass and the cutoff_frequency, are summarized as following:

low-pass image	high-pass image	cutoff_frequency
dog	cat	7
bicycle	motorcycle	10
bird	plane	4
submarine	fish	3
marilyn	einstein	4

Table 1: Five results.

Resulting pictures are in the following pages.

References

[1] FFT-based 2D convolution, Victor Podlozhnyuk, vpodlozhnyuk@nvidia.com

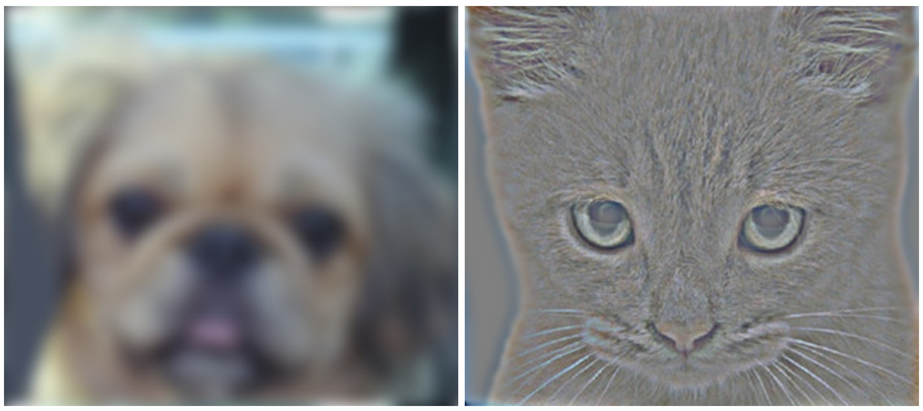

Figure 1: *Left:* low frequencies dog. *Right:* high frequencies cat.

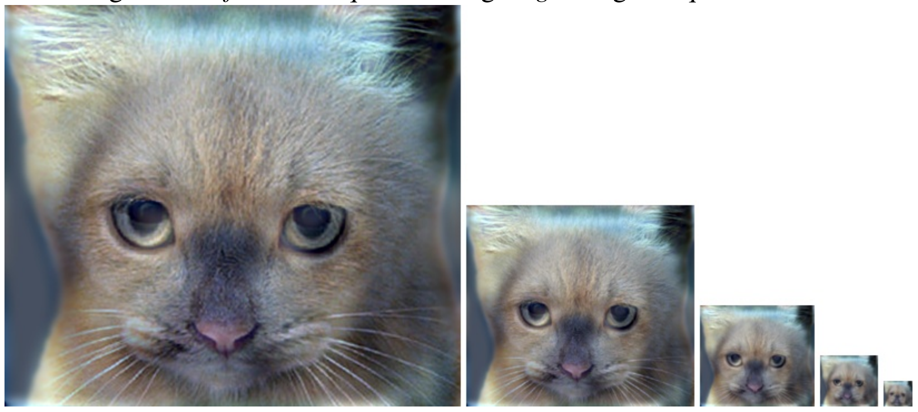

Figure 2: hybrid image: dog & cat

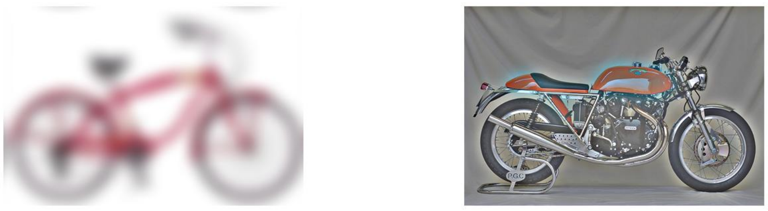

Figure 3: *Left:* low frequencies bicycle. *Right:* high frequencies motorcycle.

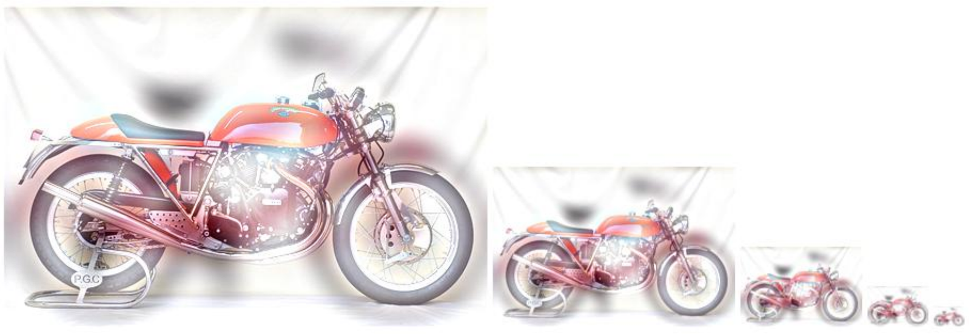

Figure 4: hybrid image: bicycle & motorcycle

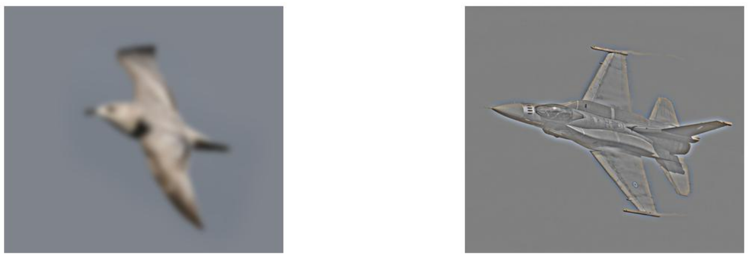

Figure 5: *Left:* low frequencies bird. *Right:* high frequencies plane.

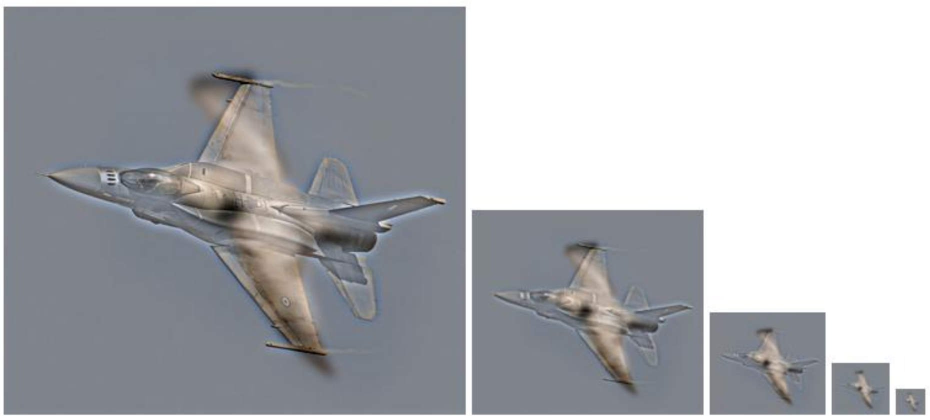

Figure 6: hybrid image: bird & plane

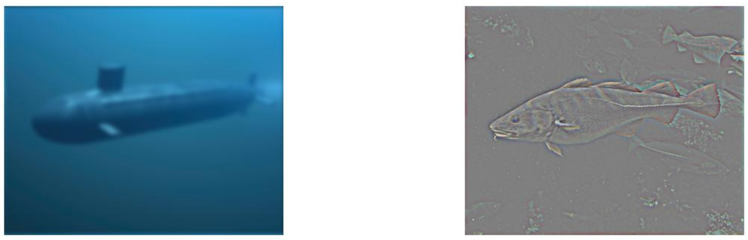
Figure 7: *Left:* low frequencies submarine. *Right:* high frequencies fish.

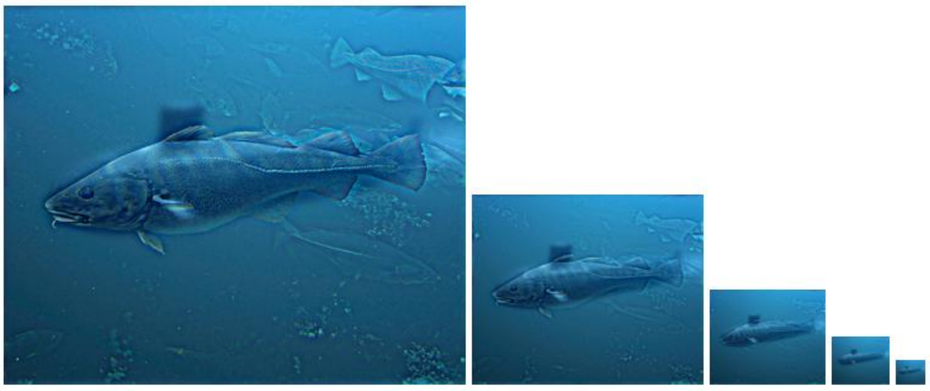
Figure 8: hybrid image: submarine & fish

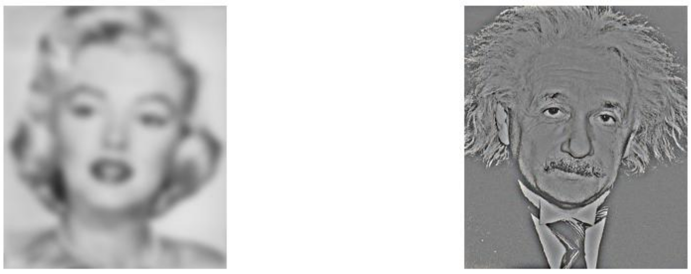

Figure 9: *Left:* low frequencies marilyn. *Right:* high frequencies einstein.

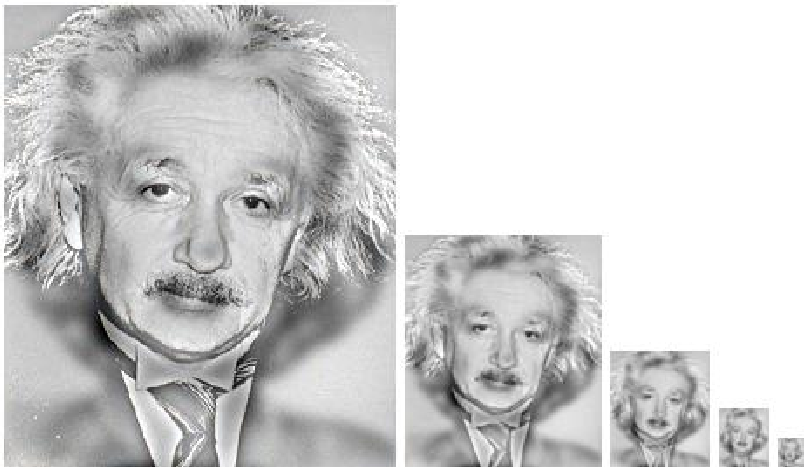

Figure 10: hybrid image: marilyn & einstein